Proceedings of the European Workshop on Mixed Integer Nonlinear Programming

12-16 April 2010 - CIRM - Marseille - France

Pierre Bonami [1]
Leo Liberti [2]
Andrew J. Miller [3]
Annick Sartenaer [4]

[1]LIF, Université de la Méditerranée, Marseille, France. pierre.bonami@lif.univ-mrs.fr
[2]LIX, École Polytechnique, Palaiseau, France. liberti@lix.polytechnique.fr
[3]IMB, Université de Bordeaux 1, France. andrew.miller@math.u-bordeaux1.fr
[4]Dept. of Maths, Université de Namur, Belgium. annick.sartenaer@fundp.ac.be

We are grateful to our sponsors

We were able to keep the registration for EWMINLP 2009 free of charge due to generous contributions of our sponsors. The organizing committee wishes to gratefully acknowledge the financial support of our sponsors, listed below. We also want to warmly thank CIRM for welcoming and supporting this conference. Finally we would like to thank the LIF staff for their help in setting up the conference.

IBM Research

LIX, École Polytechnique

LIF, Marseille

Univ. de la Méditéranée

Conference schedule

4/12/2010	Mon	4/13/2010	Tue	4/14/2010	Wed
0850-0900	OPENING				
0900-0945	**Linderoth**	0900-0945	**Leyffer**	0900-0945	**Wiegele**
0945-1015	Bienstock	0945-1015	Exler	0945-1030	**Piccialli**
1015-1045	*coffee*	1015-1045	*coffee*	1030-1100	*coffee*
1045-1130	**Grossmann**	1045-1130	**Toint**	1100-1145	**Malick**
1130-1200	Nannicini	1130-1200	Gentile	1145-1230	**Sotirov**
1200-1230	*discussion*	1200-1230	*discussion*	afternoon	free
1230-1400	*lunch*	1230-1400	*lunch*		
1400-1445	**Belotti**	1400-1445	**Adjiman**		
1445-1515	*coffee*	1445-1515	*coffee*		
1515-1600	**Westerlund**	1515-1600	**Waechter**		
1600-1645	**Tawarmalani**	1600-1630	**Guignard**		
from 1700	posters/cocktail	1630-1700	*discussion*		

Thu	4/16/2010	Fri	4/15/2010
0900-0945	**Kocvara**	0900-0945	**Lee**
0945-1015	Sager	0945-1015	Gleixner
1015-1045	*coffee*	1015-1045	*coffee*
1045-1130	**Messine**	1045-1130	**Anstreicher**
1130-1200	Fortz	1130-1200	D'Ambrosio
1200-1230	*discussion*	1200-1230	*discussion*
1230-1400	*lunch*	1230-1400	*lunch*
1400-1445	**Lasserre**	1400-1445	**Weismantel**
1445-1515	*coffee*	1445-1515	*coffee*
1515-1545	Martin-Campo	1515-1520	CLOSING
1545-1630	**Lodi**		
1630-1700	*discussion*		

`http://sites.google.com/site/ewminlp/`

Preface

The EWMINLP workshop establishes itself as the second of a series, the first having been the 2008 IMA MINLP workshop, co-organized by Jon Lee and Sven Leyffer in Minneapolis, where all of the co-editors of this volume met. We all retain fond memories of that wonderful experience: the high scientific standards, Jeff Linderoth's opening talk on MINLP and the Star Wars saga (including the wonderful remark from the audience, "Couldn't you use the source?"), but most of all the endless pitchers of high quality beers from minnesotan microbreweries. The only trouble with the IMA MINLP workshop was the outside temperature, set well below freezing in November. For the present workshop we decided to keep most of the good features, but to set the scenery in a landscape of staggering beauty, just outside Marseille, France, at a time where the temperature *should* spare us from freezing. From the scientific point of view we also decided to make the workshop somewhat more open, calling for a limited number of contributed talks. As to the beer, we are well-funded, and we promise to be well-stocked by the opening session.

P. Bonami
L. Liberti
A. Miller
A. Sartenaer

Marseille, March 2010

Contents

PART 1

Invited Lectures

1

A comparative study of molecular design formulations based on Generalized Disjunctive Programming

Claire Adjiman

Centre for Process Systems Engineering
Department of Chemical Engineering
Imperial College London
London, SW7 2AZ, United Kingdom

c.adjiman@imperial.ac.uk

Keywords: computer-aided molecular design, generalized disjunctive programming.

The design of mixtures, e.g. optimal product formulations or solvents in the pharmaceutical and personal care industries, is an important and challenging problem [**1**]. The problem has traditionally been posed as a Mixed-Integer Nonlinear Programming problem in which an optimal molecular structure is sought based on a performance metric and set of constraints. The molecule is built based on a set of atom groups which are constrained to meet chemical feasibility constraints. The problem is typically highly combinatorial: there can be several thousand molecular structures in the feasible region defined by the chemical feasibility constraints [**2**]. In addition, at the heart of the model, one often finds highly nonlinear equations that relate the structure of the molecule, as described by integer variables, to its physical properties. For instance, it may be necessary to solve phase equilibria equations. Finally, the optimisation of mixtures is often of interest. In this case, the composition of the mixture becomes an additional, continuous, design variable [**3, 4**]. Even when the components of the mixture are known a priori, the problem is highly combinatorial and nonlinear, due to the complex relationships between composition, structure and physical properties.

These difficulties have hampered the formulation and solution of the mixture design problem. In this paper, we present novel formulations of the mixture design problem based on Generalized Disjunctive Programming (GDP) [**5, 6**] and demonstrate the effectiveness of this approach on case studies. GDP offers a natural way to formulate the problem. We follow a systematic approach, considering increasingly complex formulations: we start with mixtures with a fixed number of components and progress to optimise both the number of components in the mixture as well as the components themselves.

References

1. L. E. K. Achenie, R. Gani, and V. Venkatasubramanian, editors. *Computer-Aided Molecular Design: Theory and Practice*. Elsevier Science, Amsterdam, 2003.
2. M. Folić, C. S. Adjiman, and E. N. Pistikopoulos. Design of solvents for optimal reaction rate constants . *AIChE Journal*, 53:1240–1256, 2007.
3. A. T. Karunanithi, L. E. K. Achenie, and R. Gani. A new decomposition-based computer-aided molecular/mixture design methodology for the design of optimal solvents and solvent mixtures. *Industrial & Engineering Chemistry Research*, 44(13):4785–4797, 2005.
4. A. T. Karunanithi, L. E. K. Achenie, and R. Gani. A computer-aided molecular design framework for crystallization solvent design. *Chemical Engineering Science*, 61(4):1247–1260, 2006.
5. S. Lee and I. E. Grossmann. New algorithms for nonlinear generalized disjunctive programming. *Computers & Chemical Engineering*, 24(9-10):2125–2141, 2000.
6. S. Lee and I. E. Grossmann. Logic-based modeling and solution of nonlinear discrete/continuous optimization problems. *Annals of Operations Research*, 139(1):267–288, 2005.

Comparing convex relaxations for Quadratically Constrained Quadratic Programs

Kurt M. Anstreicher

Department of Management Sciences,
University of Iowa,
Iowa City, IA 52242 USA

kurt-anstreicher@uiowa.edu

Abstract

We consider convex relaxations for the problem of minimizing a (possibly nonconvex) quadratic objective subject to (possibly nonconvex) quadratic constraints and simple bounds on the variables. We show that the standard approach of replacing the objective and constraint functions with their convex envelopes is dominated by an alternative methodology based on convexifying the range of the quadratic form $\binom{1}{x}\binom{1}{x}^T$ as the variables x vary within their bounds. We also show that the use of "αBB" underestimators as computable estimates of convex lower envelopes is dominated by the use of a relaxation of the convex hull of the quadratic form that imposes semidefiniteness and linear constraints on diagonal terms.

Keywords: Quadratically constrained quadratic programming, convex envelope, semidefinite programming.

1. Two convex relaxations for QCQP

We consider a quadratically constrained quadratic programming (QCQP) problem of the form

$$\begin{array}{llll} \text{(QCQP)} & z^* = & \min & f_0(x) \\ & & \text{s.t.} & f_i(x) \le b_i, \quad i = 1, \dots m \\ & & & 0 \le x \le e, \end{array}$$

where $f_i(x) = x^T Q_i x + c_i^T x$, $i = 0, 1, \dots, m$, each Q_i is an $n \times n$ symmetric matrix, and $e \in \Re^n$ is the vector with each component equal to one. In the case that $Q_i \succeq 0$ for each i, QCQP is a convex programming problem that can be solved in polynomial time, but in general the problem is NP-Hard. QCQP is a fundamental problem that has been extensively studied in the global optimization literature; see for example [**6**] and references therein.

A common approach to obtaining a lower bound for a nonconvex instance of QCQP is to somehow convexify the problem. In this paper we compare several different convexification techniques. One standard methodology is to replace each function $f_i(\cdot)$ in the problem specification with its convex lower envelope $\hat{f}_i(\cdot)$. We refer to the resulting convex relaxation of QCQP as $\widehat{\text{QCQP}}$, and use $\hat{z}$ to denote the solution value in $\widehat{\text{QCQP}}$. In the global optimization literature it is sometimes suggested that $\widehat{\text{QCQP}}$ is the "best possible" convex relaxation of QCQP, although $\hat{z}$ may not be computable because the required convex lower envelopes $\hat{f}_i(\cdot)$ may be impossible to obtain.

A completely different approach to convexifying QCQP is based on linearizing the problem by adding additional variables. Let X denote a symmetric $n \times n$ matrix. Then QCQP can be written

$$\begin{array}{llll} (\text{QCQP}) & z^* = & \min & Q_0 \bullet X + c_0^T x \\ & & \text{s.t.} & Q_i \bullet X + c_i^T x \le b_i, \quad i = 1, \ldots m \\ & & & 0 \le x \le e, \;\; X = xx^T. \end{array}$$

Written in the above form, QCQP is a linear problem except for the quadratic equality constraints $X = xx^T$. A convexification of the problem can then be given in terms of the set

$$\text{QPB}_n = \text{Co}\left\{ \begin{pmatrix} 1 \\ x \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix}^T \; : \; 0 \le x \le e \right\},$$

where $\text{Co}\{\cdot\}$ denotes the convex hull. Using QPB_n, we obtain a convex relaxation

$$\begin{array}{llll} (\widetilde{\text{QCQP}}) & \tilde{z} = & \min & Q_0 \bullet X + c_0^T x \\ & & \text{s.t.} & Q_i \bullet X + c_i^T x \le b_i, \quad i = 1, \ldots m \\ & & & Y(x, X) \in \text{QPB}_n, \end{array}$$

where

$$Y(x, X) = \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix}.$$

In this section we will demonstrate that the convex relaxation $\widehat{\text{QCQP}}$ cannot be tighter than $\widetilde{\text{QCQP}}$; in other words, it is always true that $\hat{z} \le \tilde{z}$. To do this we will show that there is a simple relationship between the convex lower envelopes used in $\widehat{\text{QCQP}}$ and the linearized representations of the objective and constraint functions used in $\widetilde{\text{QCQP}}$.

Theorem 1. For $0 \le x \le e$, let $f(x) = x^T Q x + c^T x$, and let $\hat{f}(\cdot)$ be the convex lower envelope of $f(\cdot)$. Then $\hat{f}(x) = \min\{Q \bullet X + c^T x \; : \; Y(x, X) \in \text{QPB}_n\}$.

Proof. For $0 \le x \le e$, let $g(x) = \min\{Q \bullet X + c^T x \; : \; Y(x, X) \in \text{QPB}_n\}$. Our goal is to show that $\hat{f}(x) = g(x)$. To do this we first show that $g(\cdot)$ is a convex function with $g(x) \le f(x)$, $0 \le x \le e$, implying that $g(x) \le \hat{f}(x)$.

Assume that for $i \in \{1, 2\}$, $0 \le x^i \le e$ and $g(x^i) = Q \bullet X^i + c^T x^i$, where $Y(x^i, X^i) \in \mathrm{QPB}_n$. For $0 \le \lambda \le 1$, let

$$x(\lambda) = \lambda x^1 + (1-\lambda)x^2, \quad X(\lambda) = \lambda X^1 + (1-\lambda)X^2.$$

Then $Y(x(\lambda), X(\lambda)) = \lambda Y(x^1, X^1) + (1-\lambda)Y(x^2, X^2) \in \mathrm{QPB}_n$, since QPB_n is convex. It follows that

$$g(x(\lambda)) \le Q \bullet X(\lambda) + c^T x(\lambda) = \lambda g(x^1) + (1-\lambda) g(x^2),$$

proving that $g(\cdot)$ is convex on $0 \le x \le e$. The fact that $g(x) \le f(x)$ follows immediately from $Y(x, xx^T) \in \mathrm{QPB}_n$ and $Q \bullet xx^T + c^T x = f(x)$.

It remains to show that $\hat{f}(x) \le g(x)$. Assume that $g(x) = Q \bullet X + c^T x$, where $Y(x, X) \in \mathrm{QPB}_n$. From the definition of QPB_n, there exist x^i, $0 \le x^i \le e$, and $\lambda_i \ge 0$, $i = 1, \ldots, k$, $\sum_{i=1}^k \lambda_i = 1$ such that

$$\sum_{i=1}^k x^i = x, \quad \sum_{i=1}^k x^i (x^i)^T = X.$$

It follows that

$$\begin{aligned} g(x) &= Q \bullet X + c^T x \\ &= Q \bullet \left(\sum_{i=1}^k \lambda_i x^i (x^i)^T \right) + c^T \left(\sum_{i=1}^k \lambda_i x^i \right) \\ &= \sum_{i=1}^k \lambda_i f(x^i). \end{aligned}$$

But $\hat{f}(\cdot)$ is convex, and $\hat{f}(x) \le f(x)$, $0 \le x \le e$, so

$$\hat{f}(x) = \hat{f}\left(\sum_{i=1}^k \lambda_i x^i \right) \le \sum_{i=1}^k \lambda_i \hat{f}(x^i) \le \sum_{i=1}^k \lambda_i f(x^i) = g(x).$$

□

The claimed relationship between $\widetilde{\mathrm{QCQP}}$ and $\widehat{\mathrm{QCQP}}$ is an immediate consequence of Theorem 1. In particular, using Theorem 1, $\widehat{\mathrm{QCQP}}$ could be rewritten in the form

$$\begin{aligned} (\widehat{\mathrm{QCQP}}) \quad \hat{z} = \quad & \min \quad Q_0 X_0 + c^T x \\ & \text{s.t.} \quad Q_i \bullet X_i + c_i^T x \le b_i, \quad i = 1, \ldots m \\ & \qquad\quad Y(x, X_i) \in \mathrm{QPB}_n, \ i = 0, 1, \ldots, m, \end{aligned}$$

so that $\widetilde{\mathrm{QCQP}}$ corresponds to $\widehat{\mathrm{QCQP}}$ with the added constraints $X_0 = X_1 = \ldots = X_m$.

Corollary 1. Let $\hat{z}$ and $\tilde{z}$ denote the solution values in the convex relaxations $\widehat{\mathrm{QCQP}}$ and $\widetilde{\mathrm{QCQP}}$, respectively. Then $\hat{z} \le \tilde{z}$.

Corollary 1 indicates that the approach to approximating QCQP taken in $\widetilde{\mathrm{QCQP}}$ has theoretical advantages over the more well-known methodology used in $\widehat{\mathrm{QCQP}}$. However, it is important to recognize that both of these approaches have practical limitations. In particular, both the problem of computing an exact convex lower envelope $\hat{f}(\cdot)$ for a quadratic function $f(\cdot)$, and the problem of characterizing QPB_n, are intractable. For the latter problem a number of interesting partial results are known. Important constraints that are valid for $Y(x,X) \in \text{QPB}_n$ include:

(1) The constraints from the Reformulation-Linearization Technique (RLT) [**7**],
$$\{0, x_i + x_j - 1\} \le x_{ij} \le \{x_i, x_j\}.$$
(2) The semidefinite programming (SDP) constraint $Y(x,X) \succeq 0$ [**8**].
(3) Constraints on the off-diagonal components of $Y(x,X)$ coming from the Boolean Quadric Polytope (BQP) [**5, 9**]; for example, the triangle inequalities for $i \ne j \ne k$,
$$\begin{aligned} x_i + x_j + x_k &\le x_{ij} + x_{ik} + x_{jk} + 1, \\ x_{ij} + x_{ik} &\le x_i + x_{jk}, \\ x_{ij} + x_{jk} &\le x_j + x_{ik}, \\ x_{ik} + x_{jk} &\le x_k + x_{ij}. \end{aligned}$$

The relationship between the SDP and RLT constraints is discussed in [**2**]. In fact for $n = 2$, the SDP and RLT constraints together give a full characterization of QPB_2 [**3**]. For $n = 3$ the triangle inequalities and RLT constraints fully characterize the BQP, but these constraints combined with the SDP constraint do *not* give a complete characterization of QPB_3 [**5**]. An "extended-variable" description of QPB_3 obtained via a triangulation of the 3-cube is given in [**3**].

It is known that QPB_n can be exactly represented using the cone of completely positive matrices. To describe this representation it is convenient to define
$$Y^+(x,X) = \begin{pmatrix} 1 & x^T & (e-x)^T \\ x & X & Z(x,X) \\ e-x & Z(x,X)^T & S(x,X) \end{pmatrix},$$
where $S(x,X) = ee^T - xe^T - ex^T + X$ and $Z(x,X) = xe^T - X$ relax $(e-x)(e-x)^T$ and $x(e-x)^T$, respectively. It can then be shown [4] that
$$\text{QPB}_n = \{Y(x,X) \ : \ Y^+(x,X) \in \mathcal{C}_{2n+1}\},$$
where $\mathcal{C}_k$ is the cone of $k \times k$ completely positive matrices, that is, matrices that can be written in the form VV^T where V is a nonnegative $k \times p$ matrix. Unfortunately, for $k \ge 5$ there is no known complete description for $\mathcal{C}_k$.

We close this section with an example that illustrates that the distinction between $\widetilde{\text{QCQP}}$ and $\widehat{\text{QCQP}}$ is already sharp for $n = 1$. Consider the problem
$$\begin{aligned} \min \quad & x_1^2 \\ \text{s.t.} \quad & x_1^2 \ge \tfrac{1}{2} \\ & 0 \le x_1 \le 1. \end{aligned}$$
Written in the form of QCQP, the constraint $x_1^2 \ge \frac{1}{2}$ is $-x_1^2 \le -\frac{1}{2}$, and the convex lower envelope of $-x_1^2$ on $[0,1]$ is $-x_1$. The relaxation $\widehat{\text{QCQP}}$ is then
$$\begin{aligned} \min \quad & x_1^2 \\ \text{s.t.} \quad & -x_1 \le -\tfrac{1}{2} \\ & 0 \le x_1 \le 1, \end{aligned}$$

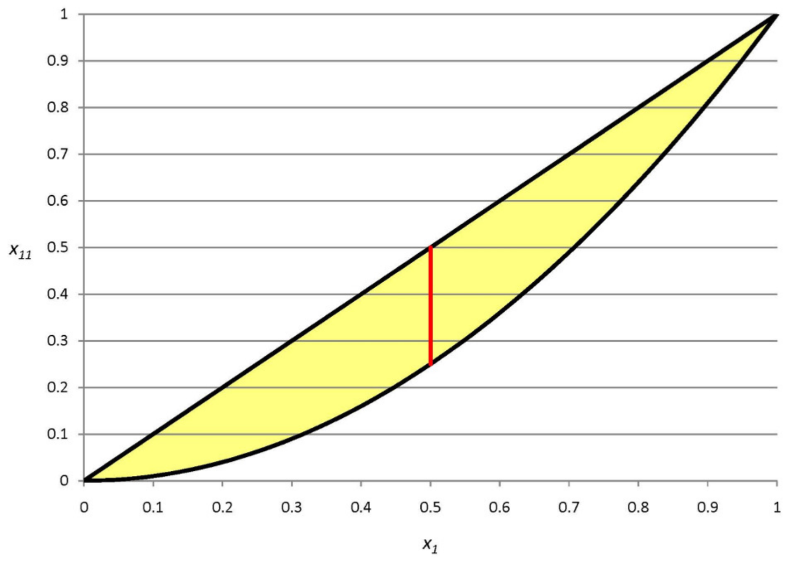


Figure 1. Set QPB_1 for example

with solution value $\hat{z} = \frac{1}{4}$. The solution value for $\widetilde{\text{QCQP}}$ is $\tilde{z} = z^* = \frac{1}{2}$. The set QPB_1 is depicted in Figure 1. Note that for $x_1 = \frac{1}{2}$, $Y(x_1, x_{11}) \in \text{QPB}_1$ for $x_{11} \in [\frac{1}{4}, \frac{1}{2}]$. The solution of $\widehat{\text{QCQP}}$ then corresponds to using $x_1 = \frac{1}{2}$ along with $x_{11} = \frac{1}{4}$ for the objective, and $x_{11} = \frac{1}{2}$ for the single nonlinear constraint.

2. Two computable relaxations

As mentioned above, in general both $\widetilde{\text{QCQP}}$ and $\widehat{\text{QCQP}}$ are intractable problems due to the complexity of computing a convex lower envelope $\hat{f}(\cdot)$, or the convex hull QPB_n. In this section we will compare two further relaxations that are computable approximations of $\widetilde{\text{QCQP}}$ and $\widehat{\text{QCQP}}$.

For a quadratic function $f(x) = x^T Qx + c^T x$ defined on $0 \le x \le e$, the well-known "αBB" underestimator [**1**] is

$$f_\alpha(x) = x^T(Q + \text{Diag}(\alpha))x + (c - \alpha)^T x,$$

where $\alpha \in \Re^n_+$ is chosen so that $Q + \text{Diag}(\alpha) \succeq 0$. Since $f_\alpha(\cdot)$ is convex, it is immediate that $f_\alpha(x) \le \hat{f}(x)$, $0 \le x \le e$. A further relaxation of $\widehat{\text{QCQP}}$ is then given by the problem

$$(\text{QCQP}_{\alpha\text{BB}}) \quad z_{\alpha\text{BB}} = \begin{array}{ll} \min & x^T(Q_0 + \text{Diag}(\alpha_0))x + (c_0 - \alpha_0)^T x \\ \text{s.t.} & x^T(Q_i + \text{Diag}(\alpha_i))x + (c_i - \alpha_i)^T x \le b_i, \quad i = 1, \dots m \\ & 0 \le x \le e, \end{array}$$

where each α_i is chosen so that $Q_i + \text{Diag}(\alpha_i) \succeq 0$.

We will compare $\text{QCQP}_{\alpha\text{BB}}$ with an approximation of $\widetilde{\text{QCQP}}$ that imposes some of the known constraints on QPB_n mentioned in the previous section. In particular, we will apply the semidefiniteness condition $Y(x, X) \succeq 0$ together with the diagonal RLT constraints $\operatorname{diag}(X) \le x$. Note that these conditions together imply the original bound constraints $0 \le x \le e$. The resulting relaxation is

$$\begin{array}{llll} (\text{QCQP}_{\text{SDP}}) & z_{\text{SDP}} = & \min & Q_0 \bullet X + c_0^T x \\ & & \text{s.t.} & Q_i \bullet X + c_i^T x \le b_i, \quad i = 1, \ldots m \\ & & & Y(x, X) \succeq 0, \quad \operatorname{diag}(X) \le x. \end{array}$$

The following theorem shows that there is a simple relationship between the convexifications used to construct $\text{QCQP}_{\alpha\text{BB}}$ and QCQP_{SDP}.

Theorem 2. For $0 \le x \le e$, let $f_\alpha(x) = x^T(Q + \operatorname{Diag}(\alpha))x + (c - \alpha)^T x$, where $\alpha \ge 0$ and $Q + \operatorname{Diag}(\alpha) \succeq 0$. Assume that $Y(x, X) \succeq 0$, $\operatorname{diag}(X) \le x$. Then $f_\alpha(x) \le Q \bullet X + c^T x$.

Proof. Let $Q_\alpha = Q + \operatorname{Diag}(\alpha)$. Since $Q_\alpha \succeq 0$,

$$\begin{aligned} f_\alpha(x) &= (c - \alpha)^T x + \min\{Q_\alpha \bullet X \ : \ X \succeq xx^T\} \\ &= \min\{Q_\alpha \bullet X + (c - \alpha)^T x \ : \ Y(x, X) \succeq 0\} \\ &= \min\{Q_\alpha \bullet X + (c - \alpha)^T x \ : \ Y(x, X) \succeq 0, \ \operatorname{diag}(X) \le x\}, \end{aligned}$$

the last because $\operatorname{diag}(X) \le x$ holds automatically for $X = xx^T$, $0 \le x \le e$. But then $Y(x, X) \succeq 0$ and $\operatorname{diag}(X) \le x$ imply that

$$\begin{aligned} f_\alpha(x) &\le Q_\alpha \bullet X + (c - \alpha)^T x \\ &= Q \bullet X + c^T x + \alpha^T(\operatorname{diag}(X) - x) \\ &\le Q \bullet X + c^T x. \end{aligned}$$

□

The following immediate corollary of Theorem 2 confirms a relationship between $\text{QCQP}_{\alpha\text{BB}}$ and QCQP_{SDP} first conjectured by Jeff Linderoth (private communication).

Corollary 2. Let $z_{\alpha\text{BB}}$ and z_{SDP} denote the solution values in the convex relaxations $\text{QCQP}_{\alpha\text{BB}}$ and QCQP_{SDP}, respectively. Then $z_{\alpha\text{BB}} \le z_{\text{SDP}}$.

Considering the example at the end of section 1, $(\alpha_1 - 1)x_1^2$ is convex for $\alpha_1 \ge 1$. Using $\alpha_1 = 1$, the problem $\text{QCQP}_{\alpha\text{BB}}$ is identical to $\widehat{\text{QCQP}}$ and has solution value $z_{\alpha\text{BB}} = \hat{z} = \frac{1}{4}$. The problem QCQP_{SDP} is identical to $\widetilde{\text{QCQP}}$, and has solution value $z_{\text{SDP}} = z^* = \frac{1}{2}$.

References

1. I.P. Androulakis, C.D. Maranas, and C.A. Floudas. αBB : A global optimization method for general constrained nonconvex problems. *J. Global Optim.*, 7:337–363, 1995.
2. K.M. Anstreicher. Semidefinite programming versus the reformulation-linearization technique for nonconvex quadratically constrained quadratic programming. *J. Global Optim.*, 43:471–484, 2009.
3. K.M. Anstreicher and S. Burer. Computable representations for convex hulls of low-dimensional quadratic forms. *Math. Prog.*, to appear, 2010.
4. S. Burer. On the copositive representation of binary and continuous nonconvex quadratic programs. *Math. Program.*, 120(2):479–495, 2009.

5. S. Burer and A.N. Letchford. On nonconvex quadratic programming with box constraints. *SIAM J. Optim.*, 20(2):1073–1089, 2009.
6. J. Linderoth. A simplicial branch-and-bound algorithm for solving quadratically constrained quadratic programs. *Math. Prog.*, 103:251–282, 2005.
7. H.D. Sherali and W.P. Adams. *A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems*. Kluwer, 1998.
8. L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38:49–95, 1996.
9. Y. Yajima and T. Fujie. A polyhedral approach for nonconvex quadratic programming problems with box constraints. *J. Global Optim.*, 13:151–170, 1998.

Valid inequalities for sets defined by multilinear functions

Pietro Belotti[1] **Andrew Miller**[2] **Mahdi Namazifar**[3]

[1] Lehigh University
200 W. Packer Ave
Bethlehem PA 18018, USA
belotti@lehigh.edu

[2] Institut de Mathématiques de Bordeaux (IMB)
351 cours de la Libération
33405 Talence, France
Andrew.Miller@math.u-bordeaux1.fr

[3] 3239 Mechanical Engineering Building
1513 University Avenue
Madison WI 53706, USA
namazifar@wisc.edu

Abstract

We describe a class of valid inequalities for the bounded, nonconvex set described by all points within a $(n+1)$-dimensional hypercube whose $(n+1)$-st coordinate is equal to the product of the first n coordinates, for any $n \geq 2$. This set can be defined as $M_n = \{x \in R^{n+1} : x_{n+1} = x_1 x_2 \cdots x_n, l_i \leq x_i \leq u_i \forall i = 1, 2 \ldots, n+1\}$, with l_i and u_i constants. Approximating the convex hull of M_n through linear inequalities is essential to a class of exact solvers for nonconvex optimization problems, namely those which use Linear Programming relaxations to compute a lower bound on the problem. Together with the well-known McCormick inequalities, these inequalities are valid for the convex hull of M_2. There are infinitely many such inequalities, given that the convex hull of M_2 is not, in general, a polyhedron. The generalization to M_n for $n > 2$ is straightforward, and allows us to define strengthened relaxations for these higher dimensional sets as well.

Keywords: nonconvex optimization, multilinear functions, convex hull.

1. Multilinear sets

Consider the following nonconvex, bounded set

$$M_n = \{x \in \mathbb{R}^{n+1} : x_{n+1} = \prod_{i=1}^{n} x_i, \ell_i \leq x_i \leq u_i, i = 1, 2 \ldots, n+1\},$$

where l and u are $(n+1)$-vectors. We assume $0 < \ell_i < u_i$ for all $i = 1, 2 \ldots, n+1$. The set M_n is nonconvex as the function $\xi_n(x) = \prod_{i=1}^{n} x_i$ is neither convex nor concave – its *epigraph* $\operatorname{epi}(M) = \{x \in \mathbb{R}^{n+1} : x_{n+1} \geq \prod_{i=1}^{n} x_i, 0 < \ell_i \leq x_i \leq u_i, i = 1, 2 \ldots, n\}$ and its *hypograph* $\operatorname{hyp}(M) = \{x \in \mathbb{R}^{n+1} : x_{n+1} \leq \prod_{i=1}^{n} x_i, 0 < \ell_i \leq x_i \leq u_i, i = 1, 2 \ldots, n\}$ are nonconvex sets.

Assuming M_n is contained in the first orthant, trivial bounds for x_{n+1} are given by $\prod_{i=1}^{n} \ell_i$ and $\prod_{i=1}^{n} u_i$, respectively, and denoted by $\bar{\ell}_{n+1}$ and $\bar{u}_{n+1}$. In general, $\bar{\ell}_{n+1} \leq \ell_{n+1} < u_{n+1} \leq \bar{u}_{n+1}$; in the remainder, we use the notation $M_n^\star$ for the set M_n where $\ell_{n+1} = \bar{\ell}_{n+1}$ and $u_{n+1} = \bar{u}_{n+1}$.

We are interested in developing a convex superset C_n of M_n defined by a system of linear inequalities, therefore we seek a polyhedral set $C_n \supseteq M_n$. Our interest is motivated by the nonconvex Optimization problem

$$\mathbf{P} : \min\{cx : x \in X\},$$

where X is, in general, a nonconvex set. In order to solve problems like $\mathbf{P}$ to optimality, one needs to implicitly enumerate all local optima, for instance by branch-and-bound algorithms. The bounding algorithms in such approaches often relies on a linear relaxation of the nonconvex problem [**3, 4**], thus benefiting from tighter linear relaxations.

2. Linear inequalities for $n = 2$

Here we develop convex inequalities for the set $M_2 = \{(x_1, x_2, x_3) \in [l, u] : x_3 = x_1 x_2\}$. Although this is the simplest case, many considerations generalize easily to the case when $n > 2$.

2.1. Unbounded x_3

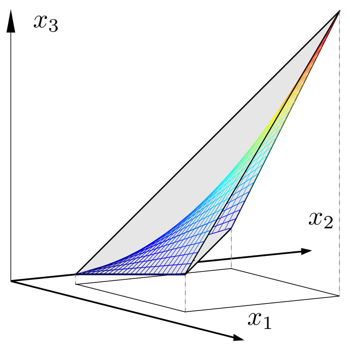
The following linear relaxation of $M_2^\star$ was introduced by McCormick [**2**] and shown to be its convex hull by Al-Khayyal and Falk [**1**]:

$$\begin{aligned} x_3 &\geq \ell_2 x_1 + \ell_1 x_2 - \ell_1 \ell_2 \\ x_3 &\geq u_2 x_1 + u_1 x_2 - u_1 u_2 \\ x_3 &\leq \ell_2 x_1 + u_1 x_2 - \ell_1 u_2 \\ x_3 &\leq u_2 x_1 + \ell_1 x_2 - u_1 \ell_2, \end{aligned}$$

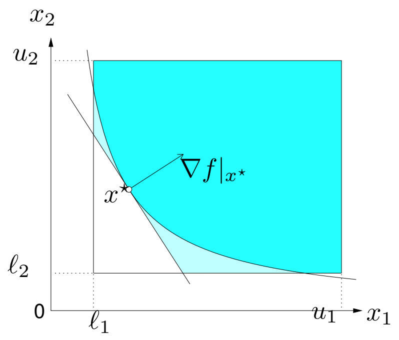
and is depicted in Figure 1a for convenience. The shaded tetrahedron is the (polyhedral) convex hull, while $M_2^\star$ is shown in colors.

2.2. Nontrivial lower and upper bound for x_3

We consider first a finite lower bound, $\ell_3 > \bar{\ell}_3 = \ell_1 \ell_2$. The darker area in Figure 1b shows the projection of M_2 onto (x_1, x_2), that is, the set $P_2 = \{(x_1, x_2) \in \mathbb{R}^2 : \ell_i \leq x_i \leq u_i, i = 1, 2, x_1 x_2 \geq \ell_3\}$. It is safe to assume here that $\ell_3 \leq \ell_1 u_2$ and $\ell_3 \leq u_1 \ell_2$, as otherwise a valid lower bound for x_1 (resp. x_2) would be $\ell_3 / u_2 > \ell_1$ (resp $\ell_3 / u_1 > \ell_2$), or equivalently, the upper left (resp. the lower right) corner of the



(a) Convex hull of $M_2^\star$.



(b) Projections of M_2 with non-trivial lower bound ℓ_3.

Figure 1

bounding box in Figure 1b would be cut out by the convex set $x_1x_2 \geq \ell_3$. Similarly we assume that $u_3 \geq \ell_1 u_2$ and $u_3 \geq u_1\ell_2$.

Projecting M_2 onto (x_1, x_2) gives a convex set P_2. Consider a point $x^\star$ on the curve $x_1x_2 = \ell_3$ with $\ell_1 \leq x_1^\star \leq u_1$ and $\ell_2 \leq x_2^\star = \ell_3/x_1^\star \leq u_2$. The tangent to the curve $x_1x_2 = \ell_3$ at $x^\star$ gives a linear inequality $a_1(x_1 - x_1^\star) + a_2(x_2 - x_2^\star) \geq 0$, that is valid for P_2. The coefficients a_1 and a_2 are given by the gradient of the function $\xi_2(x) = x_1x_2$ at $x^\star$, i.e., $a_1 = \frac{\partial\pi}{\partial x_1}|_{x^\star} = x_2^\star$ and $a_2 = \frac{\partial\pi}{\partial x_2}|_{x^\star} = x_1^\star$. The lighter area within the bounding box above the tangent line is the set of points satisfying the above linear constraint, which we rewrite here:

$$x_2^\star(x_1 - x_1^\star) + x_1^\star(x_2 - x_2^\star) \geq 0. \tag{3.1}$$

As this inequality is valid within P_2 and is independent from x_3, it is also valid for M_2. We can lift it as follows: the inequality

$$x_2^\star(x_1 - x_1^\star) + x_1^\star(x_2 - x_2^\star) + a(x_3 - \ell_3) \geq 0 \tag{3.2}$$

is clearly valid for $x_3 = \ell_3$. For it to be valid for M_2,

$$g(a_3) = \min\{x_2^\star(x_1 - x_1^\star) + x_1^\star(x_2 - x_2^\star) + a(x_3 - \ell_3) : (x_1, x_2, x_3) \in M_2\} \geq 0$$

must hold. It is easy to show that $g(a_3) = 0$ if $a \geq 0$ (a global optimum is given by $(x_1^\star, x_2^\star)$), hence $a < 0$ in all lifted inequalities (3.2).

We next show how to calculate a. First we will show how to get this coefficient and then we will prove that the inequalities we get are not dominated by any other valid inequality. To find the coefficient a in the inequality (3.2), intuitively we know that the plane

$$x_2^\star(x_1 - x_1^\star) + x_1^\star(x_2 - x_2^\star) + a(x_3 - \ell_3) = 0 \tag{3.3}$$

should touch the curve $x_1x_2 = u_3$ at exactly one point and first we want to find this point. Let's call this point $(\bar{x}_1, \bar{x}_2)$. If for a moment we disregard the bounds on x_1 and x_2, the fact that the plane (3.3) touches the curve $(\bar{x}_1, \bar{x}_2)$ at exactly one point means that the plane would be tangent to the curve $x_1x_2 = u_3$ at that point. This means that the gradient of the curve at $(\bar{x}_1, \bar{x}_2)$ is parallel to the projection of the normal to the plane onto the (x_1, x_2) space. The gradient of the curve $x_1x_2 = u_3$ at

$(\bar{x}_1, \bar{x}_2)$ is $(\bar{x}_2, \bar{x}_1)$ and the projection of the normal to the plane onto the (x_1, x_2) space is $(x_2^\star, x_1^\star)$. As a result we will have:

(3.4) $$\exists \alpha \in \mathbb{R} : \bar{x}_2 = \alpha x_2^\star, \bar{x}_1 = \alpha x_1^\star.$$

But we know that $\bar{x}_1\bar{x}_2 = u_3$, so we will have $\bar{x}_1\bar{x}_2 = u_3 = \alpha^2 x_1^\star x_2^\star = \alpha^2 \ell_3$, and as a result

(3.5) $$\alpha = \sqrt{\frac{u_3}{\ell_3}}.$$

On the other hand we know that $(\bar{x}_1, \bar{x}_2)$ is on the plane (3.3), which implies

$$x_2^\star(\bar{x}_1 - x_1^\star) + x_1^\star(\bar{x}_2 - x_2^\star) + a(u_3 - \ell_3) = 0,$$

As a result,

(3.6) $$x_2^\star(\alpha x_1^\star - x_1^\star) + x_1^\star(\alpha x_2^\star - x_2^\star) + a(u_3 - \ell_3) = 0.$$

By (3.5) and (3.6), the value for a is $\frac{2(1-\sqrt{\frac{u_3}{\ell_3}})\ell_3}{u_3-\ell_3}$.

Notice that the value of a does not depend on the value of $x_1^\star$ and $x_2^\star$ and is less than zero. Intuitively this value for a will give a tight valid inequality for M_2 if $\ell_1 \leq \bar{x}_1 \leq u_1$ and $\ell_2 \leq \bar{x}_2 \leq u_2$ (the proof will come later). But if $(\bar{x}_1, \bar{x}_2)$ is not in the domain of (x_1, x_2) we need to find a point on the curve $x_1x_2 = u_3$ at which the plane (3.3) touches the curve. Intuitively this point would be one of the end points of the curve $x_1x_2 = u_3; \ell_1 \leq x_1 \leq u_1, \ell_2 \leq x_2 \leq u_2$. In fact that point will be the one which is closer to $(\bar{x}_1, \bar{x}_2)$.

Again consider the curve $x_1x_2 = u_3$. On this curve we know that:

$$x_1 \geq \frac{u_3}{u_2} = \bar{l}_1; \quad x_2 \geq \frac{u_3}{u_1} = \bar{l}_2; \quad x_1 \leq \frac{u_3}{\ell_2} = \bar{u}_1; \quad x_2 \leq \frac{u_3}{\ell_1} = \bar{u}_2.$$

Then the real bounds over x_1 and x_2 on the curve $x_1x_2 = u_3$ would be $\ell_1^\star \leq x_1 \leq u_1^\star$, $\ell_1^\star = \max(\bar{l}_1, \ell_1)$, and $u_1^\star = \min(\bar{u}_1, u_1)$; $\ell_2^\star \leq x_2 \leq u_2^\star$, $\ell_2^\star = \max(\bar{l}_2, \ell_2)$, and $u_2^\star = \min(\bar{u}_2, u_2)$. However, based on the assumptions on the upper bound of x_3 (see Section 2.2)

(3.7) $$\ell_1^\star = \frac{u_3}{u_2}; \quad u_1^\star = u_1; \quad \ell_2^\star = \frac{u_3}{u_1}; \quad u_2^\star = u_2.$$

As a result, the end points of the curve $x_1x_2 = u_3$ in M_2 are $(\frac{u_3}{u_2}, u_2)$ and $(u_1, \frac{u_3}{u_1})$. It's easy to see that if $\alpha x_1^\star \geq u_1$ and $\alpha x_2^\star \leq \frac{u_3}{u_1}$, the the right end point would be $(u_1, \frac{u_3}{u_1})$, and if $\alpha x_1 \leq \frac{u_3}{u_2}$ and $\alpha x_2 \geq u_2$, the right end point would be $(\frac{u_3}{u_2}, u_2)$. So if $\alpha x_1^\star \geq u_1$ and $\alpha x_2^\star \leq \frac{u_3}{u_1}$ and $\frac{u_1}{\alpha} \leq x_1^\star \leq \frac{\ell_3}{\ell_2}$, the plane (3.3) would touch the curve $x_1x_2 = u_3$ at $(u_1, \frac{u_3}{u_1})$ and therefore we will have:

(3.8) $$x_2^\star(u_1 - x_1^\star) + x_1^\star(\frac{u_3}{u_1} - x_2^\star) + a(u_3 - \ell_3) = 0,$$

and as a result

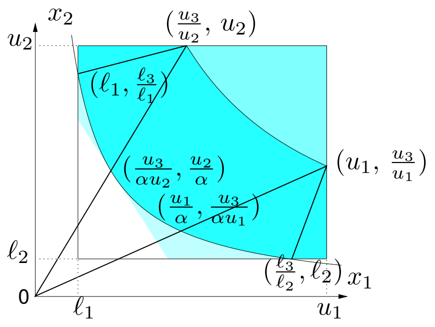
(3.9) $$a = \frac{x_2^\star(u_1 - x_1^\star) + x_1^\star(\frac{u_3}{u_1} - x_2^\star)}{\ell_3 - u_3}.$$

On the other hand, if $\alpha x_1 \leq \frac{u_3}{u_1}$ and $\alpha x_2 \geq u_2$ and $\ell_1 \leq x_1 \leq \frac{u_3}{\alpha u_2}$, the plane (3.3) would touch the curve $x_1x_2 = u_3$ at $(\frac{u_3}{u_2}, u_2)$ and we will have:

(3.10) $$x_2^\star(\frac{u_3}{u_2} - x_1^\star) + x_1^\star(u_2 - x_2^\star) + a(u_3 - \ell_3) = 0,$$

hence,

$$a = \frac{x_2^\star(\frac{u_3}{u_2} - x_1^\star) + x_1^\star(u_2 - x_2^\star)}{\ell_3 - u_3}. \tag{3.11}$$



x_2
$(\frac{u_3}{u_2},\, u_2)$
u_2
$(\ell_1,\, \frac{\ell_3}{\ell_1})$
$(\frac{u_3}{\alpha u_2},\, \frac{u_2}{\alpha})$
$(u_1,\, \frac{u_3}{u_1})$
$(\frac{u_1}{\alpha},\, \frac{u_3}{\alpha u_1})$
ℓ_2
$(\frac{\ell_3}{\ell_2},\, \ell_2)$
x_1
0
ℓ_1
u_1

Figure 2

Now we need to deal with two other cases; $\frac{\ell_3}{\ell_2} \le x_1^\star \le u_1$, $x_2^\star = \ell_2$ and $x_1^\star = \ell_1$, $\frac{\ell_3}{\ell_1} \le x_2^\star \le u_2$. First consider the case which $\frac{\ell_3}{\ell_2} \le x_1^\star \le u_1$, $x_2^\star = \ell_2$. Intuitively we can see that the plane which goes through the points $(\frac{\ell_3}{\ell_2}, \ell_2, \ell_3)$, $(u_1, \ell_2, u_1\ell_2)$, and $(u_1, \frac{u_3}{u_1}, u_3)$ gives an inequality which is valid for M_2 and is not dominated by any other inequality. Similarly for the second case in which $x_1^\star = \ell_1$, $\frac{\ell_3}{\ell_1} \le x_2^\star \le u_2$, the plane which goes through the points $(\ell_1, \frac{\ell_3}{\ell_1}, \ell_3)$, $(\ell_1, u_2, \ell_1 u_2)$, and $(\frac{u_3}{u_2}, u_2, u_3)$ gives an inequality which is valid for M_2 and is not dominated by any other inequality.

In summary, as can be seen in Figure 2, if $\frac{u_3}{\alpha u_2} < x_1^\star < \frac{u_1}{\alpha}$ and $\frac{u_3}{\alpha u_1} < x_2^\star < \frac{u_2}{\alpha}$, then the point $(\bar{x}_1, \bar{x}_2)$ (the point at which the plane (3.3) touches the curve $x_1 x_2 = u_3$) will have $\frac{u_3}{u_2} < \bar{x}_1 < u_1$ and $\frac{u_3}{u_1} < \bar{x}_2 < u_2$. Otherwise, $(\bar{x}_1, \bar{x}_2)$ happens at either $(\frac{u_3}{u_2}, u_2)$ or $(u_1, \frac{u_3}{u_1})$.

References

1. F.A. Al-Khayyal and J.E. Falk. Jointly constrained biconvex programming. *Mathematics of Operations Research*, 8:273–286, 1983.
2. G.P. McCormick. *Nonlinear programming: theory, algorithms and applications.* John Wiley & sons, 1983.
3. E.M.B. Smith and C.C. Pantelides. Global optimisation of nonconvex MINLPs. *Computers & Chemical Engineering*, 21:S791–S796, 1997.
4. M. Tawarmalani and N.V. Sahinidis. Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. *Mathematical Programming*, 99(3):563–591, 2004.

Using Semidefinite Programming for Solving Non-Convex Mixed-Integer Quadratic Problems

Christoph Buchheim[1] **Angelika Wiegele**[2]

[1] Technische Universität Dortmund
Vogelpothsweg 87
44227 Dortmund, Germany
`christoph.buchheim@udo.edu`

[2] Alpen-Adria-Universität Klagenfurt
Universitätsstr. 65-67
9020 Klagenfurt am Wörthersee, Austria
`angelika.wiegele@uni-klu.ac.at`

Abstract

We present a semidefinite relaxation for non-convex quadratic mixed-integer optimization problems. This relaxation yields tight bounds and is computationally easy to solve for medium-sized instances, even if variables are unbounded in which case the problem contains an infinite number of constraints. These constraints are separated dynamically. We use this approach as the bounding routine in an SDP based branch-and-bound framework. In case of a convex objective function, the bound of the SDP improves the bound of the continuous relaxation. Numerical experiments show that our algorithm performs very well on various types of random instances.

Keywords: Semidefinite Programming, Mixed-Integer Quadratic Programming.

1. Introduction

Semidefinite Programming has been successfully applied for solving various 0/1 optimization problems, most of them arising from applications in combinatorial optimization [**3**]. Furthermore it has been used to solve quadratic problems with box-constrained continuous variables, so-called *BoxQP*, see [**5, 2**].

We widen the area of using semidefinite relaxations for solving quadratic problems. Contrary to existing work, we can deal with any integrality condition and not only 0/1. Furthermore we are able to handle integer and continuous variables at the same time, i.e., our approach is capable of solving mixed-integer problems.

2. The Method

Let $Q \in S_n$ denote a (not necessarily positive semidefinite) matrix. Moreover, let $l \in \mathbb{R}^n$ and $c \in \mathbb{R}$. We consider the following problem.

$$\text{(4.1)} \qquad \begin{array}{rl} \min & x^\top Q x + l^\top x + c \\ \text{s.t.} & x \in D_1 \times \cdots \times D_n \ , \end{array}$$

where each D_i is a closed set in $\mathbb{R}$. For example, if $D_i = \{-1, 1\}$, we obtain the maximum cut problem, or if $D_i = \mathbb{Z}$ with a convex objective function we obtain the closest vector problem. However, Problem (4.1) is more general, e.g., one can choose $D_i = \{0\} \cup [1, \infty)$. Note that it is allowed to choose all D_i differently, so that Problem (4.1) also contains the mixed-integer case.

Using the linearization function $\ell \colon \mathbb{R}^n \to S_{n+1}$ defined as

$$\ell(x) = \begin{pmatrix} 1 \\ x \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix}^\top ,$$

and the definition of

$$\tilde{Q} = \begin{pmatrix} c & \frac{1}{2} l^\top \\ \frac{1}{2} l & Q \end{pmatrix} ,$$

which is indexed over $\{0, \ldots, n\}$, we can rewrite Problem (4.1) as

$$\text{(4.2)} \qquad \begin{array}{rl} \min & \langle \tilde{Q}, \ell(x) \rangle \\ \text{s.t.} & x \in D_1 \times \cdots \times D_n \ . \end{array}$$

Since the objective function in Problem (4.2) is linear in $\ell(x)$, we investigate the convex set conv $\ell(D_1 \times \cdots \times D_n)$. To this end, we define $P(D_i) \subseteq \mathbb{R}^2$ as the closure of conv $\{(x, x^2) \mid x \in D_i\}$.

Using this definition and in order to use semidefinite programming to solve Problem (4.1), we state the following

Theorem 4.1. *Problem (4.1) is equivalent to*

$$\text{(4.3)} \qquad \begin{array}{rrcll} \min & \langle \tilde{Q}, X \rangle & & & \\ \text{s.t.} & (x_{0i}, x_{ii}) & \in & P(D_i) & \forall i = 1, \ldots, n \\ & x_{00} & = & 1 & \\ & \operatorname{rank}(X) & = & 1 & \\ & X & \succeq & 0 \ . & \end{array}$$

Proof. Let X be a feasible solution of Problem (4.3). From $x_{00} = 1$, $\operatorname{rank}(X) = 1$ and $X \succeq 0$, we derive

$$X = \begin{pmatrix} 1 \\ x \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix}^\top ,$$

where $x \in \mathbb{R}^n$. Hence, $x_{ii} = x_{0i}^2$. With $(x_{0i}, x_{ii}) \in P(D_i)$, this implies $x_{0i} \in D_i$, using the strict convexity of the function $x_{0i} \mapsto x_{0i}^2$. Thus $(x_{0i}, \ldots, x_{0n})$ is a feasible solution for Problem (4.1). On the other hand, if x is a feasible solution for Problem (4.1), then

$$X = \begin{pmatrix} 1 \\ x \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix}^\top ,$$

defines a feasible solution of Problem (4.3). Obviously, the objective function values coincide. □

Note that the only non-convex constraint in Problem (4.3) is the rank-constraint. Dropping this constraint leads to the SDP relaxation

$$\begin{array}{rrcll} \min & \langle \tilde{Q}, X\rangle & & & \\ \text{s.t.} & (x_{0i}, x_{ii}) & \in & P(D_i) & \forall i = 1,\dots,n \\ & x_{00} & = & 1 & \\ & X & \succeq & 0\,. & \end{array} \tag{4.4}$$

If the pair (x_{0i}, x_{ii}) arises from a positive semidefinite matrix, $x_{ii} \geq x_{0i}^2$ always holds. Hence, $(x_{0i}, x_{ii}) \in P(\mathbb{R})$ already follows from the positive semidefiniteness of X.

In general, however, we need an infinite number of inequalities to model the constraints $(x_{0i}, x_{ii}) \in P(D_i)$ in Problem (4.4). By separating these inequalities we can obtain the solution to this SDP efficiently. Such a separation algorithm is possible, whenever we can solve the problems

- $\min D_i \cap [x_{0i}, \infty)$ and
- $\max D_i \cap (-\infty, x_{0i}]$

for given x_{0i}. Thus, we obtain a solution of relaxation (4.4) by iteratively adding inequalities and solving the resulting SDP by an interior point method.
Branch and Bound. In order to solve Problem (4.1) to optimality, we use the SDP relaxation (4.4) in a branch-and-bound framework. Our branching decision is motivated by the following

Observation 1. Let $X \in S_{n+1}^{+}$ with $x_{00} = 1$. Then
$x_{ii} = x_{0i}^2$ for all $i = 1,\dots,n \iff X = \ell(x)$ for $x = (x_{01},\dots,x_{0n})^\top$.

This observation suggests to branch on variable x_i with maximal $x_{ii} - x_{0i}^2$. We obtain two subproblems by splitting the corresponding D_i into

- $D_i \cap [x_{0i}, \infty)$ and
- $D_i \cap (-\infty, x_{0i}]$,

in which way the new problems fit into our problem description. As soon as $x_{ii} = x_{0i}^2$ for all i, we have found an optimal solution of Problem (4.1).

3. Numerical Experiments

We implemented the algorithm described above in C++, using CSDP [**1**] for solving the SDP relaxation. In order to illustrate the performance of our method, we carry out some numerical experiments. To this end, we implemented a random generator, which allows us to control the number of negative Eigenvalues of the cost matrix. In this way we can control whether our objective function is convex, concave or neither of them.

We consider problems of the form (4.1), where half of the D_i are equal to $\{0, 1\}$ and half of the D_i are equal to $[0, 1]$, i.e., half of the variables are binary, half of them are boxed-constrained continuous variables.

We generated data with dimensions n ranging from 10 up to 50. For each dimension we generated 10 instances having a percentage of p negative Eigenvalues, where $p = 0, 10, 20, \dots, 100$. In this way we make sure that the set of all the matrices Q we generated contains positive definite, negative definite and indefinite matrices.

Table 1. Instances with 50% binary variables and 50% box-constrained continuous variables

	avg.	time (sec)		
n	b&b-nodes	min	max	avg
10	26.4	0	1	0.1
20	97.8	0	13	0.8
30	736.7	0	176	12.8
40	5217.8	0	9714	160.0
50	11339.3	0	12934	356.4

In Figure 1 we plot the results for $n = 10, \ldots, 50$ of the average time (in seconds) over 10 instances. On the x-axes the percentage of negative Eigenvalues of matrix Q is given. As expected, instances having a convex or concave objective function are easy compared to instances where matrix Q is indefinite.

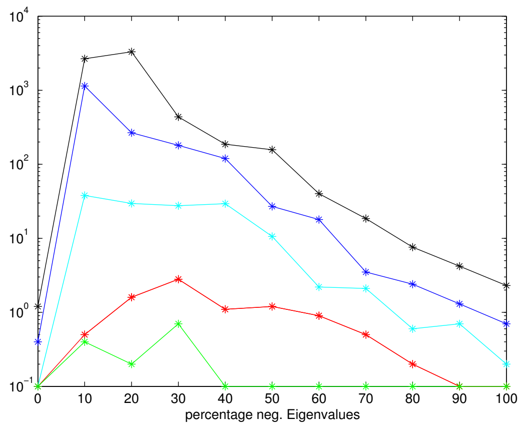


Figure 1. "Convexity versus concavity".

Summarized results of this experiment are given in Table 1. Up to dimension 30, problems are solved within seconds. For $n = 40$ one has to allow several minutes (up to 2.5 hours) to solve instances, and for dimension 50 it takes on average 6 minutes to solve a problem, but for some instances one has to wait up to 4 hours to have the solution at hand.

We carried out also experiments on ternary problems (see [**4**]) and BoxQP in order to compare to existing algorithms. These experiments demonstrate that our general algorithm achieves results in comparable or slightly longer time than in [**4**] or [**5**].

4. Conclusion

We presented an exact algorithm for solving various types of quadratic problems. The algorithm uses semidefinite relaxations in a branch-and-bound setting. To the best of our knowledge this is the first time that SDP is used for solving mixed-integer programs.

Experiments show that we efficiently solve mixed-integer quadratic problems. Various comparisons to algorithms specialized for a certain class of problems indicate that our algorithm performs reasonably well.

Dealing with explicit constraints and thus making this new concept applicable to an even wider range of problems seems to be promising and is subject of future research.

References

1. Brian Borchers. CSDP, a C library for semidefinite programming. *Optim. Methods Softw.*, 11/12(1-4):613–623, 1999. Interior point methods.
2. Samuel Burer and Dieter Vandenbussche. A finite branch-and-bound algorithm for nonconvex quadratic programming via semidefinite relaxations. *Math. Program.*, 113(2, Ser. A):259–282, 2008.
3. Monique Laurent and Franz Rendl. Semidefinite programming and integer programming. Aardal, K. (ed.) et al., Discrete optimization. Amsterdam: Elsevier. Handbooks in Operations Research and Management Science 12, 393-514 (2005)., 2005.
4. Nguyen Thi Hoai Phuong, Hoang Tuy, and Faiz Al-Khayyal. Optimization of a quadratic function with a circulant matrix. *Comput. Optim. Appl.*, 35(2):135–159, 2006.
5. Dieter Vandenbussche and George L. Nemhauser. A branch-and-cut algorithm for nonconvex quadratic programs with box constraints. *Math. Program.*, 102(3, Ser. A):559–575, 2005.

Feasibility Pump(s) for Non-Convex Mixed-Integer NonLinear Programs

Claudia D'Ambrosio[1] **Antonio Frangioni**[2] **Leo Liberti**[3] **Andrea Lodi**[1]

[1] DEIS, University of Bologna
Viale Risorgimento 2
40136 Bologna, Italy
`{c.dambrosio,andrea.lodi}@unibo.it`

[2] DI, University of Pisa
Largo B. Pontecorvo, 3
56127 Pisa, Italy
`frangio@di.unipi.it`

[3] LIX, École Polytechnique
91128 Palaiseau, France
`liberti@lix.polytechnique.fr`

Keywords: heuristic, Mixed-Integer NonLinear Programming, feasibility pump.

We present a new Feasibility Pump (FP) algorithm tailored for nonconvex Mixed Integer Nonlinear Programming (MINLP) problems.

In general, Feasibility Pump algorithms are based on iterative solution of two subproblems which represent a relaxation of the original problem to be solved. The objective functions of the two subproblems are selected so as the aim is to make the two trajectories of the solutions of the two subproblems converge to a unique point, which is feasible for both the subproblems and, consequently, for the original problem. Thus, the focus is on finding feasible solutions to a very difficult problem, in general, the first one.

In the nonconvex MINLP case, difficulties arising from nonconvexities in the models has to be overcome. We extensively discuss such difficulties. In this case the first subproblem, obtained relaxing the integer requirements on the variables, is a nonconvex Nonlinear Program (NLP), which uses as objective function the 2-norm distance function. For the second subproblem, we explore different approaches:

(i) a Mixed Integer Linear Program (MILP), obtained by relaxing and/or linearizing the nonlinear constraints and using a linear distance function as objective function, such as the 1-norm;

(ii) a Mixed Integer Quadratic Program, obtained relaxing and/or linearizing the nonlinear constraints but using a nonlinear distance function as objective function, the 2-norm;

(iii) not defining an second subproblem, but, following the original idea of FP for MILPs [**1, 2**], just round the values of the integer variables obtained solving the NLP subproblem.

We present theoretical justifications for the different approaches and practical comparisons. We exhibit computational results showing the good performance of the algorithms on instances taken from the MINLPLib.

References

1. L. Bertacco, M. Fischetti, and A. Lodi. A feasibility pump heuristic for general mixed-integer problems. *Discrete Optimization*, 4:63–76, 2007.
2. M. Fischetti, F. Glover, and A. Lodi. The feasibility pump. *Mathematical Programming*, 104:91–104, 2005.

6

Experiments with the Perspective Reformulation

Hyemin Jeon **Jeffrey T. Linderoth**

Department of Industrial and Systems Engineering
University of Wisconsin-Madison
1513 University Ave.
Madison, WI 53706-1572, USA

jeon5,linderoth@wisc.edu

Abstract

We describe some integer-programming based approaches for finding strong inequalities for the convex hull of a quadratic mixed integer nonlinear set. The techniques are closely related to recent perspective reformulations of MINLPs.
Keywords: Mixed Integer Nonlinear Programming, perspective function.

1. Introduction

Our work focuses on the convex hull of the nonlinear set

$$X = \{(x, z, v) \in \mathbb{R}^n_+ \times \mathbb{B}^n \times \mathbb{R} \mid v \geqslant x^T Q x, x_j \leqslant z_j \forall j \in N\}, \tag{6.1}$$

where $N = \{1, 2, \ldots, n\}$, and the matrix $Q \succeq 0$. The set X appears as a substructure in many practical mixed integer nonlinear programs (MINLPs), including those arising from portfolio management [**2**] or model selection [**4**].

In the case that $Q = \mathrm{diag}(q)$, Stubbs [**9**] characterized $\mathrm{conv}(X)$ as

(6.2)
$$\mathrm{conv}(X) = \mathrm{proj}_{x,z,v}\{(x, z, v, t) \in \mathbb{R}^n_+ \times [0,1]^n \times \mathbb{R} \times \mathbb{R}^n_+ \mid v \geqslant \sum_{j \in N} q_j t_j, t_j z_j \geqslant q_j x_j^2 \; \forall j \in N, \\ x_j \leqslant z_j \; \forall j \in N\}.$$

The formula (6.2) can be generalized to the set

$$Y = \{(x, z, v) \in \mathbb{R}^n_+ \times \mathbb{B}^n \times \mathbb{R} \mid v \geqslant \sum_{j=1}^{n} f_j(x_j), x_j \leqslant z_j \; \forall j \in N\},$$

for some convex, separable function $f(\cdot)$. In this case,

(6.3)
$$\operatorname{clconv}(Y) = \{(x, z, v) \in \mathbb{R}^n_+ \times [0,1]^n \times \mathbb{R} \mid v \geqslant \sum_{j=1}^{n} z_j f_j(x_j/z_j), x_j \leqslant z_j \ \forall j \in N\},$$

which is a transformation known as the *perspective reformulation* of Y. The formula (6.3) is given explicitly in [**8**], and is a specialization of a more general theorem on the convex hull of a disjunctive set found in [**3**].

The characterization of the convex hull (6.3) is also implicitly given by Frangioni and Gentile in their derivation of *perspective cuts* [**5**]. The perspective cut

$$v \geqslant (f(\hat{x}) + \nabla f(\hat{x})^T \hat{x})z + \nabla f(\hat{x})^T x \tag{6.4}$$

is a valid inequality for $\operatorname{conv}(Y)$ for all $\hat{x} \in \operatorname{dom}(f)$. Moreover, (6.4) is a maximal face of $\operatorname{conv}(Y)$ of dimension at least one.

Frangioni and Gentile demonstrate in [**7**] that in many cases, using the linear inequalities (6.4) with a cutting-plane approach is computationally superior to using the nonlinear formulation (6.3) of $\operatorname{conv}(Y)$, or even a second-order cone programming formulation as proposed by [**1, 8**].

An Integer Programming Perspective

As motivation, consider the simple 3-dimensional set

$$X_0 = \{(x, z, v) \in \mathbb{R}_+ \times \{0,1\} \times \mathbb{R} \mid v \geqslant x^2, x \leqslant z\},$$

for some convex function $f : \mathbb{R} \to \mathbb{R}$. An interesting observation is that the (linear) perspective cuts (6.4) are equivalent to building a linear outerapproximation of the set X_0 and then strengthening the linear inequalities using a logical deductive argument. To outerapproximate X_0, a set of points $B = \{\chi^1, \chi^2, \dots \chi^{|B|}\}$ are chosen, and the convexity of f implies that

$$\mathcal{O}(X_0) = \{(x, z, v) \in \mathbb{R}_+ \times \{0,1\} \times \mathbb{R} \mid v \geqslant 2(\chi^b)x_j - (\chi^b)^2 \ \forall b \in B, x \leqslant z\}$$

is a relaxation of X_0. Since $z = 0 \implies x_0$, the inequalities

$$v \geqslant 2(\chi^b)x_j - (\chi^b)^2 \ \forall b \in B$$

in the description of $\mathcal{O}(X_0)$ can each be strengthened to

$$v \geqslant 2(\chi^b)x_j - (\chi^b)^2 z_j \ \forall b \in B. \tag{6.5}$$

The simple inequalities (6.5) are precisely the perspective cuts (6.4) of Frangioni and Gentile for this instance. The fact that these inequalities can be so effective for solving convex MINLPs gives motivation to consider strengthening polyhedral relaxations of different nonlinear sets via integer programming arguments in order to derive inequalities that are useful for solving convex MINLPs.

2. Handling Nonseparability

The perspective reformulation or perspective cuts have been shown to be a very effective mechanism for solving certain classes of convex MINLPs. Unfortunately, direct application of the perspective reformulation to (6.1) is not possible, as the objective is not a separable function of the decision variables x.

Frangioni and Gentile [**5**] suggest to extract a separable component from Q and to apply the perspective reformulation to this separate component. Specifically, the matrix Q may be decomposed into $Q = R + D$, for some matrix $D = \text{diag}(d) \geqslant 0$ such that $R = Q - D \succeq 0$. With this transformation, the set (6.1) can be written as

$$X = \text{proj}_{x,z,v}\{(x, z, v, t) \in \mathbb{R}^n_+ \times \mathbb{B}^n \times \mathbb{R} \times \mathbb{R}^n_+ \mid v \geqslant x^T Rx + e^T t, t_j \geqslant d_j x_j^2,\ \forall j \in N, x_j \leqslant z_j \forall j \in N\}. \tag{6.6}$$

The set (6.6) can be strengthened via the perspective reformulation to be

$$\mathcal{P}(X) = \text{proj}_{x,z,v}\{(x, z, v, t) \in \mathbb{R}^n_+ \times \mathbb{B}^n \times \mathbb{R} \times \mathbb{R}^n_+ \mid v \geqslant x^T Rx + e^T t, z_j t_j \geqslant d_j x_j^2,\ \forall j \in N, x_j \leqslant z_j \forall j \in N\}.$$

In [**5**], Frangioni and Gentile suggest using $D = \lambda_n I$, where $\lambda_n > 0$ is the smallest eigenvalue of Q. In subsequent work, they show how "more" of the separable structure of Q can be extracted into D through the solution of a semidefinite program (SDP) [**6**]. Since the perspective reformulation and tightening will be performed on the diagonal elements of the extracted matrix D, a reasonable metric for the extracted matrix D seems to be to maximize $\sum_{j \in N} d_j = \text{tr}\, D$. This results in the SDP:

$$\max_{d \geqslant 0}\{\sum_{j \in N} d_j \mid Q - \sum_{j \in N} d_j(e_j e_j^T) \succeq 0\}.$$

Reformulation for Separability

In this work we take a different approach to handling nonseparability. Specifically, since $Q \succeq 0$, then $Q = LL^T$ for some lower triangular matrix L. If $Q \succ 0$, then L is nonsingular and unique. We then let $y = L^T x$ and consider the set

$$\begin{aligned} S \;=\; & \{(y, t, z, v) \in \mathbb{R}^n \times \mathbb{R}^n_+ \times \mathbb{B}^n \times \mathbb{R} \mid \\ & v \geqslant \sum_{j \in N} t_j, t_j \geqslant y_j^2 \quad \forall j \in N, 0 \leqslant [L^{-T} y]_j \leqslant z_j, \forall j \in N\}, \end{aligned} \tag{6.7}$$

which is related to X through the transformation $x = L^{-T} y$ and by projecting out the t variables.

One insight is that since L^{-T} is upper triangular, the last row of the inequalities $0 \leqslant [L^{-T} y]_j \leqslant z_j, \forall j \in N$ is of the form $a_{nn} y_n \leqslant z_n$, and thus one could apply the perspective reformulation to tighten S: $z_n t_n \geqslant y_n^2$. Our work uses this simple transformation, in combination with integer programming techniques, to derive strong inequalities for $\text{conv}(X)$. Preliminary computational results will be given in the talk.

Acknowledgments

This work supported in part by the US Department of Energy under grant DE-FG02-08ER25861 and the National Science Foundation by grant CCF-0830153.

References

1. S. Aktürk, A. Atamtürk, and S. Gürel. A strong conic quadratic reformulation for machine-job assignment with controllable processing times. *Operations Research Letters*, 37:187–191, 2009.

2. D. Bienstock. Computational study of a family of mixed-integer quadratic programming problems. *Mathematical Programming*, 74:121–140, 1996.
3. S. Ceria and J. Soares. Convex programming for disjunctive optimization. *Mathematical Programming*, 86:595–614, 1999.
4. D. P. Foster and E. I. George. The risk inflation criterion for multiple regression. *Annals of Statistics*, 22:1947–1975, 1994.
5. A. Frangioni and C. Gentile. Perspective cuts for a class of convex 0-1 mixed integer programs. *Mathematical Programming*, 106:225–236, 2006.
6. A. Frangioni and C. Gentile. SDP diagonalizations and perspective cuts for a class of nonseparable miqp. *Operations Research Letters*, 35:181–185, 2007.
7. A. Frangioni and C. Gentile. A computational comparison of reformulations of the perspective relaxation: SOCP vs. cutting planes. *Operations Research Letters*, 24:105–113, 2009.
8. O. Günlük and J. Linderoth. Perspective relaxation of mixed integer nonlinear programs with indicator variables. In A. Lodi, A. Panconesi, and G. Rinaldi, editors, *IPCO 2008: The Thirteenth Conference on Integer Programming and Combinatorial Optimization, Lecture Notes in Computer Science*, volume 5035, pages 1–16, 2008.
9. R. A. Stubbs. *Branch-and-Cut Methods for Mixed 0-1 Convex Programming.* PhD thesis, Northwestern University, December 1996.

7

Can PENNON be useful when solving global optimization problems?

Michal Kocvara

School of Mathematics,
University of Birmingham
Edgbaston, Birmingham, B15 2TT, UK

kocvara@maths.bham.ac.uk

Keywords: augmented Lagrangian method, global optimization.

I will present examples when the code PENNON can be used for the solution of mixed integer and global optimization problems. PENNON solves convex and nonconvex problems of nonlinear optimization with standard and matrix inequality constraints. In the first class of examples, we first solve a relaxation of the global optimization problem and then, by the same code, the original problem starting from the relaxed solution. In the second class, we use the fact that PENNON is based on the generalized augmented Lagrangian algorithm. Using a suitable solver for the unconstrained subproblems, we can thus derive a global or a derivative free solver for nonlinear and even semidefinite optimization problems. This is a joint work with Michael Stingl and Ralf Werner.

A “joint+marginal” algorithm for 0/1 programs

Jean B. Lasserre **Tung Phan Thanh**

LAAS-CNRS (and Institute of Mathematics, JbL)
Université de Toulouse
7 avenue du Colonel Roche
Toulouse, France

{lasserre,tphantha}@laas.fr

Abstract

We propose a heuristic for 0/1 programs based on the recent “joint + marginal” approach of the first author for parametric polynomial optimization. The idea is to first consider the n-variable $(x_1, \ldots, x_n)$ problem as a $(n-1)$-variable problem $(x_2, \ldots, x_n)$ with the variable x_1 being now a parameter taking value in $\{0, 1\}$. One then solves a hierarchy of what we call “joint+marginal” semidefinite relaxations whose duals provide a sequence of polynomial approximations $x_1 \mapsto J_k(x_1)$ that converges to the optimal value function $J(x_1)$ (as a function of the parameter x_1). One considers a fixed index k in the hierarchy and if $J_k(1) > J_k(0)$ then one decides $x_1 = 1$ and $x_1 = 0$ otherwise. The quality of the approximation depends on how large k can be chosen (in general, for significant size problems, $k = 1$ is the only choice). One iterates the procedure with now a $(n-2)$-variable problem with one parameter $x_2 \in \{0, 1\}$, etc. Preliminary numerical experiments are provided.

Keywords: 0/1 programs, semidefinite relaxations.

1. Introduction

Consider the general 0/1 program

$$\P : \quad f^* = \min_{\mathbf{x}} \{ f(\mathbf{x}) \ : \ \mathbf{x} \in \mathbf{K} \cap \{0, 1\}^n \}$$

where f is a polynomial and $\mathbf{K} \subset \mathbb{R}^n$ is a basic closed semi-algebraic set. One way to approximate the optimal value of $\P$ is to solve a hierarchy of either LP-relaxations as in Sherali-Adams [**11**] and Lovász-Schrijver [**9**], or semidefinite relaxations as in Lovász-Schrijver [**9**] or Lasserre [**6**]. For a comparison of those approches for 0/1 programs, the interested reader is referred to Lasserre [**7**] and Laurent [**8**]. For 0/1 programs, the convergence of the semidefinite relaxations of [**6**] is finite and there is a stopping criterion which, when met, guarantees that the semidefinite relaxation is exact and one may extract global minimizers. Despite practice seems to reveals that the finite convergence is fast, the matrix size in the k-th semidefinite relaxation of the hierarchy grows up as fast as $O(n^k)$. Hence for problems of reasonable size, and in view of the present status of SDP-solvers, one can implement only the first or second relaxation, which in general only provides a lower bound on f^*. Of course, this lower bound can be exploited in some other search procedure, like e.g. branch and bound, but more generally, the following natural question arises:

How can we use the results of the k-th semidefinite relaxation to find (or help find) an approximate solution of the original problem?

In some well-known special cases like e.g. the MAXCUT problem, one my generate a feasible solution with guaranteed performance, e.g. from the Goemans and Williamson randomized rounding procedure [**2**] that uses an optimal solution of the first semidefinite relaxation (i.e. with $k = 1$). But there is no receipe for the general case and one is left with the possibility to use the lower bound provided by the semidefinite relaxation in a standard branch & bound procedure. We here provide a simple heuristic for 0/1 polynomial programs which builds upon the so-called "joint+marginal" approach (in short (J+M)) recently developed in [**4**] for *parametric* polynomial optimization. The (J+M)-approach for polynomial optimization problems with variables $\mathbf{x} \in \mathbb{R}^n$ and parameters $\mathbf{y}$ in a simple set $\mathbf{Y}$, consists of the standard hierarchy of semidefinite relaxations in [**5, 6**] where one treats the parameters $\mathbf{y}$ also as variables but now with the additional constraint that some marginal distribution on $\mathbf{Y}$ (e.g. the uniform probability distribution on $\mathbf{Y}$) is fixed. Among other things, it permits to provide a polynomial approximation of the optimal value function $\mathbf{y} \mapsto J(\mathbf{y})$ (viewed as a function of the parameter). For more details, the interested reader is referred to [**4**].

In the context of a non-parametric 0/1 polynomial optimization, the above (J+M)-approach can be used as follows:

• (a) Treat x_1 as a parameter in $\{0, 1\}$ with distribution $(p_1, 1 - p_1)$ for some given $0 < p_1 < 1$, fixed arbitrary (typically $p_1 = 1/2$).

• (b) solve the k-th semidefinite relaxation of the (J+M)-hierarchy applied to problem ¶ with $n - 1$ variables $x_2, \ldots, x_n$ and parameter $x_1 \in \{0, 1\}$. The dual provides a polynomial map $x_1 \mapsto J_k(x_1)$ that converges to $J(x_1)$ as k increases. (The map $v \mapsto J(v)$ denotes the optimal value function of ¶ given that the variable x_1 is fixed at the value v.) Therefore, to decide if $x_1 = 0$ or 1 in an optimal solution, one replaces the exact test $J(1) > J(0)$ with the approximate test $J_k(1) > J_k(0)$, and of course, the larger k the better; in fact the latter test becomes exact for k sufficiently large.

• (c) If $J_k(1) > J_k(0)$ then fix $\tilde{x}_1 := 1$ (and $\tilde{x}_1 = 0$ otherwise). For feasibility[1], check if there exists $\mathbf{x} \in \mathbf{K} \cap \{0, 1\}^n$ with $x_1 = \tilde{x}_1$; if not then set $\tilde{x}_1 = 0$. (e.g. assuming $0 \in \mathbf{K}$, check whether $\mathbf{x} := (\tilde{x}_1, 0, \ldots, 0) \in \mathbf{K}$.)

• (d) Iterate and go to step (a) with now a 0/1 program ¶$(\tilde{x}_1)$ with $n - 2$ variables $x_3, \ldots, x_n$ and parameter x_2 in $\{0, 1\}$ with distribution $(p_2, 1 - p_2)$, and with $0 < p_2 < 1$ arbitrary. Of course, all data of ¶$(\tilde{x}_1)$ are updated according to the value 1 or 0 taken by $\tilde{x}_1$.

[1]When k is large enough, feasiblity is guaranteed because $+\infty > J_k(\tilde{x}_1) \approx J(\tilde{x}_1)$ implies that there is an optimal solution $\mathbf{x} \in \mathbf{K} \cap \{0, 1\}^n$ with $x_1 = \tilde{x}_1$.

After n iterations, one ends up wih a feasible solution $\tilde{x} = (\tilde{x}_1, \ldots, \tilde{x}_n)$. The computational cost is less than solving n times the k-th semidefinite relaxation in the (J+M)-hierarchy, which is itself of same order than the k-th semidefinite relaxation in the hierarchy defined in [**6**] (in fact it includes only one additional constraint!)

2. The "joint+marginal approach to parametric optimization

Most of the material in this section is taken from [**4**]. Let $\mathbb{R}[\mathbf{x}, \mathbf{y}]$ denote the ring of polynomials in the variables $\mathbf{x} = (x_1, \ldots, x_n)$, and the variables $\mathbf{y} = (y_1, \ldots, y_p)$, whereas $\mathbb{R}[\mathbf{x}, \mathbf{y}]_k$ denotes its subspace of polynomials of degree at most k. Let $\Sigma[\mathbf{x}, \mathbf{y}] \subset \mathbb{R}[\mathbf{x}, \mathbf{y}]$ denote the subset of polynomials that are sums of squares (in short s.o.s.). For a real symmetric matrix $\mathbf{A}$ the notation $\mathbf{A} \succeq 0$ stands for $\mathbf{A}$ is positive semidefinite.

The parametric optimization problem

Let $\mathbf{Y} \subset \mathbb{R}^p$ be a compact set, called the *parameter* set, and for each $\mathbf{y} \in \mathbf{Y}$, fixed, consider the following *parametric* optimization problem:

$$\text{(8.1)} \qquad \P_{\mathbf{y}} : \qquad J(\mathbf{y}) := \inf_{\mathbf{x}} \{ f(\mathbf{x}, \mathbf{y}) \,:\, h_j(\mathbf{x}, \mathbf{y}) \geqslant 0,\ j = 1, \ldots, m \}$$

for some polynomials $f, h_j \in \mathbb{R}[\mathbf{x}, \mathbf{y}]$, $j = 1, \ldots, m$.

The interpretation is as follows: $\mathbf{Y}$ is a set of parameters and for each instance $\mathbf{y} \in \mathbf{Y}$ of the parameter, one wishes to compute an optimal *decision* vector $\mathbf{x}^*(\mathbf{y})$ that solves problem (8.1). Let φ be a Borel probability measure on $\mathbf{Y}$, with a positive density with respect to the Lebesgue measure on $\mathbb{R}^p$ (or with respect to the counting measure if $\mathbf{Y}$ is discrete). For instance

$$\varphi(B) := \left(\int_{\mathbf{Y}} d\mathbf{y} \right)^{-1} \int_{\mathbf{Y} \cap B} d\mathbf{y}, \qquad \forall B \in \mathcal{B}(\mathbb{R}^p),$$

is uniformly distributed on $\mathbf{Y}$. Sometimes, e.g. in the context of optimization with data uncertainty, φ is already specified.

The idea is to use φ (or more precisely, its moments) to get information on the mapping $\mathbf{y} \mapsto J((\mathbf{y})$ and on the distribution of optimal solutions $\mathbf{x}^*(\mathbf{y})$ of $\P_{\mathbf{y}}$, viewed as random vectors.

A related infinite-dimensional linear program

Let $\mathbf{K} \subset \mathbb{R}^n \times \mathbb{R}^p$ be the set:

$$\text{(8.2)} \qquad \mathbf{K} := \{ (\mathbf{x}, \mathbf{y}) \,:\, \mathbf{y} \in \mathbf{Y}\,; \quad h_j(\mathbf{x}, \mathbf{y}) \geqslant 0, \quad j = 1, \ldots, m \},$$

and for each $\mathbf{y} \in \mathbf{Y}$, let

$$\text{(8.3)} \qquad \mathbf{K}_{\mathbf{y}} := \{ \mathbf{x} \in \mathbb{R}^n \,:\, h_{\mathbf{y}j}(\mathbf{x}) \geqslant 0, \quad j = 1, \ldots, m \}.$$

In what follows we assume that for every $\mathbf{y} \in \mathbf{Y}$, the set $\mathbf{K}_{\mathbf{y}}$ in (8.3) is nonempty.

Let $\mathbf{M}(\mathbf{K})$ be the set of finite Borel probability measures on $\mathbf{K}$, and consider the following infinite-dimensional linear program $\P$:

$$\text{(8.4)} \qquad \rho := \inf_{\mu \in \mathbf{M}(\mathbf{K})} \{ \int_{\mathbf{K}} f \, d\mu \,:\, \pi\mu = \varphi \}$$

where $\pi\mu$ denotes the marginal of μ on $\mathbb{R}^p$, that is, $\pi\mu$ is a probability measure on $\mathbb{R}^p$ defined by $\pi\mu(B) := \mu(\mathbb{R}^n \times B)$ for all $B \in \mathcal{B}(\mathbb{R}^p)$. Notice that $\mu(\mathbf{K}) = 1$ for any feasible solution μ of ¶. Indeed, as φ is a probability measure and $\pi\mu = \varphi$ one has $1 = \varphi(\mathbf{Y}) = \mu(\mathbb{R}^n \times \mathbb{R}^p) = \mu(\mathbf{K})$.

The dual of ¶ is the the following infinite-dimensional linear program:

$$(8.5)\qquad \begin{array}{rl} \rho^* := \sup\limits_{p\in\mathbb{R}[\mathbf{y}]} & \displaystyle\int_{\mathbf{Y}} p(\mathbf{y})\, d\varphi(\mathbf{y}) \\ & f(\mathbf{x}) - p(\mathbf{y}) \geqslant 0 \quad \forall(\mathbf{x},\mathbf{y}) \in \mathbf{K}. \end{array}$$

Recall that a sequence of measurable functions (g_n) on a measure space $(\mathbf{Y}, \mathcal{B}(\mathbf{Y}), \varphi)$ converges to g *φ-almost uniformly* if and only if for every $\epsilon > 0$, there is a set $A \in \mathcal{B}(\mathbf{Y})$ such that $\varphi(A) < \epsilon$ and $g_n \to g$ uniformly on $\mathbf{Y} \setminus A$.

Theorem 2.1 ([**4**])**.** Let both $\mathbf{Y} \subset \mathbb{R}^p$ and $\mathbf{K}$ in (8.2) be compact and assume that for every $\mathbf{y} \in \mathbf{Y}$, the set $\mathbf{K}_\mathbf{y} \subset \mathbb{R}^n$ in (8.3) is nonempty. Let ¶ be the optimization problem (8.4) and let $\mathbf{X}^*_\mathbf{y} := \{\mathbf{x} \in \mathbb{R}^n \,:\, f(\mathbf{x},\mathbf{y}) = J(\mathbf{y})\}$, $\mathbf{y} \in \mathbf{Y}$. Then:

(a) $\rho = \displaystyle\int_{\mathbf{Y}} J(\mathbf{y})\, d\varphi(\mathbf{y})$ and ¶ has an optimal solution.

(b) Assume that for φ-almost $\mathbf{y} \in \mathbf{Y}$, the set of minimizers of $\mathbf{X}^*_\mathbf{y}$ is the singleton $\{\mathbf{x}^*(\mathbf{y})\}$ for some $\mathbf{x}^*(\mathbf{y}) \in \mathbf{K}_y$. Then there is a measurable mapping $g : \mathbf{Y} \to \mathbf{K}_\mathbf{y}$ such that

$$(8.6)\qquad g(\mathbf{y}) = \mathbf{x}^*(\mathbf{y}) \text{ for every } \mathbf{y} \in \mathbf{Y}\,; \quad \rho = \int_{\mathbf{Y}} f(g(\mathbf{y}),\mathbf{y})\, d\varphi(\mathbf{y}),$$

and for every $\alpha \in \mathbb{N}^n$, and $\beta \in \mathbb{N}^p$:

$$(8.7)\qquad \int_{\mathbf{K}} \mathbf{x}^\alpha \mathbf{y}^\beta\, d\mu^*(\mathbf{x},\mathbf{y}) = \int_{\mathbf{Y}} \mathbf{y}^\beta\, g(\mathbf{y})^\alpha\, d\varphi(\mathbf{y}).$$

(c) There is no duality gap between (8.4) and (8.5), i.e. $\rho = \rho^*$, and if $(p_i)_{i\in\mathbb{N}} \subset \mathbb{R}[\mathbf{y}]$ is a maximizing sequence of (8.5) then:

$$(8.8)\qquad \int_{\mathbf{Y}} |\, J(\mathbf{y}) - p_i(\mathbf{y})\,|\, d\varphi \to 0 \quad \text{as } i \to \infty.$$

Moreover, define the functions $(\tilde{p}_i)$ as follows:

$$\tilde{p}_0 := p_0, \quad \mathbf{y} \mapsto \tilde{p}_i(\mathbf{y}) := \max\,[\,\tilde{p}_{i-1}(\mathbf{y}), p_i(\mathbf{y})\,], \quad i = 1, 2, \ldots$$

Then $\tilde{p}_i \to J(\cdot)$ φ-almost uniformly.

An optimal solution μ^* of ¶ encodes *all* information on the optimal solutions $\mathbf{x}^*(\mathbf{y})$ of $¶_\mathbf{y}$. For instance, let $\mathbf{B}$ be a given Borel set of $\mathbb{R}^n$. Then from Theorem 2.1,

$$\text{Prob}\,(\mathbf{x}^*(\mathbf{y}) \in \mathbf{B}) = \mu^*(\mathbf{B} \times \mathbb{R}^p) = \int_{\mathbf{Y}} \mathrm{I}_B(g(\mathbf{y}))\, d\varphi(\mathbf{y}) = \varphi[g^{-1}(B) \cap \mathbf{Y}],$$

with g as in Theorem 2.1(b).

Moreover from Theorem 2.1(c), any optimal or nearly optimal solution of ¶* provides us with some polynomial lower approximation (p_i) of the optimal value function $\mathbf{y} \mapsto J(\mathbf{y})$ that converges to $J(\cdot)$ in the $L_1(\varphi)$ norm. Moreover, one may also obtain a piecewise polynomial approximation $(\tilde{p}_i)$ that converges to $J(\cdot)$, φ-almost uniformly.

In [**4**] the first author has defined a (J+M)-hierarchy of semidefinite relaxations $(\mathbf{Q}_k)$ to approximate as closely as desired the optimal value ρ. In particular, the dual of each semidefinite relaxation $\mathbf{Q}_k$ provides a polynomial $p_k \in \mathbb{R}[\mathbf{y}]$ bounded above by $J(\mathbf{y})$, and $\mathbf{y} \mapsto \tilde{p}_k(\mathbf{y}) := \max_{\ell=1,\dots k} p_k(\mathbf{y})$ converges φ-almost uniformly to the optimal value function J. This last property is the rationale behind the heuristic for polynomial 0/1 programs developed below.

3. A "joint+marginal" algorithm for 0/1 polynomial programs

Let $\mathbb{N}^n_i := \{\alpha \in \mathbb{N}^n : |\alpha| \leqslant i\}$ with $|\alpha| = \sum_i \alpha_i$. With a sequence $\mathbf{z} = (z_\alpha)$, $\alpha \in \mathbb{N}^n$, indexed in the canonical basis $(\mathbf{x}^\alpha)$ of $\mathbb{R}[\mathbf{x}]$, let $L_\mathbf{z} : \mathbb{R}[\mathbf{x}] \to \mathbb{R}$ be the linear mapping:

$$f\ (= \sum_\alpha f_\alpha(\mathbf{x})) \ \mapsto\ L_\mathbf{z}(f) \ :=\ \sum_\alpha f_\alpha\, z_\alpha, \qquad f \in \mathbb{R}[\mathbf{x}].$$

Moment matrix. The moment matrix $\mathbf{M}_i(\mathbf{z})$ associated with a sequence $\mathbf{z} = (z_\alpha)$, has its rows and columns indexed in the canonical basis $(\mathbf{x}^\alpha)$, and with entries.

$$\mathbf{M}_i(\mathbf{z})(\alpha,\beta) \ =\ L_\mathbf{z}(\mathbf{x}^{\alpha+\beta}) \ =\ z_{\alpha+\beta}, \quad \forall\, \alpha,\beta \in \mathbb{N}^n_i.$$

Localizing matrix. Let q be the polynomial $\mathbf{x} \mapsto q(\mathbf{x}) := \sum_u q_u \mathbf{x}^u$. The localizing matrix $\mathbf{M}_i(q\,\mathbf{z})$ associated with $q \in \mathbb{R}[\mathbf{x}]$ and a sequence $\mathbf{z} = (z_\alpha)$, has its rows and columns indexed in the canonical basis $(\mathbf{x}^\alpha)$, and with entries.

$$\mathbf{M}_i(q\,\mathbf{z})(\alpha,\beta) \ =\ L_\mathbf{z}(q(\mathbf{x})\mathbf{x}^{\alpha+\beta}) \ =\ \sum_{u\in\mathbb{N}^n} q_u z_{\alpha+\beta+u}, \quad \forall\, \alpha,\beta \in \mathbb{N}^n_i.$$

A sequence $\mathbf{z} = (z_{\alpha\beta}) \subset \mathbb{R}$ is said to have a *representing* finite Borel measure supported on $\mathbf{K}$ if there exists a finite Borel measure μ such that

$$z_\alpha \ =\ \int_\mathbf{K} \mathbf{x}^\alpha\, d\mu, \qquad \forall\, \alpha \in \mathbb{N}^n.$$

3.1. A "joint+marginal" approach

With $\{f, (g_j)_{j=1}^m\} \subset \mathbb{R}[\mathbf{x}]$, let $\mathbf{K} \subset \mathbb{R}^n$ be the basic semi-algebraic set

(8.9) $$\mathbf{K} := \{\mathbf{x} \in \mathbb{R}^n \ :\ g_j(\mathbf{x}) \geqslant 0,\ j = 1,\dots,m\},$$

and consider the 0/1 polynomial optimization problem:

(8.10) $$\P: \qquad f^* \ =\ \min\,\{f(\mathbf{x}) \ :\ \mathbf{x} \in \mathbf{K} \cap \{0,1\}^n\,\}.$$

Let $\mathbf{Y} := \{0,1\}$ and $0 < p < 1$, and with $y \in \mathbf{Y}$, define the parametrized 0/1 polynomial program in $n-1$ variables

(8.11) $$\P_y: \qquad J(y) \ =\ \min_\mathbf{x}\,\{f(\mathbf{x}) \ :\ \mathbf{x} \in \mathbf{K} \cap \{0,1\}^n\,;\, x_1 = y\},$$

or, equivalently $J(y) = \min\,\{f(\mathbf{x}) \ :\ \mathbf{x} \in \mathbf{K}_y\}$, where for every $y \in \mathbf{Y}$:

(8.12) $$\mathbf{K}_y := \{\mathbf{x} \in\ \mathbf{K} \cap \{0,1\}^n \ :\ x_1 = y\}.$$

Observe that by definition, $f^* = \min_y\,\{J(y) \ :\ y \in \mathbf{Y}\} \ =\ \min[J(1), J(0)]$.

Semidefinite relaxations

To compute (or at least approximate) the optimal value ρ of problem ¶ in (8.10), we now provide a hierarchy of semidefinite relaxations in the spirit of those defined in [**5**]. Define the polynomials:

$$\mathbf{x} \mapsto u_k(\mathbf{x}) := x_k^2 - x_k, \qquad k = 1, \dots, n,$$

and let $v_j := \lceil (\deg \mathrm{g_j})/2 \rceil$, $j = 1, \dots, m$. For $i \geqslant \max_j v_j$ and some fixed $p \in (0,1)$, consider the semidefinite program:

$$(8.13) \quad \rho_i = \begin{array}{ll} \inf\limits_{\mathbf{z}} & L_{\mathbf{z}}(f) \\ \text{s.t.} & \mathbf{M}_i(\mathbf{z}) \succeq 0,\ \mathbf{M}_{i-v_j}(g_j\,\mathbf{z}) \succeq 0, \quad j = 1, \dots, m \\ & \mathbf{M}_{i-1}(u_k\,\mathbf{z}) = 0,\ k = 1, \dots, n; \quad L_{\mathbf{z}}(1) = 1,\ L_{\mathbf{z}}(x_1) = p, \end{array}$$

This semidefinite program (8.13) can be simplified as done in [**6**] to work modulo the ideal $\langle (x_i^2 - x_i) \rangle$. Get rid of the constraints $\mathbf{M}_{i-1}(u_k\,\mathbf{z}) = 0$, $k = 1, \dots, n$, and replace every moment variable z_α with z_β where $\beta_i = 1$ if $\alpha_i > 0$, and $\beta_i = 0$ if $\alpha_i = 0$. Next, the row and column of $\mathbf{M}_i(\mathbf{z})$ associated with the monomial $\mathbf{x}^\alpha$ is identical to that associated with $\mathbf{x}\beta$ and can be deleted; same thing with the localizing matrices $\mathbf{M}_i(g_j\mathbf{z})$.

Letting $g_0 := 0$, the dual of (8.13) reads:

$$(8.14) \quad \begin{array}{rl} \rho_i^* = & \sup\limits_{\lambda, (\sigma_i)} \lambda_0 + p\lambda_1 \\ \text{s.t.} & f - \lambda_0 - \lambda_1 x_1 = \sigma_0 + \sum\limits_{j=1}^{m} \sigma_j\, g_j + \sum\limits_{k=1}^{n} \psi_k\, u_k \\ & \sigma_j \in \Sigma[\mathbf{x}],\ \psi_k \in \mathbb{R}[\mathbf{x}], \quad 0 \leqslant j \leqslant m;\ 1 \leqslant k \leqslant n \\ & \deg \sigma_j g_j \leqslant 2i,\ \deg \psi_k \leqslant 2i - 2, \quad 0 \leqslant j \leqslant m;\ 1 \leqslant k \leqslant n. \end{array}$$

Equivalently, and denoting by $\mathbb{R}[x_1]_t$ the space of polynomials of degree at most t, the above dual may be rewritten as:

$$(8.15) \quad \begin{array}{rl} \rho_i^* = & \sup\limits_{q, (\sigma_i)} pq(1) + (1-p)q(0) \quad (= \displaystyle\int_{\mathbf{Y}} q d\varphi) \\ \text{s.t.} & f - q = \sigma_0 + \sum_{j=1}^{m} \sigma_j\, g_j + \sum_{k=1}^{n} \psi_k\, u_k \\ & q \in \mathbb{R}[x_1]_1;\ \sigma_j \in \Sigma[\mathbf{x}],\ \psi_k \in \mathbb{R}[\mathbf{x}], \quad 0 \leqslant j \leqslant m;\ 1 \leqslant k \leqslant n \\ & \deg \sigma_j g_j \leqslant 2i,\ \deg \psi_k \leqslant 2i - 2, \quad 0 \leqslant j \leqslant m;\ 1 \leqslant k \leqslant n. \end{array}$$

Observe that with $I_1 := \langle x_1^2 - x_1 \rangle$ being the ideal of $\mathbb{R}[x_1]$ generated by the polynomial $x_1^2 - x_1$, one may also replace the polynomial q in (8.15) with $\tilde{q} := q + I_1$, i.e. an element of $\mathbb{R}[x_1]/I_1$. Indeed it is also admissible and $p\tilde{q}(1) + (1-p)\tilde{q}(0)$ is the same.

Theorem 3.1. Let $\mathbf{K}$ be as (8.9), $\mathbf{Y} = \{0, 1\}$, and $0 < p < 1$. Assume that for every $y \in \mathbf{Y}$ the set $\mathbf{K}_y$ in (8.12) is nonempty, and consider the semidefinite relaxations (8.15). Then as $i \to \infty$:

(a) $\rho_i^* \uparrow p\, J(1) + (1-p)\, J(0) \quad (= \displaystyle\int_{\mathbf{Y}} J d\varphi)$

(b) Let $(q_i, (\sigma_j^i, \psi_k^i))$ be a nearly optimal solution of (8.15), e.g. such that $pq_i(1) + (1-p)q_i(0) \geqslant \rho_i^* - 1/i$. Then $q_i(y) \leqslant J(y)$ for $y \in \mathbf{Y}$, and

$$\lim_{i\to\infty} q_i(y) \;=\; J(y), \qquad y = 0, 1. \tag{8.16}$$

Moreover if one defines

$$\tilde{q}_0 := q_0, \quad \mathbf{y} \mapsto \tilde{q}_i(y) \;:=\; \max\,[\,\tilde{q}_{i-1}(y), q_i(y)\,], \quad i = 1, 2, \ldots,$$

then $\tilde{q}_i(y) \uparrow J(y)$ for $y = 0, 1$, i.e. pointwise monotone nondecreasing convergence takes place.

Remark 3.2. If the set $\mathbf{K}_y$ is empty for either $x_1 = 0$ or $x_1 = 1$ then $\rho_i = +\infty$ provided i is sufficiently large. Indeed otherwise one may show that $\rho_i = \int_{\mathbf{Y}} J(y)d\phi(y) = (1-p)J(0) + pJ(1)$ for all $i \geqslant i^*$ for some index i^*, in contradiction with $J(0) = +\infty$ or $J(1) = +\infty$. However, if $i < i^*$ one may have $\rho_i < +\infty$ while $J(0) = +\infty$ or $J(1) = +\infty$.

In what follows we will use the primal relaxation (8.13) with index i *fixed.* Hence since i will be fixed, it is important de check whether ρ_i is finite.

A sufficiency test for persistency. In the context of 0/1 deterministic optimization, *persistency* of a boolean variable x_k in a 0/1 program ¶ is concerned with whether one may determine if either $x_k^* = 1$ or $x_k^* = 0$ in any optimal solution $\mathbf{x}^* \in \{0,1\}^n$ of ¶; see e.g. [**1, 10**]. There is a simple sufficient condition to detect whether $x_1 = 0$ (resp. $x_1 = 1$) cannot happen in any feasible solution of ¶ defined in (8.10), in which case one may safely state that $x_1^* = 1$ (resp. $x_1^* = 0$) in *any* optimal solution $\mathbf{x}^* \in \{0,1\}^n$.

Corollary 3.3. Consider the semidefinite relaxation (8.13) with $\mathbf{x} \mapsto f(\mathbf{x}) := x_1$ (resp. $\mathbf{x} \mapsto f(\mathbf{x}) := -x_1$) and without the marginal constraint $L_{\mathbf{z}}(x_1) = p$. Denote its optimal value by ρ_i^0 (resp. ρ_i^1).

If $\rho_i^0 > 0$ (resp. $-\rho_i^1 < 1$) then $x_1^* = 1$ (resp. $x_1^* = 0$) in *any* optimal solution $\mathbf{x}^* \in \{0,1\}^n$ of ¶.

Proof: This is because ρ_i^0 (resp. ρ_i^1) always produces a lower bound on f^*. □

Hence the difficult case is when solving (8.13) with $f = x_1$ and $f = -x_1$, one obtains $\rho_i^0 = 0$ and $\rho_i^1 = -1$, respectively. But then the semidefinite relaxation (8.13) with f as in (8.10) is well-defined, that is, ρ_i is finite.

Corollary 3.4. Let ¶ be the 0/1 problem defined in (8.10) and let ρ_i be the optimal value associated with the semidefinite relaxation (8.13). If $\rho_i^0 = 0$ and $\rho_i^1 = -1$ (as defined in Corollary 3.3) then ρ_i is finite.

Proof: Let $\mathbf{z}_1$ (resp. $\mathbf{z}_2$) be an optimal solution of (8.13) associated with $f = x_1$ (resp. $f = -x_1$) without the marginal constraint $L_{\mathbf{z}}(x_1) = p$, and with optimal value $\rho_i^0 = 0$ (resp. $\rho_i^1 = -1$). Then the sequence $\mathbf{z} = p\mathbf{z}_2 + (1-p)\mathbf{z}_1$ is feasible for (8.13), hence with finite value. Indeed, $L_{\mathbf{z}_1}(x_1) = 0$ and $L_{\mathbf{z}_2}(1) = 1$ so that by linearity $L_{\mathbf{z}}(x_1) = L_{(1-p)\mathbf{z}_1 + p\mathbf{z}_2}(x_1) = p$. □

3.2. The joint+marginal algorithm

Fix i and $0 < p < 1$. Denote by $\mathbf{x}_j \in \mathbb{R}^{n-j+1}$ the vector $(x_j, \ldots, x_n)$. For every $j = 2, \ldots, n$, and $\tilde{\mathbf{x}}_{j-1} = (\tilde{x}_1, \ldots, \tilde{x}_{j-1}) \in \{0,1\}^{j-1}$, let $\tilde{f}_j(\mathbf{x}_j) := f(\tilde{\mathbf{x}}_{j-1}, \mathbf{x}_j)$, and

$\tilde{g}_k^j(\mathbf{x}_j) := g_k(\tilde{\mathbf{x}}_{j-1}, \mathbf{x}_j)$, $k = 1, \ldots, m$. Similarly, let

$$\mathbf{K}_j := \{\mathbf{x}_j \in \mathbb{R}^{n-j+1} : \tilde{g}_k^j(\mathbf{x}_j) \geqslant 0,\ k = 1, \ldots, m\},$$

and let $\P(\tilde{\mathbf{x}}_j)$ denote the problem:

$$\P(\tilde{\mathbf{x}}_j): \quad \rho(\tilde{\mathbf{x}}_j) = \min_{\mathbf{x}_j} \{\tilde{f}_j(\mathbf{x}_j) : \mathbf{x}_j \in \mathbf{K}_j \cap \{0,1\}^{n-j+1}\,;\, x_j = \tilde{x}_j\},$$

i.e. the original problem $\P$ where the variable x_k is fixed at the value $\tilde{x}_k \in \{0,1\}$, $k = 1, \ldots, j$. With $\mathbf{z}$ being a sequence indexed in the monomial basis of $\mathbb{R}[\mathbf{x}_j]$, the associated semidefinite relaxation (8.13) reads:

$$(8.17)\quad \begin{array}{rll} \rho_{ji} = & \inf\limits_{\mathbf{z}} & L_{\mathbf{z}}(\tilde{f}_j) \\ & \text{s.t.} & \mathbf{M}_i(\mathbf{z}) \succeq 0,\ \mathbf{M}_{i-v_k}(\tilde{g}_k^j\,\mathbf{z}) \succeq 0, \quad k = 1, \ldots, m \\ & & \mathbf{M}_{i-1}(u_\ell\,\mathbf{z}) = 0,\ \ell = j, \ldots, n; \quad L_{\mathbf{z}}(1) = 1,\ L_{\mathbf{z}}(x_j) = p, \end{array}$$

with associated dual:

$$(8.18)\quad \begin{array}{rll} \rho_{ji}^* = & \sup\limits_{\lambda,(\sigma_i)} & \lambda_0 + p\lambda_1 \\ & \text{s.t.} & \tilde{f}_j - \lambda_0 - \lambda_1 x_j = \sigma_0 + \sum_{k=1}^m \sigma_k\,\tilde{g}_k^j + \sum_{\ell=j}^n \psi_\ell\, u_\ell \\ & & \sigma_k \in \Sigma[\mathbf{x}_j],\ \psi_\ell \in \mathbb{R}[\mathbf{x}_j], \quad k = 0, \ldots, m;\ \ell = j, \ldots, n \\ & & \deg \sigma_k \tilde{g}_k^j \leqslant 2i,\ \deg \psi_\ell \leqslant 2i - 2, \quad k = 0, \ldots, m;\ \ell = j, \ldots, n. \end{array}$$

The "joint+marginal" algorithm

For simplicity, assume that $\mathbf{x} = 0$ is feasible solution. The algorithm consists of n steps. At step j of the algorithm, the vector $\tilde{\mathbf{x}}_{j-1} = (\tilde{x}_1, \ldots, \tilde{x}_{j-1})$ (already computed) is such that $\tilde{g}_k^j(0) \geqslant 0$ for every $k = 1, \ldots, m$. That is, the vector $\mathbf{x} = (\tilde{\mathbf{x}}_j, 0)$ is a feasible solution of $\P$. For the first step $j = 1$, one has: $\tilde{\mathbf{x}}_0 = \emptyset$, $f_j = f$ and $\tilde{g}_k^1 = g_k$, $k = 1, \ldots, m$:

Step j: **Input:** $\tilde{\mathbf{x}}_{j-1} \in \{0,1\}^{j-1}$. **Output:** $\tilde{\mathbf{x}}_j = (\tilde{\mathbf{x}}_{j-1}, \tilde{x}_j) \in \{0,1\}^j$. Consider the semidefinite relaxation (8.17) for problem $\P(\tilde{\mathbf{x}}_j)$:

- Compute ρ_{ji}^0 for the semidefinite relaxation (8.17) without the moment constraint $L_{\mathbf{z}}(x_j) = p$ (see Corollary 3.3). If $\rho_{ji}^0 > 0$ then set $\tilde{x}_j = 1$ else compute ρ_{ji}^1; if $-\rho_{ji}^1 < 1$ then set $\tilde{x}_j = 0$.
- Else if $\rho_{ji}^0 = 0$ and $\rho_{ji}^1 = -1$ compute ρ_{ji} in (8.17) and extract λ_0, λ_1 from the dual (8.18). If $\lambda_1 < 0$ set $\tilde{x}_j = 1$, otherwise set $\tilde{x}_j = 0$.
- **Feasibility**[2]. If $\tilde{g}_k^j(\tilde{x}_j, 0, \ldots, 0) < 0$ for some $k \in \{1, \ldots, m\}$ then set $\tilde{x}_j = 0$.

Repeat until $j = n$.

Computational complexity. At step j of the (J+M)-algorithm, one has to solve three semidefinite programs (8.17) (two of them without the moment constraint $L_{\mathbf{z}}(x_j) = p$) whose number of variables is $O((n-j+1)^k)$ and with m semidefinite constraints with matrix size at most $O((n-j)^k)$. Observe that the semidefinite program (8.17) has exactly same computational complexity as the standrad k-th

[2]When i is large enough, feasiblity is guaranteed because $+\infty > \rho_{ji} \approx \P(\tilde{\mathbf{x}}_j)$ implies that there is a solution $\mathbf{x}^* \in \mathbf{K} \cap \{0,1\}^n$ with $\mathbf{x}_k^* = \tilde{x}_k$, $k = 1, \ldots, j$. So the above feasibility test is conservative, in case i is not large enough.

semidefinite relaxation for 0/1 programs with $n-j+1$ variables! Indeed the only difference is the single additional (linear) moment constraint $L_{\mathbf{z}}(x_j)=p$.

For instance, for the MAXCUT problem $\max_{\mathbf{x}}\{\mathbf{x}^TQ\mathbf{x}\,:\,\mathbf{x}\in\{-1,1\}^n\}$, and with $k=1$, the semidefinite relaxation (8.17) at step 1 of the (J+M)-algorithm reads

$$\max\left\{\mathrm{trace}\,(Q\,\mathbf{X})\;:\;\mathbf{X}_{ii}=1;\;\left[\begin{array}{cc}1&\mathbf{x}'\\ \mathbf{x}&\mathbf{X}\end{array}\right]\succeq 0;\;\mathbf{X}'=\mathbf{X}\in\mathbb{R}^{n\times n};\;x_1=p\right\},$$

which is the standard Goemans and Williamson (or Shor) semidefinite relaxation with the single additional constraint $x_1=p$. At step j, (8.17) reads

$$\max\left\{\mathbf{c}_j'\mathbf{x}_j+\mathrm{trace}\,(Q_j\,\mathbf{X})\;:\;\mathbf{X}_{ii}=1;\;\left[\begin{array}{cc}1&\mathbf{x}_j'\\ \mathbf{x}_j&\mathbf{X}\end{array}\right]\succeq 0;\;\mathbf{X}'=\mathbf{X}\in\mathbb{R}^{(n-j+1)\times(n-j+1)};\;\tilde{x}_j=p\right\}$$

for some vector $\mathbf{c}_j\in\mathbb{R}^{n-j+1}$.

3.3. The max-gap variant

In what we call the "max-gap" variant of the (J+M)-algorithm, at each step j one may try to optimize the choice of the variable to treat as parameter instead of the simple choice x_1 at $j=1$, then x_2 at $j=2$, etc. For instance, at step $j=1$, solve the semidefinite relaxation (8.17) with x_k as parameter, and get $(\lambda_0^k,\lambda_1^k)$ from an optimal solution of the dual, $k=1,\ldots,n$. Then select the index k for which $|\lambda_1^k|$ is maximum, and fix $\tilde{x}_k=1$ if $\lambda_1^k<0$ and $\tilde{x}_k=0$ otherwise. The rationale behind this variant is that the larger $|\lambda_1^k|$ is, the larger is the approximation $|J_k(0)-J_k(1)|$, and so the more likely the decision $x_k=0$ or $x_k=1$ is correct. Then repeat in the obvious manner with now the remaining variables $(x_1,\ldots,x_{k-1},x_{k+1},\ldots,x_n)$, etc.

4. Computational experiments

We report on a first set of computational experiments on the MAXCUT and k-cluster problems:

The MAXCUT problem. The celebrated MAXCUT problem formally consists of solving the discrete optimization problem

$$\P:\quad\min_{\mathbf{x}}\{\,\mathbf{x}'Q\mathbf{x}\;:\;\mathbf{x}\in\{-1,1\}^n\,\},$$

for some real symmetric matrix $Q=(Q_{ij})\in\mathbb{R}^{n\times n}$.

The entry Q_{ij} of the real symmetric matrix Q is set to zero with probability 1/2 and when different from zero, Q_{ij} is randomly (and independently) generated according to the uniform probability distribution on the interval $[0,10]$.

We have tested the max-gap variant of §3.3 for MAXCUT problems on random graphs with $n=15,20$ and 40 variables. For each value of n, we have generated 50 problems and 100 for $n=40$. In (8.13) the parameter $p\in(0,1)$ is set to 0.5. Let $\mathbf{Q}_1$ denote the optimal value of the primal without the marginal constraint $L_{\mathbf{z}}(x_1)=p$, that is, $\mathbf{Q}_1$ is the Shor's relaxation with famous Goemans and Williamson's 0.878 performance guarantee. Let $\P_1$ denote the cost of the solution $\mathbf{x}\in\{-1,1\}^n$ generated by the "joint+marginal" algorithm[3]. In Table 1 below, we have reported the average relative error $(\P_1-\mathbf{Q}_1)/|\mathbf{Q}_1|$, which as one may see, is comparable with the Goemans and Williamson (GW) ratio.

[3]$\mathbf{Q}_1$ and $\P_1$ were computed with the GloptiPloy software dedicated to solving the generalized problem of moments [**3**]

n	20	30	40
$(\P_1 - \mathbf{Q}_1)/\lvert\mathbf{Q}_1\rvert$	10.3%	12.3%	12.5%

Table 1. Relative error for MAXCUT

The k-cluster problem. We have also tested the (J+M)-algorithm for the k-cluster problem

$$\max_{\mathbf{x}} \{ \mathbf{x}'Q\mathbf{x} \,:\, \mathbf{x} \in \{0,1\}^n;\ \sum_{i=1}^{n} x_i = k \},$$

again for some real symmetric matrix $Q = (Q_{ij}) \in \mathbb{R}^{n\times n}$, and some fixed integer $k \in \mathbb{N}$, $1 \leqslant k < n$.

We have tested the max-gap variant on ten 20-variable problems, randomly generated as for MAXCUT, and with $k = n/2 = 10$. The average relative error $(\P_1 - \mathbf{Q}_1)/\lvert\mathbf{Q}_1\rvert$ was 7.96%.

On two problems with $n = 80$ variables the first variant gave a relative error of 5.73% and 22% respectively, but we have not implement the max-gap variant yet.

References

1. D. Bertsimas, K. Natarajan, and Chung-Piaw Teo. Persistence in discrete optimization under data uncertainty. *Math. Program. Ser. B*, 108, 2005.
2. M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42:1115–1145, 1995.
3. D. Henrion, J. B. Lasserre, and J. Lofberg. Gloptipoly 3: moments, optimization and semidefinite programming. *Optim. Methods and Softw.*, 24:761–779, 2009. http://www.laas.fr/~henrion/software/gloptipoly3/.
4. J.B. Lasserre. A "joint+marginal" approach to parametric polynomial optimization. *SIAM J. Optim.* to appear.
5. J.B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM J. Optim.*, 11:796–817, 2001.
6. J.B. Lasserre. An explicit equivalent positive semidefinite program for nonlinear 0/1 programs. *SIAM J. Optim.*, 12:756–769, 2002.
7. J.B. Lasserre. Semidefinite programming vs. LP relaxations for polynomial programming. *Math. Oper. Res.*, 27:347–360, 2002.
8. M. Laurent. A comparison of the Sherali-Adams, Lovász-Schrijver and Lasserre relaxations for 0/1 programming. *Math. Oper. Res.*, 28:470–496, 2003.
9. L. Lovász and A. Schrijver. Cones of matrices and set-functions for 0-1 optimization problems. *SIAM J. Optim*, 1:166–190, 1991.
10. K. Natarajan, Miao Song, and Chung-Piaw Teo. Persistency and its applications in choice modelling. *Manag. Sci.*, 55:453–469, 2009.
11. H. Sherali and W.P. Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM J. Discr. Math.*, 3:411–430, 1990.

9

Local search in nonlinear combinatorial optimization

Jon Lee

IBM T.J. Watson Research Center
Yorktown Heights, NY, U.S.A.

jonlee@us.ibm.com

The successful development of integer linear programming and combinatorial optimization have been intertwined for decades. As a counterpoint to the recent explosion of work in nonlinear integer programming, we develop approximation algorithms based on simple local-search for nonlinear versions of classical combinatorial optimization problems. Surprisingly, such simple algorithms give strong results. In particular, we obtain new approximation results for optimizing submodular functions subject to knapsack or matroid constraints, and for the nonlinear matroid parity problem, which interestingly has provable exponential complexity for exact optimization.

This talk is based on joint works with Vahab Mirrokni (Google), Viswanath Nagarajan (IBM, Watson), Maxim Sviridenko (IBM, Watson) and Jan Vondrák (IBM, Almaden).

References

1. J. Lee, V. Mirrokni, V. Nagarajan, M. Sviridenko. Maximizing non-monotone submodular functions under matroid or knapsack constraints. *SIAM Journal on Discrete Mathematics*, 23(4):2053–2078, 2010. (Conference version in: Proceedings of the 41st ACM Symposium on Theory of Computing (STOC 2009), pp. 323–332).
2. J. Lee, M. Sviridenko, J. Vondrák. Submodular maximization over multiple matroids via generalized exchange properties. In Dinur, I.; Jansen, K.; Naor, S.; Rolim, J.D.P. (Eds.), *Proceedings of APPROX 2009*, Lecture Notes in Computer Science, **5687**:244–257.
3. J. Lee, M. Sviridenko J. Vondrák. Matroid Matching: the Power of Local Search. *IBM Research Report RC24898*, 11/2009. To appear in: Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC 2010).

10

Experiments with MINLP Branching Techniques

Sven Leyffer

Mathematics and Computer Science Division
Argonne National Laboratory
9700 South Cass Ave
Argonne, IL 60439, USA

`leyffer@mcs.anl.gov`

Abstract

Mixed-integer nonlinear optimization problems arise in a range of scientific and operational applications, ranging from the re-ordering of nuclear fuel rods to the design of wireless networks. We present some novel mixed-integer nonlinear optimization applications, and review existing solution techniques. We present some experiments with nonlinear branch-and-bound branching techniques that lead us to promote a tighter integrating nonlinear solvers into a general branch-and-cut framework.
Keywords: nonlinear programming, mixed-integer nonlinear programming.

1. Introduction and Background

Many scientific, engineering, and public sector applications involve both discrete decisions and nonlinear system dynamics that affect the optimality of the final design. Mixed-integer nonlinear programming (MINLP) optimization problems combine the difficulty of optimizing over discrete variable sets with the challenges of handling nonlinear functions. MINLP is one of the most flexible modeling paradigms available, and an expanding body of researchers and practitioners, including computer scientists, engineers, economists, statisticians, and operations managers, are interested in solving large-scale MINLPs. MINLPs can be conveniently expressed as

$$\text{(10.1)} \qquad \underset{x,y}{\text{minimize}}\ f(x,y) \quad \text{subject to } c(x,y) \leqslant 0,\ x \in X,\ y \in Y \text{ integer},$$

where x, y are the continuous and integer variables, respectively, and X, Y are polyhedral sets. The functions f, c are assumed to be twice continuously differentiable and possibly convex. Surveys of MINLP can be found in [**22, 24, 23**].

Given the generality and flexibility of the model, MINLPs have been proposed for many diverse and important applications. A small subset of these applications includes portfolio optimization [**5, 29**], the design of water distribution networks [**10, 30**], block layout design in the manufacturing and service sectors [**11**], network design with queuing delay constraints [**9**], operational reloading of nuclear reactors [**35**], integrated design and control of chemical processes [**21**], blackout prevention for electrical power systems [**6, 15**], and minimizing the environmental impact of utility plants [**16**].

1.1. New MINLP Applications in Computer Science

Mixed integer nonlinear programs are fast becoming prevalent on the research frontiers of computer science. For example, there are many emerging applications of MINLP in communications research. Problems in wireless bandwidth allocation [**4, 36, 13**], selective filtering, [**37, 38**], network design topology, [**3, 12**], and optical network performance optimization [**17**] can all be cast as MINLPs.

We have begun building a library of MINLP test problems from computer science applications, called DIWAL, see `http://wiki.mcs.anl.gov/NEOS/index.php/DIWAL`. Current applications include:

- Nonlinear optimization of IEEE 802.11 mesh networks [**13**]: A model formulated to plan and optimize IEEE 802.11 broadband access networks.
- Distributed optimization for data-optical networking [**17**]: An model to jointly optimize optical networking provisioning and internet protocol (IP) traffic engineering.
- Energy provisioning and relay node placement for wireless sensor networks [**28**]: An model formulated to determine the optimal placement of provisioned energy amongst local aggregation and forwarding nodes (AFN) and relay nodes (RN) such that the two tiered network lifetime is maximized.
- Capacity fairness for wireless mesh networks [**25**]: A model to assign channels to user nodes and determine power of transmission for mesh routers in a wireless network.

In some cases, these applications require detailed reformulations to avoid or mitigate nonconvexities. Next, we review a particular solution methods for MINLP, namely branch-and-bound, and then present some ideas on how to improve this approach through a tighter integration of the MIP and NLP solves.

2. Nonlinear Branch-and-Bound

Nonlinear branch-and-bound dates back to [**31, 14**]. It is best explained as a tree-search. Initially, all integer restrictions are relaxed and the resulting NLP relaxation is solved. Let the solution be $(\hat{x}, \hat{y})$. If all integer variables, $\hat{y}$, are integral, then we have solved the MINLP. Otherwise, we can choose some non-integral integer to branch on. Branching on, say y_i, is achieved by creating two new NLP problems with added bounds $y_i \leqslant [\hat{y}_i]$ and $y_i \geqslant [\hat{y}_i] + 1$ respectively (where $[a]$ is the largest integer not greater than a). Next, one of these two NLPs is selected and solved, and the process is repeated. We can fathom a node, if one of the following conditions is satisfied.

(1) An infeasible NLP is detected, implying that the whole subtree is infeasible.
(2) An integer feasible node is detected, which provides an upper bound on the optimum of the MINLP.
(3) A lower bound on the NLP solution is greater or equal than the current upper bound, which implies that we cannot find a better solution in this subtree.

After a node has been fathomed the algorithm backtracks to another open node until all nodes are fathomed. Heuristics for selecting a branching variable and nodes are discussed in [**26, 39**].

Typically, every NLP is solved from a previously saved primal-dual solution. In MILP it is sufficient to safe a basis, because a basis uniquely determines a primal-dual iterate for an LP. This situation does not generalize to MINLPs. Given a basis (or active set) is not sufficient to determine a starting point, because the Jacobian also depends on value of the variables, (x, y). In this paper we concentrate on a closer integration of the NLP solver and branch-and-bound, concentrating on one particular branching rule that has proven to be successful in MILP, namely strong branching [**2**].

2.1. Preliminary Experience with Nonlinear Branch-and-Bound

We present some preliminary numerical results that motivate our interest in nonlinear branch-and-bound. We start by noting, that MINLPBB [**18**] is typically outperformed by more modern approaches such as LP/NLP-based branch-and-bound [**34, 8, 32, 1**]. Figure 1 shows that MINLPBB is hopelessly outperformed by newer approaches. It shows a performance profile of several MINLP solvers on a set of medium-sized problems. A performance profile can be interpreted as the probability distribution that a solver is at worst 2^x times worst than the best solver. Solvers, whose lines are towards the left top are best.

We note, that MINLPBB is a fairly simplistic nonlinear branch-and-bound solver. It implements a depth-first tree-search with maximum fractional branching, which has been shown to be notoriously poor. Strong branching is usually superior to maximum fractional branching for solving MILPs [**2**]. We can readily generalize strong branching to MINLP. Given a solution of parent node NLP, P, with optimum value f^p, we perform the following steps:

(1) Find all *non-integral* integer variables $y_i, i \in C$.
(2) For every candidate $y_i \in C$ solve *two child NLPs*:
   - A down NLP: $P \cup \{y_i = \lfloor y_i \rfloor\}$ with optimal value f_i^-.
   - An up NLP: $P \cup \{y_i = \lfloor y_i \rfloor + 1\}$ with optimal value f_i^+.
(3) For every candidate $y_i \in C$ compute its score:

$$\text{score}_i := (1-\mu)\min(f_i^- - f^p,\ f_i^+ - f^p) \ +\ \mu\max(f_i^- - f^p,\ f_i^+ - f^p),$$

where $\mu = 1/6$.
(4) Branch on the variable y_i that maximizes score_i.

The goal of this procedure is to maximize the change in the objective, and select branching variables that changes problem the most [**2**].

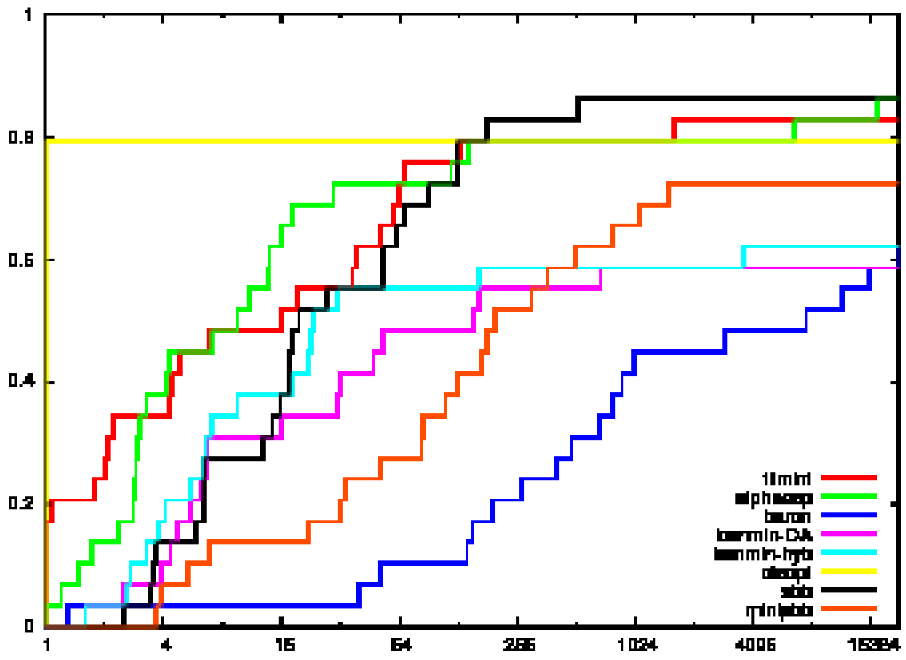


Figure 1. Performance profile of CPU time of several MINLP solvers on a set of medium-sized problems.

Figure 2 shows the effect of strong branching for nonlinear branch-and-bound. The number of nodes in the tree is reduced significantly compared to maximum-fractional branching. However, the additional CPU time needed to solve these NLPs, even using SQP warm-starts is still prohibitive, and strong branching is outperformed even by maximum fractional branching. The plots also show pseudo-cost branching, which outperforms both other options.

Motivated by these observations, we next consider a closer integration of the NLP solver within nonlinear branch-and-bound to reduce the CPU time required for strong branching.

2.2. Challenges in Integrating NLP and MIP

We have already mentioned that in NLP, we cannot generate a vertex or primal-dual solution simply from a knowledge of the basis, or active set. The reason is that even given an optimal active set, we still need to solve a nonlinear problem (using e.g. Newton's method) to obtain its solution, whereas in LP, we simply update basis factors, and perform a forward and a backward solve with the basis.

In principle, NLP solvers also compute factors that could be re-used. Unfortunately, these factors are *always outdated after a solve.* To see why, consider a simple Newton iteration. At iteration k, we factor the Jacobian matrix, and compute a step, $z_{k+1} = z_k + d$. If z_{k+1} satisfies our stopping criterion, then we exit the solver without forming new factors. This situation is exacerbated in NLP, where we not only have outdated factors, but the convergence test requires us to update

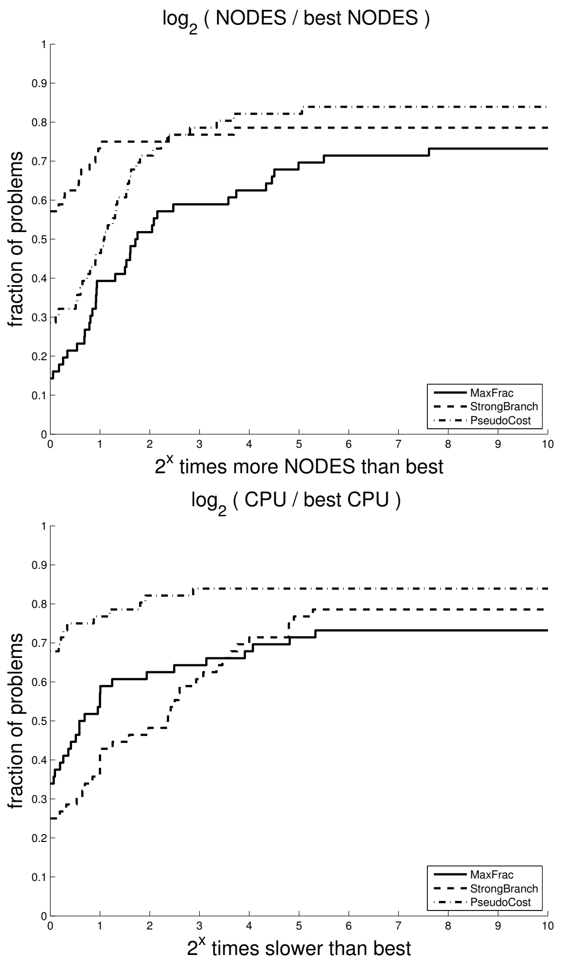


Figure 2. Performance profile of comparing strong branching and maximum-fractional branching. The left plot shows the number of nodes, the right shows the CPU time.

the gradients (i.e Jacobian), so that factors and the stored matrices are out-of-sink after an NLP solve.

3. Integrating NLP and MIP

Our NLP solver is a sequential quadratic programming (SQP) method, see [**27, 33, 7**]. SQP methods successively minimize a quadratic model, $m_k(x)$, subject to a linearization of the constraints about $z_k = (x_k, y_k)$. We define the displacement $d := z - z_k$ and obtain the QP

$$\text{(10.2)} \qquad \underset{d}{\text{minimize}}\ m_k(d) := g_k^T d + \frac{1}{2} d^T H_k d \quad \text{subject to } c_k + A_k^T d \leqslant 0,$$

where $g_k = \nabla f(x_k, y_k)$ is the objective gradient, $c_k = c(x_k, y_k)$ are the values of the constraints, $A_k = \nabla c(x_k, y_k)$ is the Jacobian matrix, $H_k \simeq \nabla^2 L(z_k, \lambda_k)$ approximates the Hessian of the Lagrangian, and λ_k is the multiplier estimate at iteration k. The new iterate is $z_{k+1} = z_k + d$, together with the multipliers λ_{k+1} of the linearized constraints of (10.2).

We use the SQP solver FilterSQP [**19**] which implements a trust-region SQP method. Convergence is enforced with a filter [**20**], whose components are the ℓ_1-norm of the constraint violation, and the objective function.

There are two obvious ways how we can improve strong branching. The first, is to replace the costly NLP solve for every problem on the list of candidates branching variables, C, by a single QP solve. The second approach is to go one step further, and re-use as much of the final QP solve from the previous iteration.

3.1. Approximate Strong Branching

The simplest way to improve strong-branching is by replacing a complete NLP solve by a single iteration of SQP. Recall, that we have already solved the parent problem, so we have a reasonable approximation of the solution that we obtained, if we branched on one variable. This approach is readily implemented. However, because the Hessian, H_k, and the Jacobian, A_k are outdated, we cannot readily re-use their factors (which are available after a solve with FilterSQP), and only perform a warm-start in which we send the final optimal active set to the QP solver. We refer to this kind of branching as approximate strong branching. Some special care has to be taken, because every solve is only an approximate NLP, so the usual fathoming rules during strong branching have to be adapted in an obvious way.

Our preliminary numerical results in Figure 3 show that approximate strong branching (black line) improves on strong branching, and is almost competitive with the simpler pseudo-cost branching both in terms of number of nodes and CPU time.

We can improve our branching decisions further, by adapting reliability branching to NLP. Reliability branching computes pseudo-cost estimates by strong branching until the resulting pseudo-cost estimate is deemed sufficiently reliable (measured by the number of times pseudo-costs have been updated for each integer variable). We use a threshold of 2 in our experiments, and only apply approximate strong branching, rather than complete NLP solves. The results are displayed in Figure 4.

It is clear from Figure 4, that reliability branching is the method of choice for MINLP. We are currently investigating the optimal choice of the reliability parameters for MINLP. Next, we present an approach that allows us to reuse the factors of the final QP solve, in an attempt to gain further performance advances.

3.2. Hot-Starting QP Solves

The re-use of existing factors of the previous QP solves is the most appealing way to obtain pseudo-cost estimates. In our implementation, after solving the parent NLP, we resolve the final QP to synchronize the factors with the solution of the NLP, and then store these factors so that we can re-use them in every QP during the strong-branching phase. We use a special feature in the QP solver that allows us to hot-start the QP and is comparable to a dual-active-set method.

Table 1 shows the CPU times for some reasonably-sized QP approximations. The first column gives the problem name, # ints shows the number of integer variables, and the next three columns give the CPU times for full NLP solve, single QP solve, and a hot-started QP solve, respectively. These results show that the

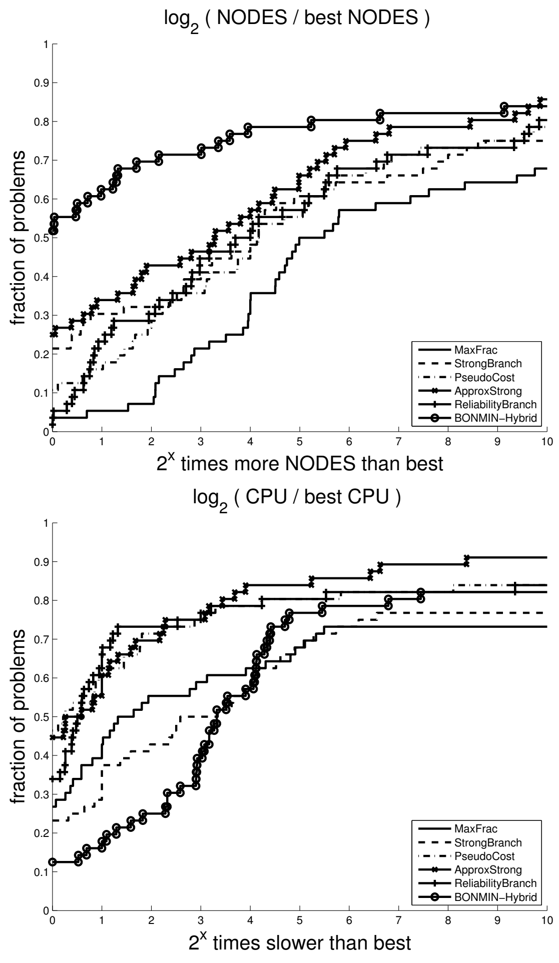


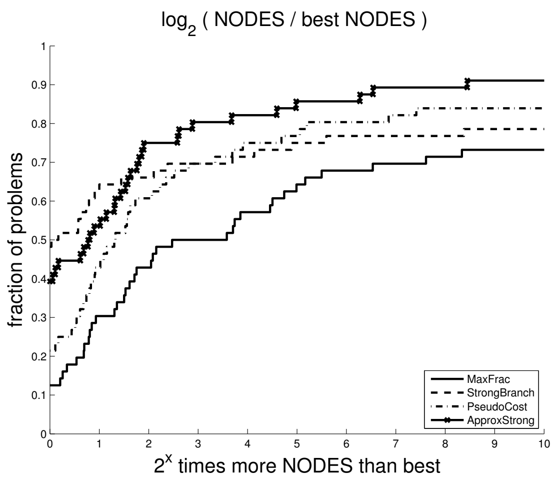
Figure 3. Performance profile of comparing approximate strong branching with strong branching, maximum-fractional branching, and pseudo-cost branching. The left plot shows the number of nodes, the right shows the CPU time.

benefit obtained by solving just a single QP is only a factor two or three, whereas hot-started QPs are faster by a factor of up to 40!

Table 1. CPU times [s] for full NLP solve, single QP solve, and hot-started QP solve.

problem	# ints	Full NLP	Single QP	Hot QP
stockcycle	480	4.08	3.32	0.532
RSyn0805H	296	78.7	69.8	1.94
SLay10H	180	18.0	17.8	1.25
Syn30M03H	180	40.9	14.7	2.12

We believe that these preliminary results are encouraging, as they hold the promise of cheaper strong branching decision for the whole tree. An alternative use



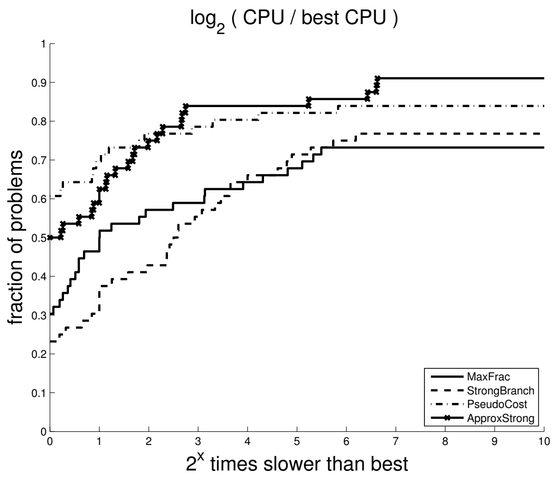


Figure 4. Performance profile of comparing approximate strong branching with strong branching, maximum-fractional branching, and pseudo-cost branching. The left plot shows the number of nodes, the right shows the CPU time.

of hot-started QPs that we are exploring is to replace the NLP-based tree-search by a QP-based tree-search with only occasional updates to compute bounds. We believe that this approach may become competitive with the prevalent approaches to MINLP that use LP-based tree-search techniques.

4. Conclusions

We have presented new MINLP applications arising in computer science, including the optimization of IEEE 802.11 mesh Networks, the design of data-optical networks, the optimization of energy provisioning in relay node placements for wireless sensor networks, and the optimal assignment of channels to users for mesh routers in a wireless network. These models form part of DIWAL, see `http://wiki.mcs.anl.gov/NEOS/index.php/DIWAL`.

We have investigated the tighter integration of MIP and NLP solvers for the solution of these problems. In particular, we have shown that simple heuristics for performing strong branching based on single QP information is superior to strong branching based on NLP solves.

Acknowledgments

This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357. This work was also supported by the U.S. Department of Energy through grant DE-FG02-05ER25694 and by the National Science Foundation through grant 0830035.

References

1. K. Abhishek, S. Leyffer, and J. T. Linderoth. FilMINT: An outer-approximation-based solver for nonlinear mixed integer programs. Preprint ANL/MCS-P1374-0906, Mathematics and Computer Science Division, Argonne National Laboratory, 2006.
2. T. Achterberg, T. Koch, and A. Martin. Branching rules revisited. *Operations Research Letters*, 33:42–54, 2004.
3. D. Bertsekas and R. Gallager. *Data Networks.* Prentice-Hall, Endlewood Cliffs, NJ, 1987.
4. Randeep Bhatia, Adrian Segall, and Gil Zussman. Analysis of bandwidth allocation algorithms for wireless personal area networks. *Wireless Networks*, 12:589603, 2006.
5. D. Bienstock. Computational study of a family of mixed-integer quadratic programming problems. *Mathematical Programming*, 74:121–140, 1996.
6. D. Bienstock and S. Mattia. Using mixed-integer programming to solve power grid blackout problems. *Discrete Optimization*, 4:115–141, 2007.
7. P.T. Boggs and J.W. Tolle. Sequential quadratic programming. *Acta Numerica*, 4:1–51, 1995.
8. P. Bonami, L. T. Biegler, A. R. Conn, G. Cornuéjols, I. E. Grossmann, C. D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, and A. Wächter. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 2008. To appear.
9. R. Boorstyn and H. Frank. Large-scale network topological optimization. *IEEE Transactions on Communications*, 25:29–47, 1977.
10. C. Bragalli, C. D'Ambrosio, J. Lee, A. Lodi, and P. Toth. An MINLP solution method for a water network problem. In *Algorithms - ESA 2006 (14th Annual European Symposium. Zurich, Switzerland, September 2006, Proceedings)*, pages 696–707. Springer, 2006.
11. I. Castillo, J. Westerlund, S. Emet, and T. Westerlund. Optimization of block layout deisgn problems with unequal areas: A comparison of milp and minlp optimization methods. *Computers and Chemical Engineering*, 30:54–69, 2005.
12. Kaikai Chi, Xiaohong Jiang, Susumu Horiguchi, and Minyi Guo. Topology design of network-coding-based multicast networks. *IEEE Transactions on Mobile Computing*, 7(4):1–14, 2008.

13. Enrique Costa-Montenegro, Francisco J. González-Casta no, Pedro S. Rodr ́guez-Hernández, and Juan C. Burguillo-Rial. Nonlinear optimization of ieee 802.11 mesh networks. In Y. Shi et al., editor, *ICCS 2007, Part IV*, number 4490 in LNCS, pages 466–473. Springer Verlag, Berlin Heidelberg, 2007.
14. R. J. Dakin. A tree search algorithm for mixed integer programming problems. *Computer Journal*, 8:250–255, 1965.
15. V. Donde, V. Lopez, B. Lesieutre, A. Pinar, C. Yang, and J. Meza. Identification of severe multiple contingencies in electric power networks. In *Proceedings 37th North American Power Symposium*, 2005. LBNL-57994.
16. A. M. Eliceche, S. M. Corvalán, and P. Martínez. Environmental life cycle impact as a tool for process optimisation of a utility plant. *Computers and Chemical Engineering*, 31:648–656, 2007.
17. A. Elwalid, D. Mitra, and Qiong Wang. Distributed nonlinear integer optimization for data-optical internetworking. *IEEE Journal on Selected Areas in Communications*, 24(8):1502–1513, 2006.
18. R. Fletcher and S. Leyffer. Minlp (ampl input). `http://www-neos.mcs.anl.gov/neos/solvers/MINCO:MINLP-AMPL`.
19. R. Fletcher and S. Leyffer. User manual for filterSQP. Numerical Analysis Report NA/181, University of Dundee, April 1998.
20. R. Fletcher and S. Leyffer. Nonlinear programming without a penalty function. *Mathematical Programming*, 91:239–270, 2002.
21. A. Flores-Tlacuahuac and L. T. Biegler. Simultaneous mixed-integer dynamic optimization for integrated design and control. *Computers and Chemical Engineering*, 31:648–656, 2007.
22. C.A. Floudas. *Nonlinear and Mixed–Integer Optimization.* Topics in Chemical Engineering. Oxford University Press, New York, 1995.
23. I. E. Grossmann. Review of nonlinear mixed-integer and disjunctive programming techniques. *Optimization and Engineering*, 3:227–252, 2002.
24. I. E. Grossmann and Z. Kravanja. Mixed–integer nonlinear programming: A survey of algorithms and applications. In A.R. Conn L.T. Biegler, T.F. Coleman and F.N. Santosa, editors, *Large–Scale Optimization with Applications, Part II: Optimal Design and Control*, New York, Berlin, 1997. Springer.
25. Wenxuan Guo and Xinming Huang. Achieving capacity fairness for wireless mesh networks. *Wirel. Commun. Mob. Comput.*, 2009.
26. O. K. Gupta and A. Ravindran. Branch and bound experiments in convex nonlinear integer programming. *Management Science*, 31:1533–1546, 1985.
27. S.P. Han. A globally convergent method for nonlinear programming. *Journal of Optimization Theory and Applications*, 22(3):297–309, 1977.
28. Y. Thomas Hou, Yi Shi, Hanif D. Sherali, and Scott F. Midkiff. On energy provisioning and relay node placement for wireless sensor networks. *IEEE Transactions on Wireless Communications*, 4(5), 2005.
29. N. J. Jobst, M. D. Horniman, C. A. Lucas, and G. Mitra. Computational aspects of alternative portfolio selection models in the presence of discrete asset choice constraints. *Quantitative Finance*, 1:489–501, 2001.
30. R. Karuppish and I. E. Grossmann. Global optimization for the synthesis of integrated water systems in chemical processes. *Computers and Chemical Engineering*, 30:650–673, 2006.

31. A. H. Land and A. G. Doig. An automatic method for solving discrete programming problems. *Econometrica*, 28:497–520, 1960.
32. S. Leyffer. Generalized outer approximation. In C.A. Floudas and P.M. Pardalos, editors, *Encyclopedia of Optimization*, volume 2, pages 247–254. Kluwer Academic Publishers, 2001.
33. Powell, M.J.D. A fast algorithm for nonlinearly constrained optimization calculations. In G.A. Watson, editor, *Numerical Analysis, 1977*, pages 144–157, Berlin, 1978. Springer–Verlag.
34. I. Quesada and I. E. Grossmann. An LP/NLP based branch–and–bound algorithm for convex MINLP optimization problems. *Computers and Chemical Engineering*, 16:937–947, 1992.
35. A. J. Quist, R. van Gemeert, J. E. Hoogenboom, T. Ílles, C. Roos, and T. Terlaky. Application of nonlinear optimization to reactor core fuel reloading. *Annals of Nuclear Energy*, 26:423–448, 1998.
36. Waseem Sheikh and Arif Ghafoor. An optimal bandwidth allocation and data droppage scheme for differentiated services in a wireless network. ECE Technical Report 349, Purdue University, 2007.
37. Rajnish Sinha, Aylin Yener, and Roy D. Yates. Noncoherent multiuser communications: Multistage detection and selective filtering. *EURASIP Journal on Applied Signal Processing*, 12:14151426, 2002.
38. Majid Soleimanipour, Weihua Zhuang, and George H. Freeman. Optimal resource management in wireless multimedia wideband CDMA systems. *IEEE Transactions on Mobile Computing*, 1(2):143–160, 2002.
39. O. V. Volkovich, V. A. Roshchin, and I. V. Sergienko. Models and methods of solution of quadratic integer programming problems. *Cybernetics*, 23:289–305, 1987.

Solving convex bound constrained MINLP problems

Giampaolo Liuzzi[1] Sara Mattia[2] Laura Palagi[2] Veronica Piccialli[3]

[1] IASI-CNR, viale Manzoni, 30
Rome, 00185, Italy
liuzzi@iasi.cnr.it

[2] Dipartimento di Informatica e Sistemistica
Sapienza University of Rome, via Ariosto, 25
Rome, 00185, Italy
{mattia,palagi}@dis.uniroma1.it

[3] Dipartimento di Ingegneria dell'Impresa
University of Rome Tor Vergata, via del Politecnico, 1
Rome, 00133, Italy
piccialli@disp.uniroma2.it

Abstract

In the paper we present an exact method for solving mixed integer bound constrained problems with convex objective function. We develop a branch-and-bound algorithm where the lower bounds are determined by means of nonlinear techniques. Computational results are reported.

Keywords: Mixed Integer Nonlinear Programming, convex optimization.

1. Introduction

In recent years, Mixed Integer NonLinear Programming (MINLP) problems have been attracting more and more attention from the mathematical programming community, see, for example, the recent survey papers [**2**], [**12**] and the references therein. A MINLP problem is an optimization problem where the objective function and/or the constraints are nonlinear and some of the decision variables are restricted to assume integer values. These are challenging problems that can accurately model many real-world applications in chemical engineering (process design [**16**], product portfolio optimization [**19**]), electronics (signal synthesis [**4**]), finance (portfolio management and optimization [**15**]), unit-commitment of electricity generators [**10, 6**].

MINLP problems include as special cases NP-Hard problems. In the particular case of linear objective and constraints we get the well-known and well-studied class of Mixed Integer Linear Programming (MILP) problems. Research advances for MILP problems over the past years have led to the development of many both commercial and open-source codes able to solve large-scale MILP problems. On the other hand, by relaxing the integrality requirement on the variables, we get NonLinear Programming (NLP) problems for which in recent years different and efficient approaches have been proposed allowing the definition of many software packages and routine libraries for their solution.

In this work we focus our attention on convex MINLP and more specifically on the minimization of a convex objective function with bound and integrality constraints on the variables, that is

$$\begin{array}{ll} \min & f(x) \\ & l_i \leqslant x_i \leqslant u_i, \quad i \in \{1,\dots,n\} \\ & x_i \in D \subset \mathbb{Z}, \quad i \in \mathcal{I} \subseteq \{1,\dots,n\}, \end{array} \tag{11.1}$$

where D is a subset of the integer numbers $\mathbb{Z}$, $\mathcal{I}$ is the index set of integer variables and $f(x)$ is convex. If $f(x)$ is a quadratic function then problem (11.1) includes as a special case the max-cut problem and hence the quadratic boolean optimization problem, see for example [**3**].

2. Existing methods

In the literature there are some exact methods for solving convex MINLP, and they can be roughly classified in NLP based branch-and-bound methods [**14**], generalized Benders decomposition [**11**], Outer Approximation decomposition algorithms [**7**] ,[**9**], [**20**] and extended cutting plane method [**22**]. The branch and bound based methods use as ingredient the branch and bound algorithm originally introduced for solving MILP, and at each node a continuous (linear or nonlinear) relaxation of problem (11.1) is solved. Then the search space is explored by partitioning the set of solutions in smaller subset as in the standard branch and bound. This class of methods is implemented in solvers MINLP-BB [**18**], SBB [**5**] and BONMIN [**1**].

The outer approximation method solves a sequence of MILP problems built by linearizing the objective and constraint functions around a set of solutions of the NLP obtained by fixing the integer variables. This method is implemented in the software DICOPT [**13**] and in BONMIN [**1**]. The generalized Benders decomposition also solves a sequence of MILP, but instead of using a linearization of each constraint, it introduces in the master problem a single constraint combining all the linearizations. Finally, the extended cutting plane method is an extension of the cutting plane method [**17**] for solving convex NLPs. The algorithm is based on the iterative solution of a master problem where linearizations of violated constraints are added during the iterations. This method is implemented in the software α-ECP [**21**].

3. Our algorithm

Our algorithm belongs to the class of the NLP based branch and bound algorithms. In particular, we implement a branch and bound method where the lower bound at each node is computed by solving the bound constrained continuous NLP problem obtained by removing the integrality constraint in problem (11.1). This is done by means of the algorithm introduced in [**8**] which is an active set-type algorithm. Indeed, at each iteration k, the algorithm estimates the variables that will be supposedly at their lower and upper bounds at the (continuous) solution. On the basis of this partition, the algorithm performs (sort of) an unconstrained minimization in the subspace of free variables using a local Newton-type method. The quality of the points generated by the algorithm is assessed by using a very simple differentiable exact penalty function which allows also to generate infeasible points with a prefixed "degree of infeasibility" during the continuous optimization. From a computational point of view, it turns out that the algorithm is able to rapidly identify the active bounds at the solution. We exploit this information in the branching phase and define an heuristic for upper bound computation that keeps into account the structure of the problem at hand. We evaluate our method on a set of test problems and compare it with other existing branch-and-bound methods.

References

1. P. Bonami, L.T. Biegler, A.R: Conn, G. Cornuéjouls, I.E. Grossmann, C.D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, and A. Wachter. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5:186–204, 2008.
2. P. Bonami, M. Kilinc, and J. Linderoth. Algorithms and software for convex mixed integer nonlinear programs. Technical Report 1664, University of Wisconsin, Madison, 2009.
3. E. Boros and P. L. Hammer. The max-cut problem and quadratic 0?1 optimization; polyhedral aspects, relaxations and bounds. *Annals of Operations Research*, 33:227–252, 1991.
4. C. Buchheim, A. Caprara, and A. Lodi. An effective branch-and-bound algorithm for convex quadratic integer programming. Technical Report IPCO 2010, 2010.
5. M. R. Bussiek and A. Drud. Sbb: a new solver for for mixed integer nonlinear programming. Technical report, OR2001, 2001.
6. G.M. Casolino, G. Liuzzi, and A. Losi. Unit commitment in oligopolistic markets by nonlinear mixed variable programming. *Optimization and Engineering*, 2009. DOI 10.1007/s11081-009-9102-6, to appear.
7. M.A. Duran and I.E. Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming*, 36:307–339, 1986.
8. F. Facchinei, S. Lucidi, and L. Palagi. A truncated newton algorithm for large scale box constrained optimization. *SIAM Journal on Optimization.*, 12(4):1100–1125, 2002.
9. IR. Fletcher and S. Leyffer. Solving mixed integer nonlinear programs by outer approximation. *Mathematical Programming*, 66:327–349, 1994.
10. A. Frangioni and C. Gentile. Solving nonlinear single-unit commitment problems with ramping constraints. *Operations Research*, 54(4):767–775, 2006.
11. A. Geoffrion. Generalized benders decompositions. *Journal of Optimization Theory and Applications*, 10:237–260, 1972.

12. I.E. Grossmann. Review of nonlinear mixed-integer and disjunctive programming techniques. *Optimization and Engineering*, 3:227–252, 2002.
13. I.E. Grossmann, J. Viswanathan, A.V. Raman, and E. Kalvelagen. Gams/dicopt: A discrete continuous optimization package. *Mathematical Methods in the Applied Sciences*, 11:649–664, 2001.
14. O. K. Gupta and A. Ravindran. Branch and bound experiments in convex nonlinear integer programming. *Management Science*, 31:1533–1546, 1985.
15. N.J. Jobst, M.D. Horniman, C.A. Lucas, and G. Mitra. Computational aspects of alternative portfolio selection models in the presence of discrete asset choice constraints. *Quantitative Finance*, 1:489–501, 2001.
16. J. Kallrath. Mixed integer optimization in the chemical process industry: experience, potential and future. *International Journal on Chemical Engineering*, 78 part A:809–822, 2000.
17. J.E. Kelley. The cutting plane method for solving convex programs. *Journal of SIAM*, 8:703–712, 1960.
18. S. Leyffer. User manual for minlp-bb. Technical report, University of Dundee, 1998.
19. X. Lin, C.A. Floudas, and J. Kallrath. Global solution approach for a nonconvex minlp problem in product portfolio optimization. *Journal of Global Optimization*, 32:417–431, 2005.
20. I. Quesada and I.E. Grossmann. An lp/nlp based branch-and-bound algorithm for for convex minlp optimization problems. *Computers and Chemical Engineering*, 16:937–947, 1992.
21. T. Westerlund and K. Lundqvist. Alpha-ecp, version 5.101. an interactive minlp-solver based on the extended cutting plane method. Technical Report 01-178-A, Process Design Laboratory, Abo Akademi University, 2005.
22. T. Westerlund and F. Petterson. A cutting plane method for solving convex minlp problems. *Computers and Chemical Engineering*, 19:s131–s136, 1995.

12

Computing dense subgraphs with semidefinite programming

Jérôme Malick[1] **Frédéric Roupin**[2]

[1] CNRS, LJK (Lab. J. Kunztmann)
51, rue des maths, 38041 Grenoble, France
`jerome.malick@inria.fr`

[2] CNAM, CEDRIC Lab.
292, rue St-Martin, 75003 Paris, France
`frederic.roupin@cnam.fr`

Abstract

We present a method for finding densest subgraphs, a classical NP-hard problem in combinatorial optimization. This problem consists in finding the densest subgraph with k nodes in a unweighted graph. We use a branch-and-bound approach that applies a new bounding procedure, based on recent semidefinite programming techniques. These semidefinite bounds are less accurate than standard semidefinite bounds, but cheaper to get. The experiments show that our method is competitive with the best existing approaches.

Keywords: combinatorial optimization, 0-1 quadratic optimization, graphs, semidefinite optimization, Lagrangian duality, branch-and-bound, k-densest subgraph, k-cluster.

1. A difficult combinatorial optimization problem

In this communication, we focus on the following standard problem of combinatorial optimization. Let $G = (V, E)$ be an undirected, unweighted graph with n vertices. Given an integer $1 \leqslant k \leqslant n$, we want to compute a subgraph of G with k vertices and with as many edges as possible. This problem can be formulated as the $\{0,1\}$-quadratic optimization problem:

$$(k\text{-densest}) \begin{cases} \max \quad y^\top W\, y \\ \quad y_1 + \cdots + y_n = k \\ \quad y \in \{0,1\}^n \end{cases}$$

where W is the (half) adjacency matrix of G. The densest subgraph problem is a difficult problem of combinatorial optimization (NP-hard and more, see [**5**]). Relaxations and approaches using convex optimization have been proposed to solve it exactly or approximately (e.g. [**1**] using linear optimization, [**4**] using semidefinite optimization and [**2**] using convex quadratic optimization). Moreover, [**10**] notices that semidefinite relaxations of this problems give very tight bounds which are also very expensive to compute.

The objectives of this communication are:

(1) to present new bounds, trading tightness for cpu time, while keeping SDP-like quality,
(2) to show their interest for solving densest subgraph problem by brand-and-bound,
(3) to compare with the best exact resolution method [**2**].

2. Reformulations as a special semidefinite optimization problem

We propose a new convex relaxation of the densest subgraph problem. This relaxation is SDP-like, but differs intrinsically from the other SDP bounds. In this section, we reformulate the problem as needed, and in the next section we relax it. We start reformulating with strandard techniques:

- enforcement of contraints - by adding n additional product-contrainsts,
- change of variable - to have a $\{-1, 1\}$ formulation,
- homogenisation - to get a pure quadratic problem (in $\mathbb{R}^{n+1}$).

Then we apply the standard "lifting" (see e.g. [**3**]) in the space of symmetric matricesof size $n+1$, equipped with the inner product $\langle X, Y\rangle = \mathrm{trace}(XY)$, to get the reformulation

$$\text{(SDP)} \begin{cases} \max \quad \langle Q, X\rangle \\ \quad \langle Q_j, X\rangle = 4k - 2n, \quad j \in \{0, \dots, n\} \\ \quad \langle E_i, X\rangle = 1, \quad i \in \{0, \dots, n\} \\ \quad \mathrm{rank}(X) = 1, \quad X \succeq 0, \end{cases}$$

with Q, Q_j, E_i that are special symmetric matrices of size $n+1$ (see the details in [**9**]). The final step consists in noting that, in this situation, the rank-one constraint is equivalent to the constraint $\|X\|^2 = (n+1)^2$, called the "spherical constraint" [**7**]. Thus we have reformulated our $\{0, 1\}$ quadratic optimization problem as an equivalent linear SDP problem with one quadratic constraint.

3. Lagrangian duality and bounds

The Lagrangian dualization of the spherical constraint by real parameter α introduces a family of SDP bounds with interesting properties (see the details in [**9**]). The bound denoted $\Theta(\alpha)$ is always less accurate that the standard SDP bound, but it is very close when α is small. Moreover it is computed easily by solving a SDP least-squares problem

$$\begin{cases} \min \quad \|X - Q/\alpha\|^2 \\ \quad \langle Q_j, X\rangle = 4k - 2n, \; j \in \{0, \dots, n\} \\ \quad \langle E_i, X\rangle = 1, \; i \in \{0, \dots, n\} \\ \quad X \succeq 0 \end{cases}$$

with the help of efficient algorithms [**6**]. The parameter α acts as a cursor trading tightness for computation time, which is highly appreciable in exact solving procedure.

4. Exact resolution by branch-and-bound

We propose a branch-and-bound algorithm to solve the densest subgraph problem to optimality. The characteristics of the algorithm are:

- bounding with the new bounds $\Theta(\alpha)$ (with prematured interruptions and warm restart);
- branching with a standard strategy using feasible solutions computed by greedy heuristics.

We compare this approach with the best one [**2**] that mixes nicely semidefinite programming with CPLEX. The numerical experiments on classical test-problems show that our results are comparable and sometimes better. The reason may be that the bound of [**2**] is of SDP-quality at the root of the branch-and-bound tree only, and deteriorates when getting down in the tree, while $\Theta(\alpha)$ is of SDP-quality all way long, without paying the price in computing time [**8**]. So we enumerate 10 times less nodes in average, and we are then able to solve densest subgraph problems of size 100 and more, in reasonnable cpu times. All this is precisely illustrated and commented in [**9**].

References

1. A. Billionnet. Different formulations for solving the heaviest k-subgraph problem. *Information Systems and Operational Res.*, 43(3):171–186, 2005.
2. A. Billionnet, S. Elloumi, and M.-C. Plateau. Improving the performance of standard solvers for quadratic 0-1 programs by a tight convex reformulation. *Discrete Applied Mathematics*, 157(6):1185–1197, 2009.
3. M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 6:1115–1145, 1995.
4. G. Jäger and A. Srivastav. Improved approximation algorithms for maximum graph partitioning problems. *Journal of Combinatorial Optimization*, 10(2):133–167, 2005.
5. S. Khot. Ruling out ptas for graph min-bisection, dense k-subgraph, and bipartite clique. *SIAM Journal on Computing*, 36:1025–1071, 2005.
6. J. Malick. A dual approach to semidefinite least-squares problems. *SIAM Journal on Matrix Analysis and Applications*, 26, Number 1:272–284, 2004.
7. J. Malick. Spherical constraint in Boolean quadratic programming. *Journal of Global Optimization*, 39(4), 2007.
8. J. Malick and F. Roupin. Numerical study of SDP bounds for the k-cluster problem. *to appear in E. Discrete Math.: Proceedings of ISCO 2010*, 2010.
9. J. Malick and F. Roupin. Solving k-cluster to optimality with semidefinite programming. *Submitted*, 2010.
10. F. Roupin. From linear to semidefinite programming: an algorithm to obtain semidefinite relaxations for bivalent quadratic problems. *Journal of Combinatorial Optimization*, 8(4), 2004.

IBBA: an Exact Global Optimization Software for the Design of Electromechanical Actuators

Frédéric Messine

ENSEEIHT-IRIT
2 rue Camichel
Toulouse, 31071, France

messine@n7.fr

Keywords: interval arithmetic, Branch-and-Bound, global optimization, mixed variables, categorical variable, design problem, electromechanical actuator.

The problem of the design of electromechanical actuators (such as electrical motors) is understood as an *inverse problem*: from some characteristical values we have to find the parameters of the studied actuator. We formulate this inverse problem as a *MINLP* one. Moreover, this MINLP problem is *non-homogeneous* and the discrete variables can be *integer* but also *boolean* or *categorical* ones. Indeed some variables of the corresponding models are continuous, such as for example the diameter or the length of an electrical motor, some other variables are integer ones, such as the number of magnets, and some can be of category such as the kind of material which will be used for the magnets.

IBBA (for Interval Branch and Bound Algorithm) is an exact global optimization algorithm based on *interval analysis* (for the computation of bounds) and *Branch and Bound techniques.* The particularity of IBBA is that it is able to solve some difficult MINLP of design of electromechanical actuators. Thus, during the collaboration with researchers of the LAPLACE Laboratory we extended IBBA to deal with:

- mixed integer problems including categorical variables,
- non-homogeneous problems,
- hard non-linear and non-convex constraints; yielding specific interval constraint propagation techniques,
- some methods of computation of the bounds,
- some constraints of black-box type: computation of some constraints using a finite element method.

Acknowledgments

This work was done in collaboration with the LAPLACE Laboratory of ENSEEIHT of Toulouse, France.

References

1. E. Fitan, F. Messine, B. Nogarède, *The Electromagnetical Actuators Design Problem: a General and Rational Approach*, IEEE Transaction on Magnetics, Vol. 40, N. 3, 2004.
2. E. Hansen, *Global Optimization Using Interval Analysis*, MARCEL DEKKER, Inc. 270 Madison Avenue, New York 10016, 1992.
3. R. B. Kearfott, *Rigorous Global Search: Continuous Problems*, Kluwer Academic Publishers, Dordrecht, Boston, London, 1996.
4. F. Messine, V. Monturet, B. Nogarède, *An Interval Branch and Bound Method Dedicated to the Optimal Design of Piezoelectric Actuators*, Mathematics and Computers in Sciences and Engineering, ISBN 960-8052-36-X, WSES Press, pp. 174–180, 2001.
5. F. Messine, *A Deterministic Global Optimization Algorithm for Design Problems.* To appear as chapter of the book "Essays and Surveys in Global Optimization", Kluwer Academic Publishers, 2005.
6. V. Monturet, B. Nogarède, Optimal dimensioning of a piezoelectric bimorph actuator, *European Physical Journal*, Vol. 17, pp. 107–118, 2002.
7. R. E. Moore, *Interval Analysis*, Prentice Hall, Inc. Englewood Cliffs, N.J., 1966.
8. H. Ratschek, J. Rokne, *New Computer Methods for Global optimization*, ELLIS HORWOOD LIMITED Market Cross House, Cooper Street, Chichester, West Sussex, PO19 1EB, England, 1988.

Strengthening of Lower Bounds in the Global Optimization of Bilinear Generalized Disjunctive Programs

Juan P. Ruiz Ignacio E. Grossmann

Department of Chemical Engineering
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213, USA

grossmann@cmu.edu

Generalized Disjunctive Programming (GDP), developed by Raman and Grossmann [**19**], has been proposed as a framework that facilitates the modeling of discrete-continuous optimization problems by allowing the use of algebraic and logical equations through disjunctions and logic propositions that are expressed in terms of Boolean and continuous variables. In order to take advantage of existing solvers [**25, 24, 20, 14, 11, 5**], GDPs are often reformulated as MILP/MINLP problems by using either the Big-M (BM) [**17**], or the Convex Hull (CH) [**12**] reformulation.

In the particular case of nonconvex GDP problems the direct application of traditional algorithms to solve the reformulated MINLPs such as Generalized Benders Decomposition (GBD) [**4, 8**] or Outer Approximation (OA) [**6**], may fail to find the global optimum since the Xsolution of the NLP subproblem may correspond to a local optimum and the cuts in the master problem may not be valid. Therefore, specialized algorithms must be used in order to find the global optimum [**9, 22, 7**].

Nonconvex GDP problems with bilinear constraints are of particular interest since these arise in many applications, for instance, in the design of pooling problems [**16**], in the synthesis of integrated water treatment networks [**10**], or generally, in the synthesis of process networks with multicomponent flows [**18**]. To tackle this problem, Lee and Grossmann [**13**] proposed a global optimization method that first relaxes the bilinear terms by using the convex envelopes of McCormick [**15**] and the concave terms by using linear under-estimators. The convex hull [**3**] is then applied to each disjunction. This formulation is then used within a spatial branch and bound technique in which the branching is first performed on the Boolean variables followed by the continuous variables. While the method proved to be effective in solving several problems, a major question is whether one might be able to obtain stronger lower bounds to enhance the efficiency for globally optimizing GDP problems.

The general structure of a nonconvex GDP can be represented as follows [**19, 23, 12**]:

$$
(GDP_{NC}) \quad
\begin{array}{ll}
\min\ Z = f(x) + \sum_{k \in K} c_k & \\
\text{s.t.} \quad g^l(x) \le 0 & l \in L \\
\bigvee_{i \in D_k} \begin{bmatrix} Y_{ik} \\ r^l_{ik}(x) \le 0 \ \ j \in J_{ik} \\ c_k = \gamma_{ik} \end{bmatrix} & k \in K \\
\Omega(Y) = \mathsf{true} & \\
x^{lo} \le x \le x^{up} & \\
x \in R^n,\ c_k \in R^1,\ Y_{ik} \in \{\mathsf{true}, \mathsf{false}\} &
\end{array}
$$

where $f : R^n \to R^1$ is a function of a continuous variables x in the objective function, $g^l : R^n \to R^1, l \in L$, belongs to the set of global constraints, the disjunctions $k \in K$, may be composed of a number of terms $i \in D_k$, that are connected by the OR operator. In each term there is a Boolean variable Y_{ik}, a set of inequalities $r^j_{ik}(x) \leqslant 0$, $r^j_{ik} : R^n \to R^1$, and a cost variable c_k. If Y_{ik} is true, then $r^j_{ik} \leqslant 0$ and $c_k = \gamma_{ik}$ are enforced; otherwise they are ignored. Also, $\Omega(Y) = \mathsf{true}$ are logic propositions for the Boolean variables. As indicated in Sawaya and Grossmann [**21**], we assume that the logic constraints $\underline{\vee}_{j \in J} Y_{ik}$ are contained in $\Omega(Y) = \mathsf{true}$. In a nonconvex GDP, f , r_{ik} and/or g^l are nonconvex functions.

Bilinear GDPs (BGDP) are the class of nonconvex GDP problems that we address in this paper. A BGDP is a nonconvex GDP where the functions in the constraints only contain bilinear and linear terms. In general we can represent a BGDP as:

$$
(GDP_B) \quad
\begin{array}{ll}
\min\ Z = d^T x + \sum_{k \in K} c_k & \\
\text{s.t.} \quad x^T Q^l x + a^l x \leqslant b^l & l \in L \\
\bigvee_{i \in D_k} \begin{bmatrix} Y_{ik} \\ x^T Q^l_{ik} x + a^l x \leqslant b^l_{ik} \ \ j \in J_{ik} \\ c_k = \gamma_{ik} \end{bmatrix} & k \in K \\
\Omega(Y) = \mathsf{true} & \\
x^{lo} \le x \le x^{up} & \\
x \in R^n,\ c_k \in R^1,\ Y_{ik} \in \{\mathsf{true}, \mathsf{false}\} &
\end{array}
$$

where some of the matrices Q^l , Q^j_{ik} are indefinite

In order to solve (GDPB) with a spatial branch and bound method, a convex GDP relaxation is required. A valid Linear GDP relaxation can be obtained by finding suitable under- and over-estimating functions of the nonconvex constraints.

Although this set of estimators is not unique, we propose to use the convex envelopes proposed by McCormick [**15**] for bilinear terms (see also Al-Khayyal and Falk [**1**]).

Defining $X = xx^T$ we can find a relaxation for each term $X_{ij} = x_i x_j$ as:

$$\begin{array}{ll}
X_{ij} \leqslant x_i xj^{up} + x_j x_i^{lo} - x_j^{up} x_i^{lo} & \\
X_{ij} \leqslant x_i xj^{lo} + x_j x_i^{up} - x_j^{lo} x_i^{up} & 1 \leq i < j < n+1 \\
X_{ij} \geqslant x_i xj^{lo} + x_j x_i^{lo} - x_j^{lo} x_i^{lo} & \\
X_{ij} \geqslant x_i xj^{up} + x_j x_i^{up} - x_j^{up} x_i^{up} &
\end{array}$$

This leads to the following Linear GDP,

$$(GDP_{RB}) \quad \begin{array}{ll}
\min \ Z = d^T x + \sum_{k \in K} c_k & \\
\text{s.t.} \quad Q^l \cdot X + a^l x \leqslant b^l & l \in L \\
\bigvee_{i \in D_k} \left[\begin{array}{c} Y_{ik} \\ Q^l_{ik} \cdot X + a^l x \leqslant b^l_{ik} \quad j \in J_{ik} \\ c_k = \gamma_{ik} \end{array} \right] & k \in K \\
\begin{array}{l}
X_{ij} \geqslant x_i xj^{up} + x_j x_i^{lo} - x_j^{up} x_i^{lo} \\
X_{ij} \geqslant x_i xj^{lo} + x_j x_i^{up} - x_j^{lo} x_i^{up} \\
X_{ij} \geqslant x_i xj^{lo} + x_j x_i^{lo} - x_j^{lo} x_i^{lo} \\
X_{ij} \geqslant x_i xj^{up} + x_j x_i^{up} - x_j^{up} x_i^{up}
\end{array} & 1 \leq i < j < n+1 \\
\Omega(Y) = \mathsf{true} & \\
x^{lo} \leq x \leq x^{up} & \\
x \in R^n, \ c_k \in R^1, \ Y_{ik} \in \{\mathsf{true}, \mathsf{false}\} &
\end{array}$$

where $\cdot$ represents the scalar product of matrices.

Traditionally, (GDPRB) has been used to predict lower bounds in the spatial branch and bound method [**13**]. In this work we will show that by the application of a systematic procedure we can improve the strength of the continuous relaxation of (GDPRB), leading to stronger lower bound predictions for (GDPB). In this work we first build on the work by Sawaya and Grossmann [**21**] exploiting the newly discovered hierarchy of relaxations in order to solve more efficiently bilinear GDP problems. Sawaya and Grossmann [**21**] have recently established new connections between Linear GDP and the Disjunctive Programming theory by Balas [**2**]. As a result, a family of tighter reformulations has been identified. These are obtained by performing a sequence of basic steps on the original disjunctive set (i.e. each basic step is characterized by generating a new set of disjunctions by intersecting the former), bringing it to a form closer to the Disjunctive Normal Form (DNF), and hence tightening its discrete relaxation [**3**]. It is important to note that each intersection usually creates new variables and constraints. Therefore, it is important to recognize when it may be useful to make these intersections. Some general rules are described in this work. The implementation of this framework is illustrated by finding a relaxation for a small example. We then outline the implementation of the tighter reformulation within a spatial branch and bound procedure whose performance is compared with current methodologies (i.e. Lee and Grossmann [**13**]) on a set of small process optimization problems.

Finally, we present a methodology to find tight convex relaxations for a special set of quadratic constraints given by bilinear and linear terms that frequently arise in the optimization of process networks. In general, process networks are composed of a set of nodes (N) connected by a set of streams (S). Each stream in S is associated with a flow F and a set of properties J whose values P^j are relative to F. The flows and properties of the set of streams leaving a node (O_n) are related to the flow and properties of the set of streams entering the node (I_n) and the characteristics of the node itself. The most frequent subset of equations found in these networks is represented by the following set of quadratic and linear constraints:

$$\textstyle\sum_{i\in I_n}(F_{in}P^j_{in}) - F_{on}P^j_{on} = 0 \quad \forall n \in N\,, \forall j \in J \tag{1}$$

$$\textstyle\sum_{i\in I_n} F_{in} - F_{on} = 0 \quad \forall n \in N \tag{2}$$

where F_{in} represents the flow of the entering stream $i \in I$ at a node $n \in N$ and P^j_{in} the value for the property j in that stream. Similarly, F_{on} and P^j_{on} are the characteristic values for the outlet stream. Without loss of generality we assume that only one stream leaves the node, namely, $|O_n| = 1$. The constraints in (1) and (2) typically correspond to mass or energy balances. Although very simple in its representation, this set of constraints define a nonconvex region, and when embedded they require global optimization techniques.

The basic idea in the proposed relaxation lies on exploiting the interaction between the vector spaces where the different set of variables are defined to tighten the relaxation of traditional approaches. In order to do so we first describe the system in vectorial form exposing the interaction of the different vector spaces. Second, we define and characterize the elementary building blocks of the system given by "minimal sets". This allows us to understand this interaction and generate from its properties cuts that are proved to be non-redundant for the original relaxation. Finally we assess the performance of the method by testing it in several case studies by embedding the resulting relaxation within a spatial branch and bound framework.

References

1. F.A. Al-Khayyal and J.E. Falk. Jointly constrained biconvex programming. *Mathematics of Operations Research*, 8(2):273–286, 1983.
2. E. Balas. Disjunctive programming. *Annals of Discrete Mathematics*, 5:3–51, 1979.
3. E. Balas. Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. *SIAM Journal on Algebraic and Discrete Methods*, 6:466–486, 1985.
4. J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.

5. L.T. Biegler A.R. Conn G. Cornuejols I.E. Grossmann C.D. Laird J. Lee A. Lodi F. Margot N. Sawaya A. Wchter Bonami, P. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5:186–204, 2008.
6. M. A. Duran and I. E. Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming*, 36:307, 1986.
7. C.A. Floudas. *Deterministic Global Optimization: Theory, Methods and Applications*. Kluwer Academic Publishers, 2000.
8. A.M. Geoffrion. Generalized benders decomposition. *Journal of Optimization*, 10(4):237–260, 1972.
9. R. Horst and H. Tuy. *Global Optimization deterministic approaches (3rd Ed).* Springer-Verlag, 1996.
10. R. Karuppiah and I. E. Grossmann. Global optimization for the synthesis of integrated water systems in chemical processes. *Computers and Chemical Engineering*, 20:650–673, 2006.
11. Allgor R. J. Gatzke E. P. Barton P. I. Kesavan, P. Outer approximation algorithms for separable nonconvex mixed-integer nonlinear programs. *Mathematical Programming*, 100(3):517–535, 2004.
12. S. Lee and Grossmann I. E. New algorithms for nonlinear generalized disjunctive programming. *Computers and Chemical Engineering*, 24:2125–2141, 2000.
13. S. Lee and Grossmann I. E. Global optimization of nonlinear generalized disjunctive programming with bilinear inequality constraints: application to process networks. *Computers and Chemical Engineering*, 27:1557–1575, 2003.
14. S. Leyffer. Integrating sqp and branch and bound for mixed integer nonlinear programming. *Computational Optimization and Applications*, 18:295–309, 2001.
15. G. P. McCormick. Computability of global solutions to factorable nonconvex programs. part i. convex underestimating problems. *Mathematical Programming*, 10:146–175, 1976.
16. C. Meyer and C.A. Floudas. Global optimization of a combinatorially complex generalized pooling problem. *AIChE Journal*, 52:1027–1037, 2006.
17. G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization.* Wiley, 1988.
18. I. Quesada and Grossmann I. E. Global optimization of bilinear process networks with multicomponent flows. *Computers and Chemical Engineering*, 19:1219–1242, 1995.
19. R. Raman and I. E. Grossmann. Modeling and computational techniques for logic based integer programming. *Computers and Chemical Engineering*, 18(7):563–578, 1994.
20. N. Sahinidis. Baron: A general purpose global optimization software package. *Journal of Global Optimization*, 8(2):201–205, 1996.
21. N. Sawaya and I. E. Grossmann. Reformulations, relaxations and cutting planes for linear generalized disjunctive programming. *Submitted for publication*, 2009.
22. M. Tawarmalani and N. Sahinidis. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming.* Kluwer Academic Publishers, 2002.

23. M. Turkay and I.E. Grossmann. Disjunctive programming techniques for the optimization of process systems with discontinuous investment costs-multiple size regions. *Industrial and Engineering Chemistry Research*, 35:2611–2623, 1996.
24. Viswanathan and I.E. Grossmann. A combined penalty function and outer-approximation method for minlp optimization. *Computers and Chemical Engineering*, 14(7):769–782, 1990.
25. T. Westerlund and F. Pettersson. An extended cutting plane method for solving convex minlp problems. *Computers and Chemical Engineering*, 19:131–136, 1995.

Improved SDP bounds for Quadratic Assignment Problem with suitable symmetry

Renata Sotirov

Tilburg University,
Warandelaan 2,
Tilburg, 5000 LE Tilburg, The Netherlands

r.sotirov@uvt.nl

Keywords: quadratic assignment problem, semidefinite programming, group symmetry.

We study the quadratic assignment problem (QAP) in the following form:

$$\min_{\pi \in \mathcal{S}_n} \sum_{i,j=1}^{n} a_{ij} b_{\pi(i),\pi(j)},$$

where $A = [a_{ij}]$ and $B = [b_{ij}]$ are given symmetric $n \times n$ matrices, and $\mathcal{S}_n$ is the symmetric group on n elements, *i.e.* the group of all permutations of $\{1, \ldots, n\}$. The matrices A and B are often called the flow and distance matrices respectively. The physical interpretation is that we are given n facilities with specified flows between facilities given by the matrix B, as well as n locations with relative distances between these locations given as the entries of A. The objective is to assign the facilities to locations such that the 'flow $\times$ distance' is minimal when summed over all pairs.

The QAP may be rewritten in terms of $n \times n$ permutation matrices as follows:

$$\min_{X \in \Pi_n} \operatorname{tr} AXBX^T$$

where Π_n is the set of $n \times n$ permutation matrices. It is well-known that the QAP contains the traveling salesman problem as a special case and is therefore NP-hard in the strong sense. Moreover, experience has shown that instances with $n = 30$ are already very hard to solve in practice. Thus it is typically necessary to use massive parallel computing to solve even moderately sized QAP instances; see [**1**]. The successful computational work in Anstreicher at al. employed convex relaxation of the QAP in a branch and bound setting. One class of convex relaxations that has been suggested for the QAP is via semidefinite programming, see [**5**].

Semidefinite programming (SDP) is a generalization of linear programming where the nonnegativity constraints are replaced by positive semidefiniteness on the matrix variables. Semidefinite programming studies show that it is a very promising method for providing tight relaxations for hard combinatorial problems, notably QAP. Derived SDP relaxations are often large scale and therefore hard to solve with the currently available solvers.

In particular, SDP relaxations of QAP introduced in [**5**] turn out to be quite good in practice, but computationally demanding for interior point solvers, even for relatively small instances (say $n \geq 15$). Lower order methods can solve the SDP relaxations for somewhat larger instances, but are known to be much slower than interior point methods.

For QAP instances where the data matrices have large *automorphism groups*, the SDP bounds can be computed more efficiently, as was shown by De Klerk and Sotirov [**2**], who computed the SDP bound of Zhao et al. [**5**] for some instances with n up to 128 with interior point solvers.

In this talk we show how one may obtain even stronger bounds for QAP instances where one of the data matrices has a *transitive automorphism* group, see [**3**]. Note that when we fix some entry in the permutation matrix X to one, we obtain a QAP problem that is one dimensional smaller than the original one. In terms of the physical interpretation of the QAP, we are assigning facility s to location r for a given index pair (r, s). In general, these bounds are not lower bounds for the original QAP problem, but if $\text{aut}(A)$ or $\text{aut}(B)$ is transitive, we do obtain such global lower bounds. We summarize this in the following lemma.

Lemma 0.1. [**3**] If $\text{aut}(A)$ or $\text{aut}(B)$ is transitive, then any lower bound for the QAP subproblem obtained by assigning facility s to location r is also a lower bound for the original QAP.

In the other words, every child node at the first level of the branching tree yields a lower bound on the global minimum of the QAP. Moreover, in [**3**] is proved that so obtained bounds dominate the SDP bound of Zhao et al. that were previously known as the strongest SDP bounds for the QAP. Further, we show that if one of the automorphism groups of the data matrices is transitive, say $\text{aut}(B)$, then the number of different subproblems in the first level of the branching tree depends on the number of orbits of $\text{aut}(A)$. This results with computing at most n different subproblems for the given data matrices A and B.

Our approach is very suitable for QAP instances with Hamming distance matrices. To illustrate our approach, we compute improved lower bounds for several test problems from the QAP library QAPLIB. In the table below we list the previous lower bounds, the new SDP lower bounds that we computed using symmetry reduction, the best know upper bounds, and computation times for the given problems.

It is clear from the table that improved lower bounds are obtained for all the instances in the table. The stronger bounds are obtained at a significant computational cost, though, as may be seen from the solution times listed in the table. The bounds for $esc32a$, $esc32b$ and $esc64a$ were computed by SDPA-DD solver[1], since these problems showed poor numerical conditioning. The high running times for these instances reflect the fact that SDPA-DD uses high precision computations.

[1]Available at `http://sdpa.indsys.chuo-u.ac.jp/sdpa/software.html`

instance	previous l.b.	new SDP l.b.	best known u.b.	time(s)
esc32a	104	107	130	191,510
esc32b	132	141	168	21,234
esc32c	616	618	642	256
esc32d	191	194	200	132
esc32h	425	427	438	1,313
esc64a	98	105	116	24,275

All other bounds were done with SeDuMi using the Yalmip interface on a Pentium IV 3.4 GHz dual-core processor.

The described approach has several potential areas of applications. The first aspect is that QAP problems with Hamming distance matrices arise in several applications: in the design of hardwired VLSI control units and in information theory with applications in channel coding. The second aspect of the presented approach is that we obtain a new SDP bound for all QAP instances where the automorphism group of one of the data matrices is transitive. One famous example of such an instance is the QAP reformulation of the traveling salesman problem (TSP). The new SDP relaxation of TSP dominates the previous best known SDP relaxation (see [**4**]) that is known to be independent of the Held-Karp bound. Another example of QAP instance where the automorphism group of one of the data matrices is transitive is the QAP reformulation of the equipartition problem (EQP). In this talk we also present some preliminary numerical results for the new TSP and EQP bounds.

References

1. K.M. Anstreicher, N. Brixius, J. Linderoth, and J.-P. Goux. Solving Large Quadratic Assignment Problems on Computational Grids. *Mathematical Programming, Series B*, 91:563–588, 2002.
2. E. de Klerk and R. Sotirov. Exploiting group symmetry in semidefinite programming relaxations of the quadratic assignment problem, *Mathematical Programming A*, 122(2):225–246, 2010.
3. E. de Klerk and R. Sotirov. Improved semidefinite programming bounds for quadratic assignment problems with suitable symmetry. Preprint, 2009.
4. E. de Klerk, D.V. Pasechnik and R. Sotirov. On semidefinite programming relaxations of the traveling salesman problem. *SIAM Journal of Optimization*, **19**(4):1559–1573, 2008.
5. Q. Zhao, S.E. Karisch, F. Rendl, and H. Wolkowicz. Semidefinite Programming Relaxations for the Quadratic Assignment Problem. *Journal of Combinatorial Optimization*, **2**, 71–109, 1998.

16

Polyhedrality and Inclusion Certificates in Convexification

Mohit Tawarmalani

Purdue University
100 S. Grant Street,
West Lafayette, IN 47907-2076, USA

mtawarma@purdue.edu

Abstract

Convex extensions have been found useful in constructing relaxations of nonlinear functions. In this paper, we use convex extensions to study inclusion certificates. Then, we use these certificates to gain insights into polyhedrality of multilinear sets. Finally, we discuss algorithmic techniques for generating facets for polyhedral sets occurring in nonlinear programs.

Keywords: convex extensions, envelopes, separation, multilinear.

1. Introduction

We are interested in solving nonlinear programs (NLP) described symbolically as:

$$\begin{array}{lll} (P) & \min & f(x) \\ & \text{s.t.} & g(x) \leq 0 \\ & & x \in \mathbb{R}^n \end{array}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$, $g : \mathbb{R}^n \mapsto \mathbb{R}^m$ are continuous functions.

The potential gains from global optimization of MINLPs has motivated a stream of recent research efforts in this direction. One of the successful deterministic techniques for solving NLPs is the branch-and-bound algorithm which has been implemented in various commercial and open-source software; [**1, 14, 10, 4**]. The branch and bound algorithm bounds (P) by solving convex relaxations over successively refined partitions; see [**9**]. The current relaxation techniques are derived from the factorable programming technique of [**11**]. They proceed by introducing new variables z_i, $i = 1, \ldots, k$ for sub-expressions of $f(x)$ and $g_j(x)$, $j = 1, \ldots, m$. Let $K = \{1, \ldots, k\}$ and for each $i \in K$, let $h_i(x)$ be the sub-expression that is replaced by z_i.

Rectangular partitioning in the most commonly used partitioning technique, which naturally yields bounds on variables at each node of the branch-and-bound tree. Then, for a given $I \subseteq K$, we are interested in relaxing $Z(I) = \{(z, x) \mid z_i = h_i(x) \forall i \in I, x \in \mathcal{H}\}$, where $\mathcal{H}$ is a hypercube obtained from the bounds on x at the specific partition in consideration. Most research has focused on the case where I is a singleton. In this case, it suffices to find the convex and concave envelopes of $h_i(x)$ over $\mathcal{H}$. There has been a considerable amount of research in identifying such envelopes for various classes of functions; see for example [**2, 13, 15, 19, 5, 12**]. For more details on relaxation techniques, the reader is referred to the recent surveys [**7, 8**]. Incidentally, since the envelopes approach $h_i(x)$ when the diameter of the underlying set approaches 0 (see [**3**]), they along with exhaustive partitioning techniques and appropriate node-selection rules satisfy the conditions needed to establish the asymptotic convergence of the branch-and-bound algorithm. Problems related to bounding probabilities of a finite set of events are shown to be related to computing the inclusion certificates.

2. Polyhedral Envelopes

Convex envelopes of various classes of functions have polyhedral properties. The polyhedral relaxations of the bilinear function used in factorable programming were shown to constitute the convex/concave envelope of the a bilinear term in [**2**]. In [**13**], the author showed that the convex/concave envelopes of multilinear functions are polyhedral. This has been exploited in [**15**] to develop closed-form expressions for convex/concave envelopes of the combinatorial multilinear functions over the unit hypercube. In [**19**], the authors developed techniques to identify the subset of the domain that determines the convex envelope of a lower-semicontinuous function. Using convex extensions, [**17, 18**] showed that the multilinear set is also polyhedral, *i.e.*, when each h_i is multilinear, the convex hull of $Z(I)$ can be obtained by restricting x to the extreme points of the underlying hypercube. In [**6**], the authors derive the the convex and concave envelope of $\sum_{i,j} a_{ij} x_i y_j$. In [**5**], the author developed a closed-form expression for the concave envelope of a monomial function over a hypercube in the non-negative orthant. The concave envelope of an almost-positive multilinear function is developed in [**12**]. In [**16**], the author discusses conditions under which the envelopes are polyhedral. In [**3**], the polyhedrality of the convex envelope of a multilinear function was exploited to develop a column-generation algorithm for identifying the facets of the convex/concave envelope of an arbitrary multilinear function. In [**21**], the authors identify various families of functions for which they derive closed-form expressions of convex/concave envelopes. They demonstrate that the results of [**2, 15, 5, 12**] are special cases of their constructions.

2.1. Inclusion Certificates

In this paper, we provide necessary and sufficient conditions for identifying generating sets in terms of Fenchel conjugates and apply them to the study of polyhedrality of nonlinear functions. We then focus on a disjunctive union of convex sets. We associate with each x in the convex hull of the disjunctive union an inclusion certificate that provides convex multipliers for each convex set in the union and certifies x as a member of the convex hull. Note that inclusion certificates are not unique.

However, given one functional representation of inclusion certificate, we show that other inclusion certificates are obtained via convexification. This convexification can be performed using disjunctive programming techniques. Then, we discuss how inclusion certificates for addition and Cartesian product of disjunctive unions can be obtained from the inclusion certificates of the constituent disjunctive unions. We specialize the results to the case when each disjunction is a single point. In this case, the inclusion certificates are the barycentric coordinates. The barycentric coordinates for a Cartesian product of simplices are well-known and give insights into the polyhedrality of multilinear sets. However, we show that for general polyhedra and bounded integer sets the barycentric coordinates cannot be polynomial functions. In [**20**], the authors have discovered rational functions that can be used as barycentric coordinates for arbitrary polyhedral sets.

2.2. Separation Techniques

We then focus on $Z(I)$ and develop algorithmic techniques to identify facets of $Z(I)$ that separate a given point (x, z) from $Z(I)$. Towards this end, we discuss general results regarding separation from convex sets. We develop a primal-dual pair of convex programs that have nice separation properties. In this construction, we exploit normalization strategies for cones. For example, we show that a pointed cone can be normalized by using a point in the relative interior of the polar cone. We also discuss other normalization strategies that use information about the point being separated. These are then exploited in our algorithm for separating points from $Z(I)$.

We specialize the discussion to polyhedral sets. For this case, the separation problems discussed above reduce to linear programming problems. We interpret the separation problems geometrically, gain insights, and derive various structural results. For example, we show that, under certain technical restrictions, the facets of $Z(I)$ correspond to the minimal faces of the primal linear program. When I is a singleton, the separation problem reduces to the problem discussed in [**15, 3, 21**]. This allows us to extend the results in the literature to the case when I has more than one element.

We show that the sub-problem associated with a specific Lagrangian relaxation reduces to the separation problem of [**3**]. Then, the decomposition algorithm of [**3**] can be exploited with a sub-gradient ascent scheme to identify the facet of $Z(I)$ that separates (x, z). Future work will focus on implementing the algorithm and identifying other classes of functions that are simultaneously polyhedral.

Acknowledgments

This research was supported in part by NSF Award DMI-0900065.

References

1. C. S. Adjiman, I. P. Androulakis, and C. A. Floudas. A global optimization method, αBB, for general twice-differentiable constrained NLPs–II. Implementation and computational results. *Computers & Chemical Engineering*, 22:1159–1179, 1998.
2. F. A. Al-Khayyal and J. E. Falk. Jointly constrained biconvex programming. *Mathematics of Operations Research*, 8:273–286, 1983.

3. X. Bao, N. V. Sahinidis, and M. Tawarmalani. Multiterm polyhedral relaxations for nonconvex, quadratically constrained quadratic programs. *Optimization Methods & Software*, 24:485–504, 2009.
4. P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Waechter. Branching and bounds tightening techniques for non-convex MINLP. Available at `http://www.optimization-online.org/DB\_HTML/2008/08/2059.html`, 2008.
5. H. P. Benson. On the construction of convex and concave envelope formulas for bilinear and fractional functions on quadrilaterals. *Computational Optimization and Applications*, 27:5–22, 2007.
6. D. Coppersmith, O. Günlük, J. Lee, and J. Leung. A Polytope for a Product of Real Linear Functions in 0/1 Variables. *IBM Research Report*, 2003.
7. C. A. Floudas and C. E. Gounaris. A review of recent advances in global optimization. *Journal of Global Optimization*, 45:3–38, 2009.
8. R. Hemmecke, M. Köppe, J. Lee, and R. Weismantel. Nonlinear integer programming. In *50 years of integer programming 1958–2008: The Early Years and State-of-the-Art Surveys*, pages 561–618. Springer-Verlag, 2010.
9. R. Horst and H. Tuy. *Global Optimization: Deterministic Approaches.* Springer Verlag, Berlin, Third edition, 1996.
10. LINDO Systems Inc. LINGO 11.0 optimization modeling software for linear, nonlinear, and integer programming. Available at `http://www.lindo.com`, 2008.
11. G. P. McCormick. Computability of global solutions to factorable nonconvex programs: Part I—Convex underestimating problems. *Mathematical Programming*, 10:147–175, 1976.
12. C. A. Meyer and C. A. Floudas. Convex envelopes of edge-concave functions. *Mathematical programming*, 103:207–224, 2005.
13. A. D. Rikun. A convex envelope formula for multilinear functions. *Journal of Global Optimization*, 10:425–437, 1997.
14. N. V. Sahinidis and M. Tawarmalani. *BARON.* The Optimization Firm, LLC, Urbana-Champaign, IL, 2005. Available at `http://www.gams.com/dd/docs/solvers/baron.pdf`.
15. H. D. Sherali. Convex envelopes of multilinear functions over a unit hypercube and over special discrete sets. *Acta Mathematica Vietnamica*, 22:245–270, 1997.
16. F. Tardella. Existence and sum decomposition of vertex polyhedral convex envelopes. *Optimization Letters*, 2:363–375, 2008.
17. M. Tawarmalani. Polyhedral basis, probability spaces, and links to disjunctive programming. Technical Report, Krannert School of Management, 2002.
18. M. Tawarmalani. Convexification and global optimization of nonlinear programs. Presented at Workshop on Integer Programming and Continuous Optimization, Chemnitz, 2004.
19. M. Tawarmalani and N. V. Sahinidis. Convex extensions and convex envelopes of l.s.c. functions. *Mathematical Programming*, 93:247–263, 2002.
20. J. Warren, S. Schaefer, A. N. Hirani, and M. Desbrun. Barycentric coordinates for convex sets. *Advances in computational mathematics*, 27:319–338, 2007.
21. C. Xiong, M. Tawarmalani, and J.-P. P. Richard. Deriving polyhedral envelopes through polyhedral subdivisions. working paper, 2010.

17

Two problems involving discrete variables: algorithm optimization in DFO and image reconstruction in acoustics

Philippe Toint[1] **Serge Gratton**[2] **M'Barek Fares**[3]

[1] Department of Mathematics, FUNDP-University of Namur
61, rue de Bruxelles, B-5000 Namur, Belgium
philippe.toint@fundp.ac.be

[2] ENSEEIHT-IRIT,
2, rue Camichel, 31000 Toulouse, France
serge.gratton@enseeiht.fr

[3] CERFACS,
42, avenue Coriolis, 31057 Toulouse Cedex 01, France
fares@cerfacs.fr

We present recent advances in two topics where the occurence of discrete variables plays a significant role. The first is the numerical optimization of the performance of algorithms. We consider here a "brute-force" algorithm for derivative-free minimization of mixed-integer nonlinear problems. After some description of this method and of the meaning of its parameters, we indicate how robust optimization has been used to tune these parameters to improve the performance of the minimizers on a set of test problems from the literature. The second problem of interest is the reconstruction of the shape of unknown objects from measurements of reflected plane waves (far-field measures). In this case, one is faced with the (approximate) reconstruction of an indicator function in the discretized 3D space. A new method (SVD-tail) is proposed in the framework of the linear sampling methods, for which image reconstruction is shown to be both extremely fast and visually satisfying.

18

An Interior-Point Algorithm for Nonlinear Optimization Using Iterative Linear Solvers

Andreas Wächter

IBM T.J. Watson Research Center
1101 Kitchawan Road
Yorktown Heights, NY 10598, USA

andreasw@us.ibm.com

Keywords interior point, inexact step computation.

In many large-scale applications, the computational time in a continuous optimization algorithm is dominated by the solution of large, sparse linear systems, required for the step computations. In some cases, particularly in the context of PDE-constrained optimization, direct factorization of the matrices in these systems creates a lot of fill-in, resulting in very expensive iterations. In this talk, we present an interior-point line-search algorithm that utilizes an iterative linear solver instead. Special attention is paid to the termination criteria of the iterative procedure to ensure that the resulting inexact steps still guarantee convergence of the overall optimization algorithm to a local solution under mild assumptions. The algorithm has been implemented in the Ipopt open-source optimization package, in conjunction with the Pardiso linear solver library. Numerical results for some PDE-constrained optimization problems will be presented.

This is joint work with Frank Curtis (Lehigh University) and Olaf Schenk (University of Basel).

Complexity of Nonlinear Discrete Parametric Optimization

Robert Weismantel

ETH
Rämistrasse 101
Zürich, 8032, Switzerland

weismant@imo.math.uni-magdeburg.de

Abstract

We consider the problem of optimizing a nonlinear objective function over a weighted independence system presented by a linear-optimization oracle. While this problem is generally intractable, we are able to provide approximate or exact polynomial-time algorithms for various special cases. In every such case one considers a apecialized combinatorial feasible domain. The cases do not only differ with respect to the schemes of encoding the input data, but also depend on different presentations of the oracle for the nonlinear objective function.
Keywords: parametric optimization, nonlinear discrete optimization.

1. Introduction

This talk focuses on nonlinear discrete optimization problems defined by a nonlinear function $f: \mathbb{R}^d \mapsto \mathbb{R}$, a matrix $W \in \mathbb{Z}^{d\times n}$ such that $f(Wx)$ is to be minimized over the feasible domain $\mathbf{F}$ described by rational data in binary encoding and given as

$$\mathbf{F} = \{x \in \mathbb{Z}^n,\ Ax = b,\ 0 \leqslant x \leqslant u\}. \tag{19.1}$$

In the special case when $d = 1$ we always use w in place of W.

The purpose of this talk is to study the computational complexity of the problem in different special cases regarding the structure of the feasible domain, the encoding of the weight matrix W and the oracle presenting the objective function.

We first argue that the presentation of the oracle affects the efficieny of algorithms for solving nonlinear discrete optimization problems. In order to demonstrate this fact we consider an algorithm that is allowed to perform arithmetic and logic operations on the elements of a Graver basis in order to be able to conclude whether or not a given feasible point x^0 is optimal for $\min f(w^T x)$ subject to $x \in \mathbf{F}$. It will be shown that different oracles for presenting f influence the efficiency of algorithms testing optimality. For example, if f is quasi convex and presented by a comparison oracle which, when queried on $x, y \in \mathbf{F}$, decides whether $f(x) < f(y)$, $f(x) = f(y)$, or $f(x) > f(y)$ holds, an efficient algorithm cannot exist.

2. Convex minimization over an N-fold System

We next consider the solution of our generic nonlinear discrete optimization for minimizing a convex univariate function via augmentation procedures. We show that a so-called greedy augmentation procedure that employs only directions from a Graver basis needs only polynomially many augmentation steps to solve the given problem. We extend these results to convex N-fold integer minimization problems and to convex 2-stage stochastic integer minimization problems. More precisely, we consider data $A \in \mathbb{Z}^{m\times n}$, $u \in \mathbb{Z}^n$, $b \in \mathbb{Z}^m$, $W \in \mathbb{Z}^{d\times n}$. Let $f : \mathbb{R}^d \mapsto \mathbb{R}$ denote a separable convex function presented by a comparison oracle which, when queried on $x, y \in \mathbb{Z}^{s+1}$, decides whether $f(x) < f(y)$, $f(x) = f(y)$, or $f(x) > f(y)$. Moreover, let H be an upper bound for the difference of maximum and minimum value of f over the feasible set $\mathbf{F} = \{x \in \mathbb{Z}^n \mid Ax = b, 0 \leqslant x \leqslant u\}$. Assuming that the encoding length of H is of polynomial size in the encoding lengths of A, u, b, W, any feasible solution z^0 to $\mathbf{F}$ can be augmented to an optimal solution by iteratively applying the following greedy procedure:

> Choose a greedy direction αg from a Graver basis $G(A, W)$ and set $z^0 := z^0 + \alpha g$.
> If $\alpha g = 0$, that is if $\alpha = 0$ for all $g \in G(A, W)$, return z_0 as optimal solution.

The number of augmentation steps in this augmentation procedure is polynomially bounded in the encoding lengths of A, u, b, W, and z^0.

As an application of this result we obtain

Theorem 19.1. *Let A and W be fixed and $f^1, \ldots, f^N : \mathbb{R}^n \mapsto \mathbb{R}$ be separable convex functions presented by a comparision oracle. The optimization problem*

$$\min \left\{ \sum_{i=1}^{N} f^{(i)}\left(x^{(i)}\right) : \sum_{i=1}^{N} x^{(i)} = b^{(0)}, Ax^{(i)} = b^{(i)}, 0 \leqslant x^{(i)} \leqslant u^{(i)}, x^{(i)} \in \mathbb{Z}^n, i = 1, \ldots, N \right\},$$

can be solved time that is polynomial in N [**2**].

3. Nonlinear Optimization over a Weighted Independence System

An *independence system* is a nonempty set of vectors $\mathbf{F} \subseteq \{0,1\}^n$ with the property that $x \in \{0,1\}^n$, $x \leqslant y \in \mathbf{F}$ implies $x \in \mathbf{F}$.

The computational complexity of the nonlinear optimization problem (19.1) over an independence system depends on the number d of weight vectors, on the weights w_j^i , on the type of function f and its presentation, and on the type of independence system $\mathbf{F}$ and its presentation. For example, when $\mathbf{F}$ is a *matroid*, the problem can be solved in polynomial time for any fixed d , any $\{0,1,\ldots,p\}$-valued weights w_j^i with p fixed, and any function f presented by a *comparison oracle*, even when $\mathbf{F}$ is presented by a mere *membership oracle*, see [**1**]. On the other hand, for *convex* f , already with fixed $d = 2$ and $\{0,1\}$-valued weights w_j^i , it includes as a special case the notorious *exact matching problem*, the complexity of which is long open [**4, 5**].

In view of the difficulty of the problem already for $d = 2$, we concentrate on the special case with $d = 1$, single weight vector $w = (w_1, \ldots, w_n) \in \mathbb{Z}^n$, and univariate function $f : \mathbb{Z} \to \mathbb{R}$. The function f can be arbitrary and is presented by a comparison oracle. The weights w_j take on values in a p-tuple $a = (a_1, \ldots, a_p)$ of positive integers. It turns out that solving this problem to optimality may require exponential time (see Theorem 19.3), and so we aim at an approximate solution in the following sense. For a nonnegative integer r , we say that $x^* \in \mathbf{F}$ is an *r-best solution* to the optimization problem over $\mathbf{F}$ if there are at most r better objective values attained by feasible solutions. In particular, a 0-best solution is optimal. Recall that the *Frobenius number* of a primitive a is the largest integer $\mathrm{Frob}(a)$ that is not expressible as a nonnegative integer combination of the a_i . We prove the following theorem.

Theorem 19.2. *For every primitive p-tuple $a = (a_1, \ldots, a_p)$, there is a constant $r(a)$ and an algorithm that, given any independence system $\mathbf{F} \subseteq \{0,1\}^n$ presented by a linear-optimization oracle, weight vector $w \in \{a_1, \ldots, a_p\}^n$, and function $f : \mathbb{Z} \to \mathbb{R}$ presented by a comparison oracle, provides an $r(a)$-best solution to the nonlinear problem $\min\{f(wx) \ : \ x \in \mathbf{F}\}$, in time polynomial in n . Moreover:*

(1) If a_i divides a_{i+1} for $i = 1, \ldots, p-1$, then the algorithm provides an optimal solution.

(2) For $p = 2$, that is, for $a = (a_1, a_2)$, the algorithm provide an $\mathrm{Frob}(a)$-best solution.[**3**]

In fact, we give an explicit upper bound on $r(a)$ in terms of the Frobenius numbers of certain subtuples derived from a .

Because $F(2,3) = 1$, Theorem 19.2 (Part 2) assures us that we can efficiently compute a 1-best solution in that case. It is natural to wonder then whether, in this case, an optimal (i.e., 0-best) solution can be calculated in polynomial time. We present a general result that indicates that this cannot be done.

Theorem 19.3. *There is no polynomial time algorithm for computing an optimal (i.e., 0-best) solution of the nonlinear optimization problem $\min\{f(wx) \ : \ x \in \mathbf{F}\}$ over an independence system presented by a linear optimization oracle with f presented by a comparison oracle and weight vector $w \in \{2,3\}^n$* [**3**].

Acknowledgments

This paper is based on joint work with Raymond Hemmecke, Jon Lee and Shmuel Onn.

References

1. Berstein, Y., Lee, J., Maruri-Aguilar, H., Onn, S., Riccomagno, E., Weismantel, R., Wynn, H.: Nonlinear matroid optimization and experimental design. *SIAM Journal on Discrete Mathematics* (to appear)
2. Hemmecke, R., Onn, S., Weismantel, R.: A polynomial oracle-time algorithm for convex integer minimization. *Mathematical Programming* (to appear)

3. Lee, J., Onn, S., Weismantel, R.: Approximate Nonlinear Optimization over Weighted Independence Systems. *SIAM Journal on Discrete Mathematics* 23:1667–1681 (2009)
4. Mulmuley, K., Vazirani, U.V., Vazirani, V.V.: Matching is as easy as matrix inversion. *Combinatorica* 7:105–113 (1987)
5. Papadimitriou, C.H., Yanakakis, M.: The complexity of restricted spanning tree problems. *Journal of the Association for Computing Machinery* 29:285–309 (1982)

20

Global Optimization of Signomial Programming Problems

Tapio Westerlund **Andreas Lundell**

Åbo Akademi University
Process Design and Systems Engineering Laboratory
COE in Optimization and Systems Engineering
Biskopsgatan 8
Turku, 20500, Finland

`http://www.abo.fi/ose`, `tapio.westerlund@abo.fi`

Abstract

In this presentation, an overview of a signomial global optimization algorithm is given. As the name indicates, the algorithm can be used to solve mixed integer nonlinear programming problems containing signomial functions to global optimality. The method employs single-variable power and exponential transformations for convexifying the nonconvex signomial functions termwise. By approximating the transformations using piecewise linear functions, piecewise convex underestimators for the nonconvex signomial functions as well as a relaxed convex problem can be obtained. In the algorithm, the approximations resulting from the piecewise linear functions are subsequentially improved resulting in a set of subproblems whose optimal solution converges to that of the original nonconvex problem. Finally, some recent theoretical results regarding the underestimation properties of the convexified signomial terms obtained using different transformations are also given.

Keywords: global optimization, signomial functions, transformation techniques, convex relaxations, convex underestimators.

1. Introduction

Optimization problems containing so-called signomial functions appear in many different application areas. Signomial functions are often highly nonlinear and nonconvex. Deterministic global optimization methods such as BARON [**7**], αBB [**1**], as well as methods utilizing strategies such as those given by Li et al [**2**] can be used to solve optimization problems of this kind. These types of optimization algorithms apply convex underestimators in partitions of the whole domain and subdomains resulting from using a branch-and-bound type method, while the signomial global optimization (SGO) algorithm uses piecewise convex underestimators defined over the entire domain instead.

Signomial programming (SP) and mixed integer signomial (MISP) problems are special classes of mixed integer nonlinear programming (MINLP) problems. The SGO algorithm is applicable to the following type of MISP problems:

$$\begin{aligned}
&\text{minimize} && f(\mathbf{x}),\\
&\text{subject to} && \mathbf{A}\mathbf{x} = \mathbf{a}, \quad \mathbf{B}\mathbf{x} \leqslant \mathbf{b},\\
& && \mathbf{g}(\mathbf{x}) \leqslant 0,\\
& && \mathbf{q}(\mathbf{x}) + \boldsymbol{\sigma}(\mathbf{x}) \leqslant 0.
\end{aligned} \tag{20.1}$$

In this formulation, the objective function f and the functions g and q occuring in the nonlinear constraints are assumed to be convex or pseudoconvex depending on the MINLP solver used to solve the subproblems. The vector of variables $\mathbf{x}$ can contain both real- and integer-valued variables, however, the variables in the signomial functions σ are assumed to be strictly positive.

A signomial function is defined as the sum of signomial terms, where each term consists of products of power functions $x_i^{p_i}$, *i.e.*

$$\sigma(\mathbf{x}) = \sum_{j=1}^{J} c_j \prod_{i=1}^{N} x_i^{p_{ji}}, \tag{20.2}$$

When the coefficients c_j are all positive, the signomial function is a posynomial. Optimization problems containing posynomial functions, so called geometric programming (GP) problems, can often be solved quite easily to global optimality, but this is not generally the case of SP and MISP problems.

In the methods used here, the convexity of the transformed problem is guaranteed by ensuring that each signomial term is convex. The convexity of signomial terms have been discussed previously, see *e.g.,* [**3, 5, 8, 6**], so here only a brief summary is given:

- A positive signomial term is convex if all powers p_i in the term are negative, or if one power p_k is positive, the rest p_i $i \neq k$ are negative, and the sum of these are greater than or equal to one, $i.e.\sum_i p_i \geqslant 1$.
- A negative signomial term is convex if all powers p_i are positive and the sum of these are between zero and one, $i.e. 0 < \sum_i p_i \leqslant 1$.

By applying different mappings based on the above cases of convexity, as well as the exponential mapping discussed in *e.g.,* [**3, 4**] the nonconvex signomial terms can be convexified.

2. Different types of transformations

For both negative and positive terms, single-variable power transformations (PTs) of the type $x_i = {X_{ji}}^{Q_{ji}}$ (where X_{ji} is the transformation variable corresponding to the transformation of the variable x_i in the j-th signomial term in the problem) can be used to transform nonconvex signomial terms to convex ones. However, the convexity conditions must be fulfilled, so additional restrictions on the transformation powers Q_{ji} need to be imposed. For positive terms, the first type of transformation is the positive power transformation (PPT) which corresponds to the case where exactly one power in the transformed term is positive and the sum of the powers is greater than or equal to one; the second type of transformation, the negative power transformation (NPT), is where all powers in the convexified term are negative. Additionally, as in GP, for positive terms the exponential transformation (ET) can be utilized, see [**3, 4**]. For negative terms, power transformations are used. Here the conditions are that all powers in the convexified term are positive and that the sum of the powers in the transformed term is between zero and one.

The transformation procedure for a generalized signomial constraint can be illustrated as follows:

$$q(\mathbf{x}) + \sigma(\mathbf{x}) \leqslant 0 \xrightarrow{(i)} q(\mathbf{x}) + \sigma^C(\mathbf{x}, \mathbf{X}) \leqslant 0 \xrightarrow{(ii)} q(\mathbf{x}) + \sigma^C(\mathbf{x}, \hat{\mathbf{X}}) \leqslant 0. \tag{20.3}$$

In equation (20.3), $\mathbf{X}$ represent the vector of transformation variables and $\hat{\mathbf{X}}$ their approximations obtained from piecewise linear functions (PLFs). In step *(i)* the nonconvex signomial terms are convexified using either of the applicable transformations mentioned above. Although the generalized signomial constraints are now convexified, the transformed problem is still nonconvex, since nonlinear equality constraints corresponding to the relation between the original and new transformation variables must be included. Hence, the nonconvexities have only been moved over from the terms to these relations. However, in step *(ii)*, the relations between the original and transformation variables are approximated using PLFs, and the nonconvexities are moved to the gridpoints of the PLFs. The transformed problem obtained is now convex and relaxed.

Thus, using the transformation technique, a convex relaxation of the nonconvex problem will be obtained. Furthermore, the solution to the transformed problem will be a valid lover bound of the solution to the original problem, and by improving the approximation by subsequentially including more breakpoints in the PLFs, the global optimal solution to the nonconvex problem can be found.

The different types of transformations, and also the powers in the PTs, can often be chosen in many different ways. Application of different sets of transformations often lead to convex relaxations of different tightness and combinatorial complexity. In this presentation, some recent theoretical results [**3, 5**] regarding the underestimating properties of the different transformations are given. Furthermore, since the transformation step is included in a global optimization algorithm, a method for automatically obtaining the transformations is needed: By formulating and solving a mixed integer linear programming (MILP) problem, an optimized set of transformations can be found. This set can be regarded as optimal with respect to certain strategy parameters, and for example the total number of transformations needed can be minimized (resulting in a combinatorically simpler transformed problem).

The MILP method has been discussed in several sources previously, see *e.g.,* [**3, 4**] while only an overview of these results are given in this presentation. In the MILP problem formulation, it is also possible to favor certain types of transformations. This is important, since it has been shown that different transformations have different underestimation properties. For example, the ET always gives a tighter convex underestimator than the NPT for positive signomial terms. Also, under some additional conditions, the same is true for the PPT, *i.e.*it gives a tighter convex underestimator than the NPT. In constrast, it has been proved that neither of the PPT and the ET gives a tighter convex underestimator in the whole original domain of the variables.

Acknowledgments

The financial support from the Academy of Finland is gratefully acknowledged.

References

1. C. S. Adjiman, S. Dallwig, C. A. Floudas, and A. Neumaier. A global optimization method, αBB, for general twice-differentiable constrained NLPs – I. Theoretical advances. *Computers and Chemical Engineering*, 22(9):1137–1158, 1998.
2. H.-L. Li, J.-F. Tsai, and C. A. Floudas. Convex underestimation for posynomial functions of positive variables. *Optimization Letters*, 2(3):333–340, 2007.
3. A. Lundell. *Transformation Techniques for Signomial Functions in Global Optimization.* PhD thesis, ?bo Akademi University, 2009.
4. A. Lundell, J. Westerlund, and T. Westerlund. Some transformation techniques with applications in global optimization. *Journal of Global Optimization*, 43(2):391–405, 2009.
5. A. Lundell and T. Westerlund. Convex underestimation strategies for signomial functions. *Optimization Methods and Software*, 24:505–522, 2009.
6. C. D. Maranas and C. A. Floudas. Finding all solutions of nonlinearly constrained systems of equations. *Journal of Global Optimization*, 7:143–182, 1995.
7. N. V. Sahinidis and M. Tawarmalani. *BARON 7.2.5: Global optimization of mixed-integer nonlinear programs, user's manual*, 2005.
8. T. Westerlund. Some transformation techniques in global optimization. In L. Liberti and N. Maculan, editors, *Global Optimization: From Theory to Implementation*, volume 84 of *Nonconvex Optimization and its Applications*, pages 47–74. Springer, 2005.

PART 2

Contributed Lectures

Collision avoidance for the ATM problem: A mixed 0–1 nonlinear model

Antonio Alonso-Ayuso **Laureano Escudero**
Javier Martin-Campo

Department of Statistics and Operations Research
Universidad Rey Juan Carlos
Móstoles (Madrid), Spain
{antonio.alonso,laureano.escudero,javier.martin.campo}@urjc.es

Abstract

A 0–1 nonlinear model for the Collision Avoidance in Air Traffic Management (ATM) problem is presented. The aim of this problem is deciding the best strategy for an arbitrary aircraft configurations (continuous velocity changes) such that all conflicts in the airspace are avoided. A conflict is the loss of the minimum safety distance that two aircrafts have to maintain in their flight plans. A 0–1 nonlinear optimization model based on geometric transformations is developed knowing the initial flight plan (coordinates, angles and velocities in each time period) and minimizing acceleration variations where aircrafts are forced to return to the original flight configuration when no aircrafts are in conflict. A linear approximation by using Taylor polynomials is developed to solve the problem in linear terms.
Keywords: collision avoidance, air traffic management, mixed 0–1 nonlinear programming.

1. Problem description

Aircraft conflict detection and resolution is currently attracting the interest of many air transportation service providers and is concerned with the following question: Given a set of airborne aircraft and their intended trajectories, what control strategy should be followed by the pilots and the air traffic service provider to prevent the aircraft from coming too close to each other?

Methods for maintaining separation between aircraft in the current airspace system have been built from a foundation of structured routes and evolved procedures. Humans are an essential element in this process due to their ability to integrate information and make judgments. However, because failures and operational errors can occur, automated systems have begun to appear both in the cockpit and on the ground to provide decision support and to serve as traffic conflict alerting systems. These systems use sensor data to predict conflicts between aircraft and alert humans to a conflict and may provide commands or guidance to resolve the conflict. Relatively simple conflict predictors have been a part of air traffic control automation for several years, and the traffic alert and collision avoidance system (TCAS) has been in place onboard domestic transport aircraft since the early 1990s. Together, these automated systems provide a safety net should normal procedures and controller and pilot actions fail to keep aircraft separated beyond established minimums.

Recently, interest has grown toward developing more advanced automation tools to detect traffic conflicts and assist in their resolution. These tools could make use of future technologies, such as a data link of current aircraft flight plan information, to enhance safety and enable new procedures to improve traffic flow efficiency.

With the growth of airspace congestion, there is an emerging need to implement these types of tools to assist the human operators in handling the expanding traffic loads and improve flow efficiency.

Different methods have been proposed by various researchers to address airborne conflict detection and resolution (CDR). These methods have been developed not only for aerospace, but also for ground vehicle, robotics, and maritime applications because the fundamental conflict avoidance issues are similar across transportation modes. A review of recent CDR research suggests that the current environment is one in which a given solution approach to the problem is proposed and exercised, typically through a set of constrained and simplified examples.

A model based in 0–1 nonlinear constrained model is developed by using the geometric and theoretical model ideas of the Velocity Changes problem (VC) presented in Pallottino et al. (2002) [**3**] and [**4**]. The VC model assumes instantaneous changes in velocity to avoid a conflict, but this assumption can not be applied in realistic instances, since aircrafts need time to reach the new velocity. In the new model, so-called VCTP (Velocity Changes with Time Periods), continuous velocity changes are proposed by using the properties of a rectilinear movement and uniformly accelerating movement. Another assumption in the VC model is that aircrafts fly along a straight line where an aircraft can make a turn maneuver. Our model tackles this problem considering the polygonal (in each time period) of the trajectory.

For this model it is supposed that the preliminary trajectories of F aircrafts are known and it can be extracted the aircraft configurations in T fixed time points. In these points the velocity and the position (abscissa and ordinate) of each aircraft in each point and the motion angles between two points are known. With these data we construct a new model in which it will be decided the optimal configuration changing the aircraft accelerations and avoiding all conflicts between the aircrafts. To change the aircraft accelerations the equations relating to the rectilinear movement and uniformly accelerating movement will be used.

The organization of the remainder of the note is as follows. Section 2 introduces the notation of the elements of the problem. Section 3 presents the mixed 0-1 nonlinear model. Section 4 gives the main ideas of the iterative procedure for problem solving. Finally, section 5 concludes.

2. Problem notation

The main notation for the elements in the model is presented:

Sets

- $\mathcal{F} = \{1, ..., F\}$**:** set of aircrafts in the airspace.
- $\mathcal{T} = \{0, ..., T\}$**:** set of time periods.

Parameters

- x_f^{*t}, y_f^{*t}**:** the initial configuration of position (abscissa and ordinate) for aircraft f in time period t, for $f \in \mathcal{F}$ and $t \in \mathcal{T}$.
- x_f^{*d}, y_f^{*d}**:** the departure position (abscissa and ordinate) for aircraft f, for $f \in \mathcal{F}$.
- x_f^{*r}, y_f^{*r}**:** the arrival position (abscissa and ordinate) for aircraft f, for $f \in \mathcal{F}$.
- v_f^{*t}**:** initial velocity configuration for aircraft f in the time period t, for $f \in \mathcal{F}$ and $t \in \mathcal{T}$.
- $\underline{v_f^t}, \overline{v_f^t}$**:** minimum and maximum velocities allowed for aircraft f in time period t, for $f \in \mathcal{F}$ and $t \in \mathcal{T}$.
- $\underline{a_f^t}, \overline{a_f^t}$**:** minimum and maximum acceleration allowed for aircraft f in time period t, for $f \in \mathcal{F}$ and $t \in \mathcal{T}$.
- m_f^{*t}**:** direction of motion in $(-\pi, \pi]$ for aircraft f in time period t, for $f \in \mathcal{F}$ and $t \in \mathcal{T}$.
- s**:** safety distance between aircrafts, usually, 2.5 nautical miles.
- $c_f^{a^+}, c_f^{a^-}$**:** cost for changing positive or negative unit in the acceleration variation for aircraft f, for $f \in \mathcal{F}$.
- I_t**:** length of the time period between t, i.e., time distance between $t-1$ and t, for $t \in \mathcal{T}$.
- d_f^{*t}**:** covered distance for aircraft f during time period t in the initial configuration, for $f \in \mathcal{F}$ and $t \in \mathcal{T}$.
- d_f^{tot}**:** Total length of the polygonal of the trajectory for aircraft f, for $f \in \mathcal{F}$.
- t_f^d, t_f^r**:** scheduled departure and arrival times for flight f, for $f \in \mathcal{F}$.
- e**:** distance bound for considering a pair of aircrafts.

Variables

- x_f^t, y_f^t**:** the position (abscissa and ordinate) of aircraft f in time period t, for $f \in \mathcal{F}$ and $t \in \mathcal{T}$.
- v_f^t**:** velocity for aircraft f in time period t, for $f \in \mathcal{F}$ and $t \in \mathcal{T}$.
- a_f^t**:** acceleration variation for aircraft f in time period t, for $f \in \mathcal{F}$ and $t \in \mathcal{T}$. This variable is real, and we divide it in two positive variables, say, a_f^{t+} and a_f^{t-}.
- a_f^{t+}, a_f^{t-}**:** positive and negative acceleration variations for aircraft f in time period t, for $f \in \mathcal{F}$ and $t \in \mathcal{T}$.
- d_f^t**:** covered distance for aircraft f during time period t, for $f \in \mathcal{F}$ and $t \in \mathcal{T}$.
- γ_{ft}^n**:** auxiliary 0-1 variables to model the case of early or delay arrival to the targeted time for aircraft f in time period t, for $n = 1, \ldots, 8$, for $f \in \mathcal{F}$ and $t \in \mathcal{T}$.
- $\varphi_{ijt}, \psi_{ijt}$**:** auxiliary 0-1 variables to model null denominators, for $i, j \in \mathcal{F}$ and $t \in \mathcal{T}$.
- δ_{ijt}^n**:** auxiliary 0-1 variables to model the either-or type of constraints, for $n = 1, \ldots, 8$, $i, j \in \mathcal{F}$ and $t \in \mathcal{T}$.

3. Mixed 0–1 nonlinear modelization

Next, the proposed model is presented:

$$\min \sum_{f\in\mathcal{F}}\sum_{t\in\mathcal{T}}(a_f^{t+}+a_f^{t-}) \tag{21.1}$$

$\forall f\in\mathcal{F}, t=t_f^d,\dots,t_f^r$:

$$\underline{v_f^t}\leqslant v_f^{t-1}+a_f^tI_t\leqslant\overline{v_f^t} \tag{21.2}$$

$$\underline{a_f^t}\leqslant a_f^t\leqslant\overline{a_f^t} \tag{21.3}$$

$$\big(v_f^{t_f^d}+\sum_{\ell=t_f^d}^{t-1}a_f^\ell I_\ell\big)I_t+\frac{1}{2}a_f^tI_t^2=d_f^t \tag{21.4}$$

$$\sum_{\ell=t_f^d}^{t-1}d_f^{*\ell}-\sum_{\ell=t_f^d}^{t-1}d_f^\ell\leqslant\overline{M}(1-\gamma_{ft}^1) \tag{3.5a}$$

$$\sum_{\ell=t_f^d}^{t}d_f^{*\ell}-\sum_{\ell=t_f^d}^{t}d_f^\ell\leqslant\overline{M}(1-\gamma_{ft}^1) \tag{3.5b}$$

$$\begin{aligned}&x_f^t-\Big(x_f^{t-1}+d_f^t\cos(m_f^{*t-1})-x_f^{*t}\Big)\cos(m_f^{*t-1}-m_f^{*t})-\\&\Big(y_f^{t-1}+d_f^t\sin(m_f^{*t-1})-y_f^{*t}\Big)\sin(m_f^{*t-1}-m_f^{*t})-x_f^{*t}\leqslant\overline{M}(1-\gamma_{ft}^1)\end{aligned} \tag{3.5c}$$

$$\begin{aligned}&x_f^t-\Big(x_f^{t-1}+d_f^t\cos(m_f^{*t-1})-x_f^{*t}\Big)\cos(m_f^{*t-1}-m_f^{*t})-\\&\Big(y_f^{t-1}+d_f^t\sin(m_f^{*t-1})-y_f^{*t}\Big)\sin(m_f^{*t-1}-m_f^{*t})-x_f^{*t}\geqslant\underline{M}(1-\gamma_{ft}^1)\end{aligned} \tag{3.5d}$$

$$\begin{aligned}&y_f^t+\Big(x_f^{t-1}+d_f^t\cos(m_f^{*t-1})-x_f^{*t}\Big)\sin(m_f^{*t-1}-m_f^{*t})-\\&\Big(y_f^{t-1}+d_f^t\sin(m_f^{*t-1})-y_f^{*t}\Big)\cos(m_f^{*t-1}-m_f^{*t})-y_f^{*t}\leqslant\overline{M}(1-\gamma_{ft}^1)\end{aligned} \tag{3.5e}$$

$$\begin{aligned}&y_f^t+\Big(x_f^{t-1}+d_f^t\cos(m_f^{*t-1})-x_f^{*t}\Big)\sin(m_f^{*t-1}-m_f^{*t})-\\&\Big(y_f^{t-1}+d_f^t\sin(m_f^{*t-1})-y_f^{*t}\Big)\cos(m_f^{*t-1}-m_f^{*t})-y_f^{*t}\geqslant\underline{M}(1-\gamma_{ft}^1)\end{aligned} \tag{3.5f}$$

$$\cdots$$

$$\gamma_{f,t}^1+\gamma_{f,t}^2+\gamma_{f,t}^3+\gamma_{f,t}^4=1 \tag{21.6}$$

$\forall i<j\in\mathcal{F}, t=\max\{t_i^d,t_j^d\},\dots,\min\{t_i^r,t_j^r\}$:

$$v_i^t\cos(m_i^t)-v_j^t\cos(m_j^t)\leqslant\overline{M}(1-\delta_{ijt}^1) \tag{3.7a}$$

$$-v_i^th_i^t+v_j^th_j^t\leqslant\overline{M}(1-\delta_{ijt}^1) \tag{3.7b}$$

$$v_i^t\cos(m_i^t)-v_j^t\cos(m_j^t)\leqslant\overline{M}(1-\delta_{ijt}^2) \tag{3.7c}$$

$$v_i^tk_i^t-v_j^tk_j^t\leqslant\overline{M}(1-\delta_{ijt}^2) \tag{3.7d}$$

$$-v_i^t\cos(m_i^t)+v_j^t\cos(m_j^t)\leqslant\overline{M}(1-\delta_{ijt}^3) \tag{3.7e}$$

$$v_i^th_i^t-v_j^th_j^t\leqslant\overline{M}(1-\delta_{ijt}^3) \tag{3.7f}$$

(3.7g) $$-v_i^t \cos(m_i^t) + v_j^t \cos(m_j^t) \leqslant \overline{M}(1-\delta_{ijt}^4)$$

(3.7h) $$-v_i^t k_i^t + v_j^t k_j^t \leqslant \overline{M}(1-\delta_{ijt}^4)$$

(3.7i) $$-v_i^t \sin(m_i^t) + v_j^t \sin(m_j^t) \leqslant \overline{M}(1-\delta_{ijt}^5)$$

(3.7j) $$-v_i^t h_i^{\prime t} + v_j^t h_j^{\prime t} \leqslant \overline{M}(1-\delta_{ijt}^5)$$

(3.7k) $$-v_i^t \sin(m_i^t) + v_j^t \sin(m_j^t) \leqslant \overline{M}(1-\delta_{ijt}^6)$$

(3.7l) $$v_i^t k_i^{\prime t} - v_j^t k_j^{\prime t} \leqslant \overline{M}(1-\delta_{ijt}^6)$$

(3.7m) $$v_i^t \sin(m_i^t) - v_j^t \sin(m_j^t) \leqslant \overline{M}(1-\delta_{ijt}^7)$$

(3.7n) $$v_i^t h_i^{\prime t} - v_j^t h_j^{\prime t} \leqslant \overline{M}(1-\delta_{ijt}^7)$$

(3.7o) $$v_i^t \sin(m_i^t) - v_j^t \sin(m_j^t) \leqslant \overline{M}(1-\delta_{ijt}^8)$$

(3.7p) $$-v_i^t k_i^{\prime t} + v_j^t k_j^{\prime t} \leqslant \overline{M}(1-\delta_{ijt}^8)$$

(3.7q) $$\sum_{n=1}^{8} \delta_{ijt}^n = 1 - p_{ij}^t$$

(3.8a) $$x_i^t - x_j^t - s \leqslant \overline{M}(1-\varphi_{ijt})$$

(3.8b) $$x_j^t - x_i^t - s \leqslant \overline{M}(1-\varphi_{ijt})$$

(3.8c) $$\sum_{n=1}^{4} \delta_{ijt}^n = 1 - \varphi_{ijt}$$

(3.9a) $$y_i^t - y_j^t - s \leqslant \overline{M}(1-\psi_{ijt})$$

(3.9b) $$y_j^t - y_i^t - s \leqslant \overline{M}(1-\psi_{ijt})$$

(3.9c) $$\sum_{n=5}^{8} \delta_{ijt}^n = 1 - \psi_{ijt}$$

$\forall f \in \mathcal{F}$:

(3.10) $$\sum_{i=t_f^d}^{t_f^r} d_f^i = d_f^{tot}$$

$\forall f \in \mathcal{F}, \forall t \in \mathcal{T}$

(3.11a) $$x_f^t, y_f^t, v_f^t, a_f^{t+}, a_f^{t-}, d_f^t \in \mathbb{R}^+$$

(3.11b) $$\gamma_{ft}^n \in \{0,1\} \qquad \forall n = 1, \dots, 4$$

$\forall i < j \in \mathcal{F}, \forall t \in \mathcal{T}$

(3.11c) $$\delta_{ijt}^n, \varphi_{ijt}, \psi_{ijt} \in \{0,1\} \qquad \forall n = 1, \dots, 8$$

where $\overline{M}$ and $\underline{M}$ are upper and lower bounds of the constraints. These bounds are important to the model because they can affect, on one hand, the model feasibility and, in the other hand, the computing time. The objective function (21.1) minimizes the acceleration variations in absolute value considering $|a_f^t| = a_f^{t+} + a_f^{t-}$. (21.2) and (21.3) are the bounds of the velocities and accelerations respectively in each time period. (21.4) is the covered route in each time period t. The set of constraints (3.5) projects the new aircrafts positions in the polygonal of the trajectory in case of a nonlinear one. When a acceleration maneuver is made by an aircraft, depending on the acceleration sign, the aircraft can arrive at the next way-point at an earlier or later time. Four possible cases can occur (it is presented the first one, when an aircraft arrives earlier in times period $t-1$ and t) and they are modeled by using the γ variable. Constraint (3.6) decides one of the four cases that are contemplated. The complete model can be seen in [**2**]. The set of constraints (3.7) detects if a conflict occurs between aircrafts i and j, see [**1, 3, 4**]. Constraints (3.7b),(3.7d),(3.7f),(3.7h),(3.7j),(3.7l),(3.7n) and (3.7p) are nonlinear since the terms (presented below) are nonlinear functions:

$$h_i^t = \frac{(x_i^t - x_j^t)s + (y_i^t - y_j^t)\sqrt{(x_i^t - x_j^t)^2 + (y_i^t - y_j^t)^2 - s^2}}{(x_i^t - x_j^t)\sqrt{(x_i^t - x_j^t)^2 + (y_i^t - y_j^t)^2 - s^2} - (y_i^t - y_j^t)s} \cos(m_i^t) - \sin(m_i^t)$$

$$h_j^t = \frac{(x_i^t - x_j^t)s + (y_i^t - y_j^t)\sqrt{(x_i^t - x_j^t)^2 + (y_i^t - y_j^t)^2 - s^2}}{(x_i^t - x_j^t)\sqrt{(x_i^t - x_j^t)^2 + (y_i^t - y_j^t)^2 - s^2} - (y_i^t - y_j^t)s} \cos(m_j^t) - \sin(m_j^t)$$

$$k_i^t = \frac{-(x_i^t - x_j^t)s + (y_i^t - y_j^t)\sqrt{(x_i^t - x_j^t)^2 + (y_i^t - y_j^t)^2 - s^2}}{(x_i^t - x_j^t)\sqrt{(x_i^t - x_j^t)^2 + (y_i^t - y_j^t)^2 - s^2} + (y_i^t - y_j^t)s} \cos(m_i^t) - \sin(m_i^t)$$

$$k_j^t = \frac{-(x_i^t - x_j^t)s + (y_i^t - y_j^t)\sqrt{(x_i^t - x_j^t)^2 + (y_i^t - y_j^t)^2 - s^2}}{(x_i^t - x_j^t)\sqrt{(x_i^t - x_j^t)^2 + (y_i^t - y_j^t)^2 - s^2} + (y_i^t - y_j^t)s} \cos(m_j^t) - \sin(m_j^t)$$

$$h_i^{'t} = \frac{(x_i^t - x_j^t)\sqrt{(x_i^t - x_j^t)^2 + (y_i^t - y_j^t)^2 - s^2} - (y_i^t - y_j^t)s}{(x_i^t - x_j^t)s + (y_i^t - y_j^t)\sqrt{(x_i^t - x_j^t)^2 + (y_i^t - y_j^t)^2 - s^2}} \sin(m_i^t) - \cos(m_i^t)$$

$$h_j^{'t} = \frac{(x_i^t - x_j^t)\sqrt{(x_i^t - x_j^t)^2 + (y_i^t - y_j^t)^2 - s^2} - (y_i^t - y_j^t)s}{(x_i^t - x_j^t)s + (y_i^t - y_j^t)\sqrt{(x_i^t - x_j^t)^2 + (y_i^t - y_j^t)^2 - s^2}} \sin(m_j^t) - \cos(m_j^t)$$

$$k_i^{'t} = \frac{(x_i^t - x_j^t)\sqrt{(x_i^t - x_j^t)^2 + (y_i^t - y_j^t)^2 - s^2} + (y_i^t - y_j^t)s}{-(x_i^t - x_j^t)s + (y_i^t - y_j^t)\sqrt{(x_i^t - x_j^t)^2 + (y_i^t - y_j^t)^2 - s^2}} \sin(m_i^t) - \cos(m_i^t)$$

$$k_j^{'t} = \frac{(x_i^t - x_j^t)\sqrt{(x_i^t - x_j^t)^2 + (y_i^t - y_j^t)^2 - s^2} + (y_i^t - y_j^t)s}{-(x_i^t - x_j^t)s + (y_i^t - y_j^t)\sqrt{(x_i^t - x_j^t)^2 + (y_i^t - y_j^t)^2 - s^2}} \sin(m_j^t) - \cos(m_j^t)$$

The sets of constraints (3.8) and (3.9) avoid unstable situations caused by a null denominator in the above terms. Constraint (3.10) forces aircrafts to follow the scheduled path. Finally, (3.11) is the set of constraints that represents the type of variables in the model.

4. Algorithmic approach

This model is approximated to a linear model by using Taylor polynomials, obtaining the next mathematical expression for each nonlinear constraint:

$$\left\{ C_n + \begin{pmatrix} \frac{\partial C_n}{\partial v_i^t} & \frac{\partial C_n}{\partial v_j^t} & \frac{\partial C_n}{\partial x_i^t} & \frac{\partial C_n}{\partial x_j^t} & \frac{\partial C_n}{\partial y_i^t} & \frac{\partial C_n}{\partial y_j^t} \end{pmatrix} \right\}_{|(v_i^{*t}, v_j^{*t}, x_i^{*t}, x_j^{*t}, y_i^{*t}, y_j^{*t})} \begin{pmatrix} v_i - v_i^{*t} \\ v_j - v_j^{*t} \\ x_i - x_i^{*t} \\ x_j - x_j^{*t} \\ y_i - y_i^{*t} \\ y_j - y_j^{*t} \end{pmatrix}$$

$$[\quad \leqslant M(1 - \delta_{ijt}^n)$$

The algorithm for the resolution consists on solving the linearized model and updating the model parameters with the obtained solution until the difference between one solution and the last updated parameters is less than the fixed tolerance chosen. The algorithm is now presented:

Step 1. Compute the nonlinear constraint values such that $v_i^t, v_j^t, x_i^t, x_j^t, y_i^t, y_j^t$ are replaced by $v_i^{*t}, v_j^{*t}, x_i^{*t}, x_j^{*t}, y_i^{*t}, y_j^{*t}$ respectively.

Step 2. Solve the mixed zero-one model, where the nonlinear constraints are substituted by its linear approximation. Let v_f^t, x_f^t, y_f^t be the optimal values of the respective variables.

Step 3. Optimality test. If condition (21.7) is satisfied then stop, the quasi-optimal solution has been obtained. Otherwise, go to Step 4.

$$\sum_{f\in\mathcal{F}}\sum_{t\in\mathcal{T}}(v_f^t - v_f^{*t})^2 \leqslant \varepsilon \tag{21.7a}$$

$$\sum_{f\in\mathcal{F}}\sum_{t\in\mathcal{T}}(x_f^t - x_f^{*t})^2 \leqslant \varepsilon \tag{21.7b}$$

$$\sum_{f\in\mathcal{F}}\sum_{t\in\mathcal{T}}(y_f^t - y_f^{*t})^2 \leqslant \varepsilon \tag{21.7c}$$

Step 4. Update:

$$v_f^{*t} := v_f^{*t} + \lambda(v_f^t - v_f^*) \qquad \forall f \in \mathcal{F}, \forall t \in \mathcal{T} \tag{21.8a}$$

$$x_f^{*t} := x_f^{*t} + \lambda(x_f^t - x_f^*) \qquad \forall f \in \mathcal{F}, \forall t \in \mathcal{T} \tag{21.8b}$$

$$y_f^{*t} := y_f^{*t} + \lambda(y_f^t - y_f^*) \qquad \forall f \in \mathcal{F}, \forall t \in \mathcal{T} \tag{21.8c}$$

where $0 < \lambda < 1$, and go to Step 1.

5. Conclusions

A mixed 0-1 nonlinear model as well its algorithmic approach for problem solving have been presented. The latter uses a Taylor expansion approximation by iteratively solving mixed 0-1 linear models. The current computational experience is very promising but more computational experience is needed to assess the performance of the proposed approach for (large-scale) realistic instances. As a future we are planning to test our approach against available software for directly solving mixed integer nonlinear problems.

Acknowledgments

This research was partially supported by the grants OPTIMOS MTM2009-14039-C06-03 and PLANIN MTM2009-14087-C04-01 partially funded by the Spanish Ministry of Science and Innovation, as well as the project ATLANTIDA for GMV Aerospace and Defence, Spain.

References

1. A. Alonso-Ayuso, L.F. Escudero, F.J. Martin-Campo. Collision avoidance in the ATM problem: A mixed integer linear optimization approach. *Submitted for publication* (2009).
2. A. Alonso-Ayuso, L.F. Escudero, F.J. Martin-Campo. Collision Avoidance for the ATM problem: An extended presentation of a mixed 0-1 nonlinear model approach. *Technical report TR09/05, Dept. of Statistics and Operations Research, Rey Juan Carlos University, Móstoles (Madrid), Spain.* (2009).
3. L. Pallottino. Aircraft Conflict Resolution in “FREE FLIGHT” Air Traffic Management Systems: Models and Optimal Solutions *PhD thesis, Automation and Industrial Robotics, Universit di Pisa.*, (2002).
4. L. Pallottino, E. Feron and A. Bicchi. Conflict resolution problems for air fraffic management systems solved with mixed integer programming. *IEEE Transactions on Intelligent Transportation Systems.* 3(11). 3–11 (2002).

Undercover – a primal heuristic for MINLP based on sub-MIPs generated by set covering

Timo Berthold **Ambros M. Gleixner**

Zuse Institute Berlin
Takustraße 7, 14195 Berlin, Germany
`{berthold,gleixner}@zib.de`

Abstract

We present Undercover, a primal heuristic for mixed-integer nonlinear programming (MINLP). The heuristic constructs a mixed-integer *linear* subproblem (sub-MIP) of a given MINLP by fixing a subset of the variables. We solve a set covering problem to identify a minimal set of variables which need to be fixed in order to linearise each constraint. Subsequently, these variables are fixed to approximate values, e.g. obtained from a linear outer approximation. The resulting sub-MIP is solved by a mixed-integer linear programming solver. Each feasible solution of the sub-MIP corresponds to a feasible solution of the original problem. Although general in nature, the heuristic seems most promising for mixed-integer quadratically constrained programmes (MIQCPs). We present computational results on a general test set of MIQCPs selected from the MINLPLib [**12**].

Keywords: MINLP, MIQCP, primal heuristic, large neighbourhood search, set covering.

1. Introduction

For mixed-integer programming (MIP) it is well-known that, apart from complete solving methods, general-purpose primal heuristics like the feasibility pump [**2, 14, 16**] are able to find high-quality solutions for a wide range of problems. Over the years, primal heuristics have become a substantial ingredient of state-of-the-art MIP solvers [**5, 8**]. For mixed-integer nonlinear programming (MINLP) there have only been a few publications on general-purpose primal heuristics [**7, 10, 20, 11**].

At the heart of many recently proposed primal MIP heuristics, such as Local Branching [**15**], RINS [**13**], DINS [**17**], and RENS [**6**], lies large neighbourhood search, hence the paradigm of solving a small sub-MIP which promises to contain good solutions. In this paper, we introduce *Undercover*, a large neighbourhood search start heuristic that constructs and solves a sub-MIP of a given MINLP. We demonstrate its effectiveness on a general test set of mixed-integer quadratically constrained programmes (MIQCPs) taken from the MINLPLib [**12**].

An MINLP is an optimisation problem of the form

$$\begin{aligned} \min \quad & d^\top x \\ \text{s.t.} \quad & g_i(x) \leqslant 0 && \text{for } i = 1, \dots, m, \\ & L_k \leqslant x_k \leqslant U_k && \text{for } k = 1, \dots, n, \\ & x_k \in \mathbb{Z} && \text{for } k \in \mathcal{I}, \end{aligned} \tag{22.1}$$

where $\mathcal{I} \subseteq \{1, \dots, n\}$ is the index set of the integer variables, $d \in \mathbb{R}^n$, $g_i : \mathbb{R}^n \to \mathbb{R}$ for $i = 1, \dots, m$, and $L \in (\mathbb{R} \cup \{-\infty\})^n$, $U \in (\mathbb{R} \cup \{+\infty\})^n$ are lower and upper bounds on the variables, respectively. Note that a nonlinear objective function can always be reformulated by introducing one additional constraint, hence form (22.1) is general.

2. A generic algorithm

The paradigm of fixing a subset of the variables of a given mixed-integer programme in order to obtain subproblems which are easier to solve has proven successful in many primal MIP heuristics such as RINS [**13**], DINS [**17**], and RENS [**6**]. The core difficulty in MIP solving are the integrality constraints, thus in MIP context "easy to solve" usually takes the meaning of few integer variables. While in MINLP integralities do contribute to the complexity of the problem, a specific difficulty are the nonlinearities. Hence, "easy" in MINLP can be understood as few nonlinear constraints.

Our heuristic is based on the simple observation that by fixing certain variables (to some value within their bounds) any given mixed-integer *nonlinear* programme can be reduced to a mixed-integer *linear* subproblem (sub-MIP). Every feasible solution of this sub-MIP is then a feasible solution of the original MINLP.

Whereas in general it holds that many or even all of the variables might need to be fixed in order to arrive at a linear subproblem, our approach is motivated by the experience that for several practically relevant MIQCPs fixing only a comparatively small subset of the variables already suffices to linearise the problem. The computational effort, however, is usually greatly reduced since we can apply the full strength of state-of-the-art MIP solving to the subproblem. Before formulating a first generic algorithm for our heuristic, consider the following definitions.

Definition 22.1 (cover of a function)**.** Let a function $g : D \to \mathbb{R}$, $x \mapsto g(x)$ on a domain $D \subseteq \mathbb{R}^n$ and a point $x^\star \in D$ be given. We call a set $\mathcal{C} \subseteq \{1, \dots, n\}$ of variable indices an *$x^\star$-cover* of g if and only if the set

$$\{(x, g(x)) \mid x \in D, x_k = x_k^\star \text{ for all } k \in \mathcal{C}\} \tag{22.2}$$

is affine. We call $\mathcal{C}$ a *(global) cover* of g if and only if $\mathcal{C}$ is an $x^\star$-cover of g for all $x^\star \in D$.

Definition 22.2 (cover of an MINLP)**.** Let P be an MINLP of form (22.1), let $x^\star \in [L, U]$, and $\mathcal{C} \subseteq \{1, \dots, n\}$ be a set of variable indices of P. We call $\mathcal{C}$ an *$x^\star$-cover* of P if and only if $\mathcal{C}$ is an $x^\star$-cover for $g_1, \dots, g_m$. We call $\mathcal{C}$ a *(global) cover* of P if and only if $\mathcal{C}$ is an $x^\star$-cover of P for all $x^\star \in [L, U]$.

A first generic algorithm for our heuristic is given in Figure 1. The hinge of the algorithm is clearly found in Line 5: finding a suitable cover of the given MINLP. Section 3 elaborates on this in detail with special emphasis on the case of MIQCPs.

Figure 1. Generic algorithm

```
1 Input: MINLP $P$ as in (22.1)
2 begin
3     compute a solution $x^\star$ of an approximation or
      relaxation of $P$;
4     round $x^\star_k$ for $k \in \mathcal{I}$;
5     determine an $x^\star$-cover $\mathcal{C}$ of $P$;
6     solve the sub-MIP of $P$ given by $x_k = x^\star_k$, $k \in \mathcal{C}$ ;
7 end
```

To obtain suitable fixing values for the selected variables, an approximation or relaxation of the original MINLP is used. For integer variables the approximate values are rounded. Most exact solvers for MINLP are based on branch-and-bound. If the heuristic is embedded within a branch-and-bound solver, using its (linear or nonlinear) relaxation appears as a natural choice for obtaining approximate variable values.

Large neighbourhood search heuristics which rely on fixing variables typically have to trade off between eliminating many variables in order to make the sub-MIP tractable versus leaving enough degrees of freedom such that the sub-MIP is still feasible and contains good solutions. Often their implementation inside a MIP solver demands a sufficiently large percentage of variables to be fixed to arrive at an easy to solve sub-MIP [**5, 6, 13, 17**].

For our heuristic, the situation is different since we do not aim at eliminating integrality constraints, but nonlinearities. In order to linearise a given MINLP, in general we may be forced to fix integer and continuous variables. Especially the fixation of continuous variables in an MINLP can introduce a significant error, even rendering the subproblem infeasible. Thus our heuristic will aim at fixing as *few* variables as possible to obtain as large a linear subproblem as possible.

Remark 22.3. Note that in general a minimum cover does not necessarily yield a dimension-wise largest sub-MIP that can be obtained by fixing variables in a given MINLP. First, this is because in our definition we do not look at the feasible region given by a constraint, but at the graph of its left hand side. Second, we do not take into account the interrelation of constraints with each other and with variable bounds and integrality constraints. Through these interrelations, fixing one variable may lead to further domain reductions and variable fixations which can not immediately be foreseen when looking at each nonlinearity separately.

Hence, searching for a minimum cover may be understood as an approximate method for determining dimension-wise maximal sub-MIPs. Propagation routines using these interrelations can, however, be very effectively integrated within the heuristic, see "fix-and-propagate" in Section 4.

Remark 22.4. We point out the links of our general-purpose approach for MINLP to the works on *bilinear programmes* in global optimization. Here, bilinear programmes are defined as quadratically constrained programmes which allow for a partition of the variable set into two parts such that each quadratic term is bilinear with one variable from each part. Holding the variables in either set fixed, per definition one obtains a linear programme, a simple property which has been used extensively in various solution approaches.

3. Finding minimum covers

This section describes our method for determining a minimum cover of an MINLP, i.e. a minimal subset of variables to fix in order to linearise each constraint. The idea for Undercover originated from our work on solving MIQCPs. Since its application also appears most promising for this class of problems, we start by presenting conditions for covers of quadratic constraints.

Covering quadratic functions. Suppose we are given a quadratic function $g : \mathbb{R}^n \to \mathbb{R}, x \mapsto x^\top Qx$, with $Q \in \mathbb{R}^{n\times n}$ symmetric. Let $x^\star \in \mathbb{R}^n$ and $\mathcal{C} \subseteq \{1,\dots,n\}$. Fixing $x_k = x^\star_k$ for all $k \in \mathcal{C}$ transforms $x^\top Qx$ into $y^\top \tilde{Q}y + \tilde{q}^\top y + \tilde{c}$ with variable vector $y = (x_k)_{k\notin\mathcal{C}} \in \mathbb{R}^{n-|\mathcal{C}|}$, the restricted matrix $\tilde{Q} = (Q_{uv})_{u,v\notin\mathcal{C}}$ of dimension $(n-|\mathcal{C}|)\times(n-|\mathcal{C}|)$, the vector $\tilde{q} = (2\sum_{u\in\mathcal{C}} Q_{uk}x^\star_u)_{k\notin\mathcal{C}} \in \mathbb{R}^{n-|\mathcal{C}|}$, and offset $\tilde{c} = \sum_{u,v\in\mathcal{C}} Q_{uv}x^\star_u x^\star_v$. Thus, the set (22.2) is affine if and only if $\tilde{Q}$ vanishes, i.e. if $q_{uv} = 0$ for all $u, v \notin \mathcal{C}$. In reverse, this means that for $\mathcal{C}$ to be a cover of g, it is necessary and sufficient to contain at least one out of u or v for all nonzero matrix entries Q_{uv}.

This can be interpreted as a *set covering problem*, where items are given by those $(u, v) \in \{1,\dots,n\} \times \{1,\dots,n\}$ with nonzero Q_{uv}, and sets are given by $S(k) := \{(u, v) \mid u = k \text{ or } v = k\}$ for each variable index $k = 1,\dots,n$.

Remark 22.5. Note that in the quadratic case, any $x^\star$-cover is already a global cover, thus the distinction made in Definitions 22.1 and 22.2 is void.

Covering MIQCPs. An MIQCP is an MINLP as in (22.1) where each constraint $i = 1,\dots,m$ takes the form $g_i(x) = x^\top A_i x + b_i{}^\top x + c_i \leqslant 0$ with $A_i \in \mathbb{R}^{n\times n}$ symmetric, $b_i \in \mathbb{R}^n$, and $c_i \in \mathbb{R}$. Matrices A_i are not required to be positive semidefinite, i.e. we allow for nonconvex constraints.

In order to find a cover of a given MIQCP P, we solve the set covering problem outlined above, taking into account all constraints. We introduce auxiliary binary variables α_k, $k = 1,\dots,n$, equal to 1 if and only if x_k is fixed in P. As explained above, $\mathcal{C}(\alpha) := \{k \mid \alpha_k = 1\}$ is a cover of P if and only if

(22.3)
$$\alpha_k = 1 \text{ for all } i \in \{1,\dots,m\}, k \in \{1,\dots,n\}, A^i_{kk} \neq 0, L_k \neq U_k,$$

(22.4)
$$\alpha_k + \alpha_j \geqslant 1 \text{ for all } i \in \{1,\dots,m\}, k \neq j \in \{1,\dots,n\}, A^i_{kj} \neq 0, L_k \neq U_k, L_j \neq U_j,$$

i.e. we require all square terms and one variable in each bilinear term to be fixed. Our heuristic tries to identify as large a linear subproblem as possible. Therefore, we solve the binary programme

$$\min\Big\{\sum_{k=1}^{n} \alpha_k : (22.3), (22.4), \alpha \in \{0,1\}^n\Big\}. \qquad (22.5)$$

The following lemma summarises our discussion from above:

Lemma 22.6. *Let an MIQCP P be given. Then $\alpha \mapsto \mathcal{C}(\alpha) = \{k \mid \alpha_k = 1\}$ gives a one-to-one correspondence between the feasible solutions of* (22.5) *and covers of P. A solution $\alpha^\star$ of* (22.5) *is optimal if and only if $\mathcal{C}(\alpha^\star)$ is a minimum cardinality cover of P.*

Remark 22.7. Note that the above covering problem is an optimisation version of 2-SAT, hence there is a polynomial-time algorithm for its solution. In our computational experiments the binary programme (22.5) could always be solved within a fraction of a second by a general MIP solver, hence we did not employ a specialised polynomial-time algorithm. Moreover, in the case of general MINLP, a cover generating problem contains more involved constraints.

Covering general MINLPs. Our approach for computing covers of quadratic constraints can be applied to more general nonlinearities. The immediate generalisations are multilinear and polynomial constraints. Sufficient conditions for a global cover of a monomial $x_1^{p_1} \cdots x_n^{p_n}$, $p_1, \dots, p_n \in \mathbb{N}_0$, are $\alpha_k = 1$ for all $k \in \{1, \dots, n\}, p_k \geqslant 2, L_k \neq U_k$, and $\sum_{k:p_k=1, L_k \neq U_k} (1 - \alpha_k) \leqslant 1$, similar to (22.3) and (22.4).

As can be seen from this example, with more and more general nonlinearities present, more and more variables need to be fixed to arrive at a linear subproblem. However, note that now the notion of an $x^\star$-cover may be much weaker than that of a global cover.[1]

4. Variants and extensions

The generic algorithm in Figure 1 can be extended and modified in several ways in order to make Undercover more efficient in practise. This section outlines a few of them, with main focus on avoiding infeasibility of the sub-MIPs.

Fix-and-propagate. Fixing a variable can have great impact on the original problem and the approximation we use. Therefore, we do not fix the variables simultaneously, but sequentially one by one, propagating the bound changes after each fixing. If by that, the approximation solution falls outside the feasible domain of a variable, we instead fix it to the closest bound.[2] This fix-and-propagate method resembles a method described in [**16**]. Additionally, we apply it for continuous variables and apply backtracking in case of infeasibility.

Backtracking. If the fix-and-propagate procedure deduces some variable domain to be empty, hence the subproblem to be infeasible, we apply a one-level backtracking, i.e. we undo the last bound change and try other fixing values instead. [3]

Recovering. During the fix-and-propagate routine, variables outside the precomputed cover may also be fixed. In this case, the fixing of some of the yet unfixed

[1]As a simple example consider the multilinear term $x_1 \cdots x_n$ with no variable bounds. The minimum cardinality of an $x^\star$-cover is 1 as soon as $x_k^\star = 0$ for some k. In contrast, the smallest global cover has size $n - 1$.

[2]Alternatively, we could recompute our approximation to obtain values within the current bounds.

[3]In our implementation, we try the bounds of the variable, if finite, and the midpoint between the approximation value and each finite bound. For unbounded variables, we try zero and twice the approximation value.

variables in the cover might become redundant and recomputing the cover may yield a smaller number of variable fixings still necessary.

Different covers. Our initial motivation for fixing as few variables as possible was to minimise the impact on the orginal MINLP. Other measures for the impact of fixing a variable could be used in the objective function of (22.5), such as domain size, appearance in nonlinear terms or nonlinear constraints violated by the approximation solution, variable type, or hybrid measures. In particular, if a minimum cardinality cover yields an infeasible sub-MIP, we may try a cover minimising a different impact measure.

NLP postprocessing. In the spirit of the QCP local search heuristic described in [**7**], we try to further improve the best sub-MIP solution $\tilde{x}$ by solving the NLP which results from fixing all integer variables to their values in $\tilde{x}$.

Convexification. The main idea of Undercover is to reduce the computational effort by finding easier to solve subproblems. While here we have focused on sub-MIPs, for nonconvex MINLPs, already a convex sub-MINLP may be significantly easier to solve and contain more and better solutions than a sub-MIP. This modification of Undercover only requires to weaken the constraints in the cover generating problem (22.5) suitably.

Domain reduction. Instead of fixing the variables in a cover, we could also merely reduce their domains. Especially on continuous variables this leaves significantly more freedom to the subproblem. Since domain propagation is an essential ingredient in MINLP solvers, this might still reduce the computational effort significantly on some problems.

5. Computational experiments

Only few solvers exist that handle general nonconvex MINLPs, such as BARON [**21**], Couenne [**4**], and LindoGlobal [**19**]. Others, such as BONMIN [**9**] and SBB [**3**], guarantee global optimality only for convex problems, but can be used as heuristic solvers for nonconvex problems. Recently, the solver SCIP [**1**] was extended to solve nonconvex MIQCPs to global optimality [**7**].

The target of our computational experiments is to demonstrate the potential of Undercover as a start heuristic for MINLPs applied at the root node. We implemented Undercover within the branch-cut-and-price framework SCIP [**1**] and used SCIP's linear outer approximation solution for the fixing values. We incorporated the fix-and-propagate, backtracking, and NLP postprocessing features described in Section 4. To perform the fix-and-propagate procedure, we used the standard propagation engine of SCIP. Secondary SCIP instances were used to solve both the cover generating problem and the Undercover sub-MIP.

In our experiments, we ran SCIP with all heuristics other than Undercover switched off, set a node limit of 1, and deactivated cut generation. We set a node limit of 500 both for the covering problem and the sub-MIP. For solving the sub-MIP, we used "emphasis feasibility" and "fast presolving" settings. We used SCIP 1.2.0.4 with CPLEX 12.1 [**18**] as LP solver and Ipopt 3.7 [**22**] as NLP solver for the postprocessing. This configuration we refer to as UC.

For comparison, we ran SCIP 1.2.0.4 with CPLEX 12.1 and Ipopt 3.7 in default mode, which applies ten primal heuristics at the root node. We further compared with the state-of-the-art solvers BARON [**21**] (commercial) and Couenne [**4**] (open source). For all solvers, we used node limit 1. Our goal is thus

Table 1. Computational results on MIQCP instances.

instance	nnz/var	% cov	UC	SCIP	BARON	COUENNE
du-opt	0.95	95.24	4233.8709*	4233.8709	108.331477	**41.3038865**
du-opt5	0.95	94.74	3407.05415*	**14.1684489**	–	1226.02232
ex1263	0.34	3.88	**30.1***	–	–	–
ex1264	0.36	4.26	**15.1***	–	–	–
ex1265	0.38	3.52	**15.1***	–	–	**15.1**
ex1266	0.40	3.03	**16.3***	–	–	–
fac3	0.81	78.26	130653857*	–	**38328601.6**	–
feedtray2	10.70	3.26	–	0	0	–
meanvarx	0.19	23.33	16.9968975*	**14.3692129**	14.3692321	18.701873
netmod_dol1	0.00	0.30	0*	**-0.317294979**	0	–
netmod_dol2	0.00	0.38	-0.0780227488*	**-0.50468289**	0	–
netmod_kar1	0.01	0.88	0*	**-0.132809562**	0	–
netmod_kar2	0.01	0.88	0*	**-0.132809562**	0	–
nous1	2.39	19.44	–	–	–	**1.567072**
nous2	2.39	19.44	–	1.38431729	**0.625967412**	1.38431741
nuclear14a	4.98	6.43	–	–	–	–
nuclear14b	2.42	6.43	–	–	–	**-1.11054393**
nvs19	8.00	88.89	–	0	**-1098**	–
nvs23	9.00	90.00	–	0	**-1124.8**	–
product	0.17	30.87	–	–	–	–
product2	0.37	26.15	–	–	–	–
sep1	0.40	10.53	**-510.080984***	–	**-510.080984**	**-510.080984**
space25	0.12	1.04	–	–	–	–
space25a	0.29	5.84	–	–	–	–
spectra2	3.43	35.71	26.6076018*	**23.2840887**	119.8743	–
tln5	1.39	9.09	15.1*	–	–	**14.5**
tln6	1.47	7.69	**32.3***	–	–	–
tln7	1.53	6.67	**30.3***	–	–	–
tln12	1.70	3.99	–	–	–	–
tloss	1.47	7.89	**27.3***	–	–	–
tltr	1.10	12.50	**61.1333333***	–	–	–
util	0.07	3.13	**999.578743***	1000.48517	1006.50609	–
waste	1.10	5.65	693.392795	**693.290675**	712.301232	–

not to compare the Undercover heuristic with SCIP, BARON, and COUENNE as complete solvers – a comparison rather insignificant –, but specifically with the performance of their root heuristics.

As test set we used a selection of 33 MIQCP instances from MINLPLib [**12**]. We excluded `lop97ic`, `lop97icx`, `pb302035`, `pb351535`, `qap`, and `qapw`, which are linear after the default presolving of SCIP. On the `nuclear` instances, the root LP relaxation of SCIP is often unbounded due to unbounded variables in nonconvex terms of the constraints. In this case, we cannot apply Undercover since no fixing values are available. Due to this, we only included two of those instances, `nuclear14a` and `nuclear14b`, for which the root LP of SCIP is bounded.

Results. The results are shown in Table 1. In column "nnz/var" we state the average number of nonlinear nonzeros, i.e. the number of quadratic terms, per variable as an indication of the nonlinearity of the problem.[4] In column "% cov" we report the relative size of the cover used by UC as percentage of the total number of variables after preprocessing, and the objective value of the best solution found by Undercover. A star indicates that the sub-MIP was solved to optimality. For all other solvers, we provide the objective value of the best solution found during root node processing. The best solution among the four solvers is marked bold.

[4]Note that since the cover generating problem contains one constraint for each nonlinear nonzero, this corresponds to the ratio of items to sets of the set covering problem solved.

The computational results seem to confirm our expectation that often a low fixing rate suffices to obtain a linear subproblem: 12 of the instances in our test set allow a cover of at most 5% of the variables, further 10 instances of at most 15%. On the remaining third of the test set, a minimum cover contains 19–96% of the variables.

UC found a feasible solution for 21 test instances: on 16 out of the 22 instances with a cover of at most 15% of the variables, and on 5 instances in the remaining third of the test set. In comparison, BARON found a feasible solution in 15 cases, COUENNE in 9, SCIP in 14. We note that on 7 instances UC found a solution, although none of BARON, COUENNE, and SCIP did. UC could solve `ex1266` to optimality and `util` to 0.1% primal-dual gap.

To evaluate the solution quality of Undercover, for each other solver consider the instances on which both UC and this solver found a solution: On those instances the solution found by Undercover is better than the one found by SCIP in 1 case (equal in 1, worse in 8), better than BARON 4 times (equal 4, worse 3), and better than COUENNE in 1 case (equal in 2, worse in 3 cases).

The overall time for SCIP preprocessing, solving the root LP and applying Undercover was always less than two seconds. Thereof, the time for applying Undercover was always less than 0.2 seconds, except for the instance `waste`, where Undercover ran for 1.1 seconds. The major amount of time was usually spent in solving the sub-MIP. Although the polytope described by (22.5) is not integral, the covering instance could always be solved to optimality in the root node by SCIP's default heuristics and cutting plane algorithms. We note that in 10 out of the 14 cases where the resulting sub-MIP was infeasible, the infeasibility was already detected during the fix-and-propagate stage. Thus in most cases, no time was invested in vain to try and find a solution to an infeasible subproblem. Also, except for instance `waste`, all feasible sub-MIPs could be solved to optimality within the imposed node limit of 500, which indicates that – with a state-of-the-art MIP solver at hand – the generated subproblems are indeed significantly easier than the full MINLP.

6. Conclusion and future work

Altogether, Undercover seems to be a fast start heuristic, that often produces feasible solutions of reasonable quality. On the chosen test set, the experiments confirmed our expectation that a low fixing rate often suffices to obtain a feasible linear subproblem which is easy to solve. The computational results indicate, that it complements nicely with existing root node heuristics in different solvers.

Future research will focus on fully implementing and testing the described features and variants in Section 4 and experimenting with fixing values from other approximations, especially solutions of standard NLP relaxations.

Acknowledgments

This research was partially funded by the DFG Research Center MATHEON, Project B20. We thank GAMS Development Corp. for providing us with evaluation licenses for BARON. Many thanks to Stefan Vigerske for his valuable comments and to Mark Wallace for his proofreading.

References

1. T. Achterberg. *Constraint Integer Programming.* PhD thesis, Technische Universität Berlin, 2007.
2. T. Achterberg and T. Berthold. Improving the Feasibility Pump. *Disc. Opt.*, Special Issue 4(1):77–86, 2007.
3. ARKI Consulting & Development A/S and GAMS Inc. SBB. `http://www.gams.com/solvers/solvers.htm#SBB`.
4. P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Waechter. Branching and bounds tightening techniques for non-convex MINLP. Research Report RC24620, IBM, 2008.
5. T. Berthold. Primal heuristics for mixed integer programs. Diploma thesis, Technische Universität Berlin, 2006.
6. T. Berthold. RENS – Relaxation Enforced Neighborhood Search. ZIB-Report 07-28, Zuse Institute Berlin, 2007.
7. T. Berthold, S. Heinz, and S. Vigerske. Extending a CIP framework to solve MIQCPs. ZIB-Report 09-23, Zuse Institute Berlin, 2009.
8. R.E. Bixby, M. Fenelon, Z. Gu, E. Rothberg, and R. Wunderling. MIP: Theory and practice – closing the gap. In M.J.D. Powell and S. Scholtes, editors, *Systems Modelling and Optimization: Methods, Theory, and Applications*, pages 19–49. Kluwer Academic Publisher, 2000.
9. P. Bonami, L.T. Biegler, A.R. Conn, G. Cornuéjols, I.E. Grossmann, C.D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, and A. Wächter. An algorithmic framework for convex mixed integer nonlinear programs. *Disc. Opt.*, 5:186–204, 2008.
10. P. Bonami, G. Cornuéjols, A. Lodi, and F. Margot. A feasibility pump for mixed integer nonlinear programs. *Math. Prog.*, 119(2):331–352, 2009.
11. P. Bonami and J.P.M. Gonçalves. Primal heuristics for mixed integer nonlinear programs. Research Report RC24639, IBM, 2008.
12. M.R. Bussieck, A.S. Drud, and A. Meeraus. MINLPLib – A Collection of Test Models for Mixed-Integer Nonlinear Programming. *INFORMS Journal on Computing*, 15(1):114–119, 2003.
13. E. Danna, E. Rothberg, and C. Le Pape. Exploring relaxation induced neighborhoods to improve MIP solutions. *Math. Prog. A*, 102(1):71–90, 2004.
14. M. Fischetti, F. Glover, and A. Lodi. The feasibility pump. *Math. Prog. A*, 104(1):91–104, 2005.
15. M. Fischetti and A. Lodi. Local branching. *Math. Prog. B*, 98(1-3):23–47, 2003.
16. M. Fischetti and D. Salvagnin. Feasibility pump 2.0. *Math. Prog. C*, 1:201–222, 2009.
17. S. Ghosh. DINS, a MIP improvement heuristic. In *Proc. of 12th IPCO*, pages 310–323, 2007.
18. ILOG, Inc. CPLEX. `http://www.ilog.com/products/cplex`.
19. Lindo Systems, Inc. LindoGlobal. `http://www.lindo.com`.
20. G. Nannicini, P. Belotti, and L. Liberti. A local branching heuristic for MINLPs. *ArXiv e-prints*, 2008.
21. M. Tawarmalani and N.V. Sahinidis. Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. *Math. Prog.*, 99:563–591, 2004.

22. A. Wächter and L.T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Math. Prog.*, 106(1):25–57, 2006.

23

Eigenvalue techniques for convex objective, nonconvex optimization problems

Daniel Bienstock

Columbia University
New York, NY, USA

dano@columbia.edu

Consider a minimization problem given by a nonlinear, convex objective function over a nonconvex feasible region. Traditional optimization approaches will frequently encounter a fundamental difficulty when dealing with such problems: even if we can efficiently optimize over the convex hull of the feasible region, the optimum will likely lie in the interior of a high dimensional face, “far away” from any feasible point. As a result (and in particular, because of the nonlinear objective) the lower bound provided by a convex relaxation will typically be extremely poor. Furthermore, we will tend to see very large branch-and-bound (or -cut) trees with little or no improvement over the lower bound.

In this work we present theory and implementation for an approach that relies on three ingredients: (a) the S-lemma, a major tool in convex analysis (b) efficient projection of quadratics to lower dimensional hyperplanes, and (c) efficient computation of combinatorial bounds for the minimum distance from a given point to the feasible set, in the case of several signficant optimization problems.

Our current approach is guaranteed to work in case the objective is a convex quadratic. More generally, if the objective can be (locally) lower bounded by a convex quadratic, we obtain a (locally) valid lower bound; thus the approach fits well within the trust region framework for (local) optimization over non-convex sets. We will present computational experience with large cardinality constrained problems over the unit simplex. If time permits we will present experience with other problem classes.

24

A heuristic algorithm for the general nonlinear separable knapsack problem

Claudia D'Ambrosio Silvano Martello

DEIS, University of Bologna
2, Viale Risorgimento
Bologna, I-40136 Italy

{c.dambrosio,silvano.martello}@unibo.it

Keywords: nonlinear knapsack, nonconvex, separable, heuristic, local search, mixed integer nonlinear programming.

The general nonlinear knapsack problem is

$$\begin{aligned} &\max f(x)\\ &g(x) \leqslant c\\ &x \in D \end{aligned}$$

where $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$, $f(x)$ and $g(x)$ are continuous differentiable functions, and $D \subseteq \mathbb{R}^n$. In this work we consider the case where both $f(x)$ and $g(x)$ are separable functions, and D includes bounds and integrality requirements on (part of) the variables. The resulting problem can thus be described, using the knapsack terminology, as follows. Given n items j, each having a *profit* function $f_j(x_j)$ and a *weight* function $g_j(x_j)$, associated with n variables x_j limited by upper bounds u_j ($j \in N = \{1, \ldots, n\}$), determine non-negative x_j values such that the total weight does not exceed a given *capacity* c and the total produced profit is a maximum. A subset $\overline{N}$ of the x_j variables can be restricted to take integer values. Formally, we consider the Nonlinear Knapsack problem (NLK)

$$\begin{aligned} &\max \sum_{j \in N} f_j(x_j) \\ &\sum_{j \in N} g_j(x_j) \leqslant c \\ &0 \leqslant x_j \leqslant u_j \qquad \forall j \in N = \{1, \ldots, n\} \\ &x_j \text{ integer} \qquad \forall j \in \overline{N} \subseteq N \end{aligned}$$

where $f_j(x_j)$ and $g_j(x_j)$ are nonlinear increasing functions and the unique constraint is the capacity constraint. We have no further assumption on functions $f_j(x_j)$ and $g_j(x_j)$, i.e., they can be nonconvex and nonconcave.

The well-known (linear) knapsack problem (see Martello and Toth [**9**] or Kellerer, Pferschy and Pisinger [**8**]) is the special case of NLK arising when $f_j(x_j)$ and $g_j(x_j)$ are linear integer functions, i.e., $f_j(x_j) = p_j x_j$ and $g_j(x_j) = w_j x_j$ $\forall j \in N$ and $\overline{N} = N$. It follows that NLK is $\mathcal{NP}$-hard.

Nonlinear knapsack problems have many applications in various fields such as, e.g., portfolio selection, stratified sampling, production planning, resource distribution. We refer the reader to Ibaraki and Katoh [**6**] and to Bretthauer and Shetty [**2**] for extensive reviews.

A number of nonlinear separable knapsack problems has been considered in the literature. In most cases the proposed algorithms are specifically tailored to the case of convex or concave functions. In our model neither convexity nor concavity is assumed, a situation rarely treated in the literature (apart from contributions on the general global optimization, see Horst and Tuy [**5**]).

The heuristic algorithm we present starts by computing profit and weight values over a discretized solution space. Let $\bar{s}$ denote the number of samplings (identical for all functions), so $\delta_j = u_j/\bar{s}$ is the *sampling step*, and define

$$f_{jk} = f_j(k\delta_j) \quad \text{and} \quad g_{jk} = g_j(k\delta_j) \qquad (j \in N; k = 1, \dots, \bar{s}).$$

We then obtain the profit-to-weight ratios as $r_{jk} = \frac{f_{jk}}{g_{jk}}$ $(j \in N, k = 1, \dots, \bar{s})$, and we can compute, for each item j, the maximum ratio $r_{j,m(j)} = \max_{k=1,\dots,\bar{s}}\{r_{jk}\}$.

Assume by simplicity that the items are sorted according to nondecreasing $r_{j,m(j)}$ values, so, with respect to the current sampling step, $r_{1,m(1)}$ is the best available ratio and $m(1)\delta_1$ is the value of variable x_1 that produces the best filling of the first $g_{1,m(1)}$ capacity units. Consider now the second best item 2, and its best ratio $r_{2,m(2)}$ $(\leqslant r_{1,m(1)})$, and observe that item 1 could have a better ratio than item 2 also for a higher x_1 value. Specifically, we can find the highest $m'(1)$ value such that $r_{1,m'(1)} \geqslant r_{2,m(2)}$.

The idea is then to define $x_1 = m'(1)\delta_1$, i.e., to use item 1 for the first $g_{1,m'(1)}$ capacity units, then to continue with the residual capacity $\bar{c} = c - g_{1,m'(1)}$. At the second iteration, item 2 (second best available ratio) is used for the next $g_{2,m'(2)}$ capacity units, where $m'(2)$ is analogously defined as the highest value such that $r_{2,m'(2)} \geqslant r_{3,m(3)}$, and so on.

The method is further improved by refining the search for the best value $m'(j)$ of the current x_j (by iteratively decreasing the sampling step), and by applying a post-processing. The detailed statement of the overall algorithm can be found in D'Ambrosio and Martello [**4**].

The algorithm, denoted as Heur in the following, was experimentally evaluated on different sets of randomly generated functions. The general function used for generating the profits (for all test beds) was $f_j(x_j) = c_j/(1 + b_j \exp^{-a_j(x_j+d_j)})$. Different actual functions were obtained, for each item j, by randomly generating a_j, b_j, c_j and d_j in various intervals.

Different weight functions were used for different tests. A first test bed was obtained by generating the weights according to the concave non-decreasing functions $g_j(x_j) = \sqrt{p_j x_j + q_j} - \sqrt{q_j}$, obtaining different actual functions, for each item j, by randomly generating p_j and q_j. A second test bed was produced using linear non-decreasing weight functions $g_j(x_j) = w_j x_j$, by randomly generating the w_j values.

The algorithm was compared with a number of solvers, namely

- heuristic solvers: Ipopt [**7**] for real instances, Bonmin [**1**] for integer instances;
- exact solvers: SC-MINLP [**3**], both for real and integer instances.

For the cases with real variables, Algorithm Heur produced better average solutions than Ipopt, with CPU times that are roughly of the same (or lower) order of magnitude with respect to Ipopt. SC-MINLP produced better solutions than Heur (by about 2%) for $n \leqslant 200$, worse solutions (by about 2%) for $500 \leqslant n \leqslant 1000$, and no solution at all for $n = 2500$. Its computing time was however between 3 and 5 orders of magnitude higher than that of Heur.

For the cases with integer variables, the results were even better for Heur. It produced better average solutions than Bonmin, with CPU times that are always one or two orders of magnitude smaller than the two solvers. SC-MINLP produced slightly better solutions than Heur only for $n \leqslant 20$, slightly worse solutions for $50 \leqslant n \leqslant 100$, and no solution at all for $n \geqslant 200$.

Overall, our heuristic finds solutions of excellent quality for a very difficult problem within extremely short CPU times.

References

1. Bonmin. `http://projects.coin-or.org/Bonmin`.
2. K.M. Bretthauer and B. Shetty. The nonlinear knapsack problem - algorithms and applications. *European Journal of Operational Research*, 138:459–472, 2002.
3. C. D'Ambrosio, J. Lee, and A. Wächter. A global-optimization algorithm for mixed-integer nonlinear programs having separable non-convexity. *A. Fiat and P. Sanders (Eds.): ESA 2009 (17th Annual European Symposium. Copenhagen, Denmark, September 2009), Lecture Notes in Computer Science*, 5757:107–118, 2009.
4. C. D'Ambrosio and S. Martello. Heuristic algorithms for the general nonlinear separable knapsack problemy. Technical Report OR-09-15, DEIS, University of Bologna, 2009.
5. R. Horst and H. Tuy. *Global optimization: deterministic approaches*. Springer, Berlin, Germany, 1990.
6. T. Ibaraki and N. Katoh. *Resource allocation problems*. MIT Press, Cambridge, MA, 1988.
7. Ipopt. `http://projects.coin-or.org/Ipopt`.
8. H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack problems*. Springer, Berlin, Germany, 2004.
9. S. Martello and P. Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Chichester-New York, 1990.

An Outer Approximation Algorithm for Nonlinear Mixed-Integer Programming based on Sequential Quadratic Programming with Trust Region Stabilization

Oliver Exler[1] **Thomas Lehmann**[2] **Klaus Schittkowski**[1]

[1] Department of Computer Science
University of Bayreuth
Bayreuth, 95440, Germany
`{oliver.exler,klaus.schittkowski}@uni-bayreuth.de`

[2] Konrad-Zuse-Zentrum (ZIB)
Takustr. 7
Berlin-Dahlem, 14195, Germany
`lehmann@zib.de`

Abstract

We propose a modified outer approximation algorithm for solving mixed-integer nonlinear programming problems. Successive quadratic approximations stabilized by a trust region method are applied to increase the efficiency and robustness of a linear outer approximation algorithm. Numerical results are presented for a set of 100 academic mixed-integer test problems. The number of function evaluations, a very important performance criterion in practice, is significantly decreased and solution quality of non-convex problems is increased compared to a linear outer approximation algorithm.
Keyword: linear outer approximation, mixed integer nonlinear programming, MINLP, numerical algorithms.

1. Introduction

We consider the general mixed-integer nonlinear optimization problem (MINLP) to minimize an objective function f under m nonlinear inequality constraints,

$$x \in X, y \in Y : \quad \begin{array}{l} \min f(x,y) \\ g_j(x,y) \geqslant 0 \ , \quad j = 1, \ldots, m \ , \end{array} \tag{25.1}$$

where x and y denote the vectors of the continuous and integer variables, respectively. The two sets X and Y are defined as follows

$$\begin{array}{lcl} X & := & \{x \in \mathbb{R}^{n_c} : x_{lb} \leqslant x \leqslant x_{ub}\} \ , \\ Y & := & \{y \in \mathbb{N}^{n_i} : y_{lb} \leqslant y \leqslant y_{ub}\} \ , \end{array} \tag{25.2}$$

where n_c is the number of continuous variables and n_i is the number of integer variables, i.e. bound constraints on the variables are added. It is assumed that the problem functions $f(x,y)$ and $g_j(x,y)$, $j = 1, \ldots, m$, are continuously differentiable.

Numerous algorithms have been proposed in the past, see for example Floudas [**8**] or Grossmann [**9**] for review papers. Typically, these approaches require convex model functions and continuous relaxations of integer variables. By a continuous relaxation, we understand that integer variables can be treated as continuous variables, i.e., function values can also be computed for all $y \in Y_{\mathbb{R}}$, where

$$Y_{\mathbb{R}} := \{y \in \mathbb{R}^{n_i} : y_l \leqslant y \leqslant y_u\} \ . \tag{25.3}$$

denotes the convex hull of Y. Thus, the corresponding continuous relaxation of problem (25.1) is

$$x \in X, y \in Y_{\mathbb{R}} : \quad \begin{array}{l} \min f(x,y) \\ g_j(x,y) \geqslant 0 \ , \quad j = 1, \ldots, m \ . \end{array} \tag{25.4}$$

During the last years, branch-and-bound methods were developed where a series of relaxed nonlinear programs must be solved obtained by restricting the variable range of the relaxed integer variables, see Gupta and Ravindran [**10**] or Borchers and Mitchell [**2**]. When applying an SQP algorithm for solving a subproblem, it is possible to apply early branching, see Leyffer [**13**]. Pattern search algorithms are available to search the integer space, see Audet and Dennis [**1**]. After replacing the integrality condition by continuous nonlinear constraints, it is possible to solve the resulting highly non-convex program by a global optimization algorithm, see e.g. Li and Chou [**14**]. Alternatively, it is also possible to apply cutting planes as in linear programming, see Westerlund and Pörn [**16**].

Another frequently used solution method for solving convex mixed-integer nonlinear programming problems is based on linear outer approximations. The idea is introduced by Duran and Grossmann [**3**] and is extended by Fletcher and Leyffer [**7**] to solve general convex mixed-integer nonlinear programs. Convergence towards the global optimal solution of a convex mixed-integer nonlinear program is guaranteed by considering a linear model of the original convex MINLP called *master problem.* Since linearized constraints remain valid during the whole optimization process, the method cannot be applied directly to non-convex problems. It might be possible that the optimal solution is cut off.

A mixed-integer sequential quadratic programming method was introduced by Exler and Schittkowski [**6**]. We refer to this method as MISQP. The algorithm is based on the trust region method of Yuan [**17**] and is adapted to solve nonlinear mixed-integer optimization problems, e.g., by solving a sequence of mixed-integer quadratic subproblems. It is well known that the continuous version converges to a stationary point. Although the algorithm is extremely efficient for mixed-integer nonlinear problems and stops at a feasible solution after very few iterations, in most cases at the optimal solution, convergence of the underlying algorithm cannot be guaranteed in a rigorous mathematical way, even not in the convex case.

We present an algorithm that overcomes the disadvantages of both linear outer approximation and MISQP while maintaining their benefits. Our method combines mixed integer search steps obtained by local quadratic approximations with linear outer approximation techniques. The new method has the following properties:

- It guarantees global optimality for convex problems.
- It is efficient in terms of the number of function evaluations.
- It obtains good solutions for non-convex MINLPs.

The article is structured as follows. Sections 2 and 3 summarize the theoretical background of the methods we combined in our algorithm. Our algorithm is described in Section 4. Numerical results obtained for a selection of 100 academic mixed integer nonlinear test cases [**15**] are reported in section 5. Section 6 provides some conclusions.

2. Linear Outer Approximation

In order to provide convergence to a global solution of convex MINLPs (25.1), the new algorithm uses a *master problem* as outer approximation algorithms. The master problem will be formulated later, but requires some definitions first.

To simplify the notation we define $g(x,y) := (g_1(x,y),\ldots,g_m(x,y))^T$ and the transpose of the Jacobian of the constraints $\nabla g(x,y) := (\nabla g_1(x,y),\ldots,\nabla g_m(x,y))$. We use $g(x,y)^- \in \mathbb{R}^m$ that is defined as

$$g_j(x,y)^- := min(g_j(x,y),0)\ , \quad j=1,\ldots,m. \tag{25.5}$$

For any fixed $\bar{y} \in Y$, we denote by NLP($\bar{y}$) the nonlinear program

$$x \in X : \begin{array}{l} \min\ f(x,\bar{y}) \\ \\ g(x,\bar{y}) \geqslant 0\ , \end{array} \tag{25.6}$$

and by $x(\bar{y})$ its solution. Outer approximation algorithms require the exact solution of these NLP(y) (25.6) problems because they are essential to generate the master problem. In the following, we denote by T the set of integer values leading to feasible nonlinear subproblems,

$$T = \{y \in Y : NLP(y) \text{ feasible}\}\ . \tag{25.7}$$

Analogously, we denote the set of integer values y with infeasible subproblems by S, i.e.

$$S = \{y \in Y : NLP(y) \text{ infeasible}\}\ . \tag{25.8}$$

Note that $Y = T \cup S$. Consider now $y \in S$ and let

$$J(y) := \{j : 1 \leqslant j \leqslant m, \exists x \in X \text{ with } g_j(x,y) \geqslant 0\}\ . \tag{25.9}$$

With $J^\perp(y) := \{1, ..., m\} \backslash J(y)$, we obtain a feasibility problem $F(y)$ from (25.1)

$$x \in X : \begin{array}{l} \min \sum_{l \in J^\perp(y)} w_l g_l(x,y)^- \\ g_j(x,y) \geqslant 0, \ j \in J(y) \ , \end{array} \tag{25.10}$$

where w_l are appropriate nonnegative weights not all at once equal to zero and $g_l(x,y)^-$ as defined in (25.5). We denote by $x_F(y)$ the solution of (25.10) subject to $y \in S$.

Standard assumptions required to apply outer approximation methods and to guarantee global optimality are:

Assumption 1. (1) The relaxed program (25.4) is convex, i.e., $f(x,y)$ is convex and $g(x,y)$ is concave over X and $Y_{\mathbb{R}}$.
(2) $f(x,y)$ and $g(x,y)$ are continuously differentiable for all $x \in X$ and $y \in Y_{\mathbb{R}}$.
(3) The linear independent constraint qualification (LICQ) holds at the solution $x(y)$ of every nonlinear program NLP(y) (25.6).

The following Theorem was shown by Fletcher and Leyffer [**7**] and defines the *master problem.*

Theorem 2.1. If Assumption 1 holds, then the *master problem*

$$\begin{array}{lll} \min & \eta & \\ & \eta \geqslant f(x(y),y) + \nabla f(x(y),y)^T \begin{pmatrix} x - x(y) \\ z - y \end{pmatrix} & \forall y \in T \\ x \in X, z \in Y, \eta \in \mathbb{R} : & 0 \leqslant g(x(y),y) + \nabla g(x(y),y)^T \begin{pmatrix} x - x(y) \\ z - y \end{pmatrix} & \forall y \in T \\ & 0 \leqslant g(x_F(y),y) + \nabla g(x_F(y),y)^T \begin{pmatrix} x - x_F(y) \\ z - y \end{pmatrix} & \forall y \in S \end{array} \tag{25.11}$$

is equivalent to the convex MINLP (25.1) in that sense, that (x^*, y^*) solves the convex MINLP (25.1), if and only if it solves the master problem (25.11).

3. A Sequential Quadratic Programming Algorithm with Trust Region Stabilization

Exler and Schittkowski [**6**] extended the trust region algorithm suggested by Yuan [**17**] to the class of mixed-integer nonlinear programs. The subproblems are transformed into mixed-integer quadratic problems (MIQP).

The method is based on the exact L_∞-penalty function

$$P_\sigma(x,y) = f(x,y) + \sigma \| g(x,y)^- \|_\infty \ , \tag{25.12}$$

where $\sigma > 0$ is the associated penalty parameter.

To approximate $P_{\sigma_k}(x_k, y_k)$ in the k-th iteration step, where (x_k, y_k) is a current iterate, we successively solve the subproblem

(25.13)

$$\begin{array}{rl} & \min \quad \nabla f(x_k,y_k)^T d + \frac{1}{2} d^T B_k d + \sigma_k \|(g(x_k,y_k) + \nabla g(x_k,y_k)^T d)^-\|_\infty \\ d^c \in \mathbb{R}^{n_c}, d^i \in \mathbb{N}^{n_i}: & \|d^c\|_\infty \leqslant \Delta_k^c \ , \ \|d^i\|_\infty \leqslant \Delta_k^i \ , \\ & x_k + d^c \in X, y_k + d^i \in Y \ , \end{array}$$

where $d := \begin{pmatrix} d^c \\ d^i \end{pmatrix}$ with $d^c \in \mathbb{R}^{n_c}$ and $d^i \in \mathbb{N}^{n_i}$. It is assumed that $B_k \in \mathbb{R}^{(n_c+n_i)\times(n_c+n_i)}$ is positive definite for all k. Note that a solution of (25.13) always exists, now denoted by $d_k := \begin{pmatrix} d_k^c \\ d_k^i \end{pmatrix}$. Moreover, it is guaranteed that the subsequent iterate $x_{k+1} := x_k + d_k^c$, $y_{k+1} := y_k + d_k^i$, if accepted, satisfies the bounds given by (25.2). $\Delta_k^c > 0$ and $\Delta_k^i \geqslant 0$ denote the trust region radii for the continuous and integer search space, respectively. We restrict the trust region radius associated with the integer or binary variables to integer values, i.e $\Delta_k^i \in \mathbb{N}^{n_i}$. For binary variables, we have either $\Delta_k^i = 0$ or $\Delta_k^i = 1$.

During the remainder of this article we denote the objective function of the mixed-integer subproblem (25.13) by

$$\Phi_k(d) := \nabla f(x_k,y_k)^T d + \frac{1}{2} d^T B_k d + \sigma_k \|(g(x_k,y_k) + \nabla g(x_k,y_k)^T d)^-\|_\infty \ . \tag{25.14}$$

Since (25.13) is non-smooth, we introduce a slack variable $\eta \in \mathbb{R}$ to reformulate (25.13) as a mixed-integer quadratic programming problem

(25.15)

$$\begin{array}{rl} & \min \ \nabla f(x_k,y_k)^T d + \frac{1}{2} d^T B_k d + \sigma_k \eta \\ & \eta + g_j(x_k,y_k) + \nabla g_j(x_k,y_k)^T d \ \geqslant \ 0 \ , j = 1,\ldots,m \ , \\ d^c \in \mathbb{R}^{n_c}, d^i \in \mathbb{N}^{n_i}, \eta \in \mathbb{R}: & \|d^c\|_\infty \leqslant \Delta_k^c \ , \ \|d^i\|_\infty \leqslant \Delta_k^i \ , \\ & x_k + d^c \in X, y_k + d^i \in Y \ , \ \eta \geqslant 0 \ . \end{array}$$

The matrix $B_k \in \mathbb{R}^{(n_c+n_i)\times(n_c+n_i)}$ is updated during the optimization process. Best results are obtained by using a BFGS quasi-Newton formula, but other updates are possible.

The solution d_k of problem (25.15) is used to generate the next trail point. Depending on the ratio of actual and predicted reduction

$$r_k = \frac{P_{\sigma_k}(x_k,y_k) - P_{\sigma_k}(x_k + d_k^c, y_k + d_k^i)}{\Phi_k(0) - \Phi_k(d_k)} \ , \tag{25.16}$$

the step is accepted or rejected. The trust region radii are adjusted accordingly. This is a standard procedure for trust region methods.

Regarding the case with fixed integer variables $\bar{y}$, under the following Assumptions 2, one can prove convergence to at least a stationary point of NLP($\bar{y}$) (see Yuan [**17**]).

Assumption 2. (1) $f(x,y)$ and $g_j(x,y)$, $j = 1,\ldots,m$ are continuously differentiable.

(2) $\{B_k\}$ is uniformly bounded.

Extensive tests on academic and real-world problems show that the implementation MISQP [**4**] of this method is very efficient in terms of the number of function evaluations needed to obtain good solutions. But convergence to an optimal solution of problem (25.1) can not be guaranteed, even not for convex problems.

4. A Linear Outer Approximation Algorithm Combined with SQP and Trust Region Stabilization

In this section we introduce the algorithm that combines linear outer approximation and the method MISQP, described in Section 3. The goal is to develop a solution method that is on the one hand efficient in terms of the number of function and gradient evaluations needed to obtain a solution. On the other hand global optimality should be guaranteed for convex MINLP problems (25.1). Furthermore the algorithm should be able to find high quality solutions for non-convex problems.

Global optimality can be guaranteed for convex MINLPs by the global approximation built up in the mixed integer linear master problem (25.11). Therefore, we have to solve the continuous NLP($\bar{y}$) (25.6) for a fixed $\bar{y}$. In our opinion, the performance of an outer approximation algorithm can be improved if we allow changes in the integer variables during this continuous optimization process.

A change in the integer variables seems to be preferable if it reduces the L_∞-penalty function more than the step obtain with fixed integer values. This is a so-called *improving mixed integer search direction* as defined as follows.

Definition 25.1. A solution $d_k = \begin{pmatrix} d_k^c \\ d_k^i \end{pmatrix}$ of problem (25.15) with $d_k^i \in \mathbb{N}^{n_i}$ is an improving mixed integer search direction, if

$$P_{\sigma_k}(x_k + d_k^c, y_k + d_k^i) < P_{\sigma_k}(x_k + \hat{d}_k^c, y_k), \tag{25.17}$$

where $\hat{d}_k := \begin{pmatrix} \hat{d}_k^c \\ 0 \end{pmatrix}$ is the solution of the MIQP (25.15) with fixed $\hat{d}_k^i$, i.e. $\Delta_k^i = 0$.

The basic algorithm can be stated as follows:

Note that by setting the Boolean variable *IntegerStep* to *false*, we skip step 2. Thus, executing steps 1 to 8 will generate a sequence that converges to a stationary point of problem NLP(y_k) (25.6) with fixed y_k. If a stationary point is detected, step 8 is executed and the master problem (25.11) is solved. This corresponds to a linear outer approximation algorithm as described by Fletcher and Leyffer [**7**].

Our algorithm differs from a linear outer approximation algorithm by adding step 2. In step 2 search directions are calculated that also involves the integer variables. We allow changes in the integer variables y_k at an early stage of the optimization process. Using this strategy, generates solutions that are significantly better compared to those obtained by a linear outer approximation algorithm without this feature, especially if problem (25.1) is non-convex. Note that in step 8 it is necessary to check whether $y_{k_{OA}}$ differs from $y_{l_{OA}}$, for all $l_{OA} = 1, \ldots, k_{OA} - 1$. Otherwise cycling can occur.

Under the assumption

Assumption 3. (1) Let the penalty parameter σ satisfy

$$\sigma > \max\{\|u_j\|_1 : \quad j \geqslant 1,\ NLP(y_j) \text{ feasible}\} \tag{25.18}$$

```
1 Let $k := 1,\ x_1 \in X,\ y_1 \in Y,\ \sigma_1 > 0,\ B_1 \in \mathbb{R}^{(n_c+n_i)\times(n_c+n_i)}$ symmetric
  positive definite. Evaluate function values $f(x_1, y_1)$, $g(x_1, y_1)$ and gradients
  $\nabla f(x_1, y_1)$, $\nabla g(x_1, y_1)$. Set $k_{OA} := 1$ and *IntegerStep*:=true.
    (1) Solve MIQP (25.15) with $\Delta_k^i = 0$ fixed, giving $\hat{d}_k = (\hat{d}_k^c, 0)$ and evaluate
        $f((x_k, y_k) + \hat{d}_k)$ and $g((x_k, y_k) + \hat{d}_k)$ to obtain $P_{\sigma_k}((x_k, y_k) + \hat{d}_k)$.
        Calculate
            $$\hat{r}_k = \frac{P_{\sigma_k}(x_k, y_k) - P_{\sigma_k}((x_k, y_k) + \hat{d}_k)}{\Phi_k(0) - \Phi_k(\hat{d}_k))} .$$
          If *IntegerStep*=true, then go to 2.
          Else set $d_k := \hat{d}_k$ and $r_k := \hat{r}_k$. Go to 4.
    (2) Solve the MIQP (25.15) with $d_k^i \in \mathbb{N}^{n_i}$, giving $d_k = (d_k^c, d_k^i)$ and evaluate
        $f((x_k, y_k) + d_k)$ and $g((x_k, y_k) + d_k)$ to obtain $P_{\sigma_k}((x_k, y_k) + d_k)$.
        Calculate
            $$r_k = \frac{P_{\sigma_k}(x_k, y_k) - P_{\sigma_k}((x_k, y_k) + d_k)}{\Phi_k(0) - \Phi_k(d_k)} .$$
    (3) If
            $$P_{\sigma_k}((x_k, y_k) + d_k) < P_{\sigma_k}((x_k, y_k) + \hat{d}_k),$$
          then use $d_k$ and $r_k$. *(Improving mixed integer search direction)*
          Else set $d_k := \hat{d}_k$ and $r_k := \hat{r}_k$.
    (4) If $\Phi_k(0) - \Phi_k(d_k) = 0$ and convergence to stationary point, i.e. KKT
        point or infeasible stationary point of NLP($y_k$) (25.6) reached, then go
        to 8.
    (5) Dependent on $r_k$, reduce or increase the trust region radii $\Delta_{k+1}^c$ and
        $\Delta_{k+1}^i$ (apply standard trust region rules).
          If necessary, increase penalty parameter $\sigma_{k+1}$.
    (6) If $r_k \leqslant 0$, set $(x_{k+1}, y_{k+1}) := (x_k, y_k)$, $B_{k+1} := B_k$ and k:=k+1 and go to
        1.
    (7) Set $(x_{k+1}, y_{k+1}) := (x_k, y_k) + d_k$. Evaluate $\nabla f(x_{k+1}, y_{k+1})$ and
        $\nabla g(x_{k+1}, y_{k+1})$. Update $B_{k+1}$ and set $k := k + 1$. Go to 1.
    (8) Set $k_{OA} := k_{OA} + 1$. Update *linear outer approximation master probelm*
        (25.11) by adding linearizations and solve (25.11), giving $(x_{k_{OA}}, y_{k_{OA}})$.
          • If $y_{k_{OA}} \neq y_{l_{OA}}$, for all $l_{OA} = 1, \ldots, k_{OA} - 1$, then set
            *IntegerStep*:=true.
          • Else set *IntegerStep*:=false.
        Set $(x_{k+1}, y_{k+1}) = (x_{k_{OA}}, y_{k_{OA}})$. Evaluate function values $f(x_k, y_k)$,
        $g(x_k, y_k)$ and gradients $\nabla f(x_{k+1}, y_{k+1})$, $\nabla g(x_{k+1}, y_{k+1})$. Set k:=k+1
        and go to 1.
```

and

(25.19)
$$\sigma > \max\left\{\frac{f(x^*, y^*) - f(x_k, y_k)}{\|g(x_k, y_k)^-\|_\infty} : \quad k \geqslant 1, \|g(x_k, y_k)^-\|_\infty > 0 \text{ and } f(x_k, y_k) < f(x^*, y^*)\right\},$$

Solver	Optimal	Time	Func	Grad	Equ. Func	Deviation
OA	57	0.2412	370	32	492	163.3141
MISQPN	74	0.3831	403	33	532	19.6153
MISQP	89	0.6617	663	31	777	1.5022
MISQPOA	94	0.9190	3,071	137	3,917	0.3570
MINLPB4	90	0.2517	4,879	4,432	137,581	6.1644

Table 1. Test Results for Academic Test Cases

(2) A second order sufficient condition hold for all y_j such that $NLP(y_j)$ is feasible,

where u_j denote the unique Lagrange multipliers at the solution $x(y_j)$ of $NLP(y_j)$ and (x^*, y^*) a global solution of MINLP (25.1), the following Theorem can be shown.

Theorem 4.1. If Assumption 1 and Assumption 3 hold, Algorithm 2 terminates at a global solution (x^*, y^*) of the convex MINLP (25.1).

The proof relies on the fact that cycling for both the outer approximation and the nonlinear optimization process can be avoided and therefore the global solution of the mixed integer nonlinear problem (25.1) is obtained.

5. Numerical Results - 100 Academic Test Cases

The method described in the previous section is implemented as a code called MISQPN [**5**]. The performance of the solver MISQPN is compared to other MINLP solution methods on a test set of 100 academic test cases [**15**]. The following codes are based on the theoretical background outlined in the previous sections:

MISQPN [**5**]	-	SQP-based outer approximation method, successive solution of mixed-integer quadratic programs extended by linear outer approximation constraints, convergence guaranteed for convex problems
MISQP [**4**]	-	Mixed-integer SQP-based trust region method, convergence not guaranteed
MISQPOA [**12**]	-	Additional stabilization of MISQP by outer approximations, successive solution of mixed-integer nonlinear programs by MISQP, convergence guaranteed for convex problems
OA	-	Outer approximations, successive solution of nonlinear programs NLP(y) (25.6) by MISQP, convergence guaranteed for convex problems, modification of MISQPOA [**12**] (Skip Step 2 of Algorithm 2)
MINLPB4 [**11**]	-	Branch-and-bound method based on MISQP with branching subject to integer variables

The subsequent Table 1 shows the performance of the MINLP solvers. In the first column we specify the solver. The subsequent column *Optimal* describes the performance on our test set in terms of the solution quality, i.e., it shows how many problems were solved. The column labeled by *Time* informs about the average time in seconds needed for a single problem instance. In column *Func* we state the average number of function evaluations needed to find a solution for a single problem, while column *Grad* counts the average number of gradient evaluations per instance. In column *Equ. Func* we report the average number of equivalent function calls per test case, which includes the function calls needed to approximate gradients by forward differences. In the last column the average deviation from the optimal solution is reported, which indicates the quality of a local solution.

As we can see the new method MISQPN performs significantly better than the linear outer approximation method OA in terms of the solution quality. The performance in terms of the number of function evaluations or equivalent function calls for MISQPN is of the same order of magnitude compared to linear outer approximation.

MISQP is able to solve significantly more problems to global optimality and the number of function evaluations increases only moderately. The performance of the branch-and-bound solver MINLPB4 in terms of the solution quality is comparable to that of MISQP but a huge number of function evaluations and equivalent function calls are necessary to solve the problems.

MISQPOA calls MISQP within an outer approximation framework. As a consequence the obtained solution is at least as good as the one found by MISQP and obviously the number of function evaluations is significantly higher. For MISQPOA global optimality can be guaranteed for convex problems. MISQPOA is able to solve almost all problems to global optimality and compared to the cheap solvers the number of equivalent function calls raises by only a small factor especially in contrast to the branch-and-bound method.

6. Conclusions

It is well known that mixed-integer optimization problems are extremely difficult to solve, especially in the non-convex case. Nevertheless, we conclude from our numerical results that an efficient solution of nonlinear mixed-integer programs is possible.

The solution method introduced in section 3, that refers to MISQP, is based on a modified sequential quadratic programming (SQP) algorithm, where we successively solve mixed-integer quadratic programs. However, we cannot guarantee convergence of the mixed-integer SQP-type algorithm, even not in the convex case.

We thus extend the method by adding a mixed-integer linear master program with outer approximations, by which the SQP-type method is stabilized and convergence is guaranteed for convex problems. The method described in section 4 applies the outer approximation idea directly to the mixed-integer SQP methods, by combining the master program and the continuous solver for fixed integer variables.

On the one hand our new method increases the solution quality for non-convex problems significantly compared to our implementation of an outer approximation algorithm. On the other hand it is very efficient in terms of the number of function evaluations, the most important performance criterion for time-consuming simulation programs in practical applications.

Furthermore, MISQPN [**5**] is the first implementation of such an algorithm and the close relationship to the solver MISQP [**4**] indicates that a similar performance at least in terms of the solution quality should be possible.

Acknowledgments

Sponsored by Shell GameChanger, SIEP Rijswijk, under project number 4600003917.

References

1. C. Audet and E. J. Dennis. Pattern search algorithm for mixed variable programming. *Siam Journal on Optimization*, 11:573–594, 2001.
2. B. Borchers and E. J. Mitchell. An improved branch and bound algorithm for mixed integer nonlinear programming. *Computers and Operations Research*, 21(4):359–367, 1994.
3. A. M. Duran and E. I. Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming*, 36:307–339, 1986.
4. O. Exler, T. Lehmann, and K. Schittkowski. MISQP: A Fortran implementation of a trust region SQP algorithm for mixed-integer nonlinear programming - user's guide, version 2.2. Technical report, Department of Computer Science, University of Bayreuth, Germany, 2008.
5. O. Exler, T. Lehmann, and K. Schittkowski. MISQPN: A Fortran subroutine for mixed-integer nonlinear optimization by outer approximation supported by mixed-integer search steps - user's guide, version 1.0. Technical report, Department of Computer Science, University of Bayreuth, Germany, 2009.
6. O. Exler and K. Schittkowski. A trust region SQP Algorithm for mixed-integer nonlinear programming. *Optimization Letters*, 1(3):269–280, 2007.
7. R. Fletcher and S. Leyffer. Solving mixed integer nonlinear programs by outer approximation. *Mathematical Programming*, 66(1-3):327–349, 1994.
8. A. C. Floudas. *Nonlinear and mixed-integer optimization: Fundamentals and applications.* Topics in chemical engineering. Oxford Univ. Press, New York, NY, 1995.
9. E. I. Grossmann. Review of Nonlinear Mixed-Integer and Disjunctive Programming Techniques. *Optimization and Engineering*, 3:227–252, 2002.
10. K. O. Gupta and V. Ravindran. Branch and bound experiments in convex nonlinear integer programming. *Management Science*, 31:1533–1546, 1985.
11. T. Lehmann and K. Schittkowski. MINLPB4: A Fortran subroutine for mixed integer nonlinear optimization by branch and bound - user's guide, version 1.0. Technical report, Department of Computer Science, University of Bayreuth, Germany, 2009.
12. T. Lehmann and K. Schittkowski. MISQPOA: A Fortran subroutine for mixed integer nonlinear optimization by outer approximation - user's guide, version 1.0. Technical report, Department of Computer Science, University of Bayreuth, Germany, 2009.
13. S. Leyffer. Integrating SQP and branch-and-bound for mixed integer nonlinear programming. *Computational Optimization and Application*, 18:295–309, 2001.
14. L. H. Li and T. C. Chou. A global approach for nonlinear mixed discrete programming in design optimization. *Engineering Optimization*, 22:109–122, 1994.
15. K. Schittkowski. A collection of 100 test problems for nonlinear mixed-integer programming in Fortran - user's guide. Technical report, Department of Computer Science, University of Bayreuth, Germany, 2009.

16. T. Westerlund and R. Pörn. Solving pseudo-convex mixed integer optimization problems by cutting plane techniques. *Optimization and Engineering*, 3:253–280, 2002.
17. Y. Yuan. On the convergence of a new trust region algorithm. *Numerische Mathematik*, 70:515–539, 1995.

A branch-and-cut-and-price framework for convex MINLP applied to a stochastic network design problem

Bernard Fortz[1,2] **Martine Labbé** **Michael Poss**

Department of Computer Science
Université Libre de Bruxelles
Brussels, Belgium

{bfortz,mlabbe,mposs}@ulb.ac.be

[1] *also at* CORE, Université catholique de Louvain
[2] FRIA Research Fellow

Abstract

Many convex linearly constrained programs and mixed integer programs have a large number of variables, so that the variables should be generated dynamically throughout the solution algorithm. This yields to the well known "branch-and-price algorithm" and "simplicial decomposition". We present a novel "branch-and-cut-and-price algorithm" to extend this idea to certain classes of convex linearly constrained MINLP. Our algorithm incorporates the variables generation into the "LP/NLP algorithm" introduced by Quesada and Grossman. We detail our framework for the stochastic network design problem with simple recourse and present preliminary computational results.
Keywords: convex MINLP, branch-and-price, stochastic programming, network design.

1. Introduction

Many difficult Mixed Integer Programs can be solved efficiently by Dantzig-Wolfe decomposition, followed by a branch-and-price algorithm, see for instance [**10, 4**]. This reformulation has the following advantages: it reduces the number of constraints of the problem and it may provide a stronger bound than the linear relaxation of the problem. Consider for instance the fixed-charge capacitated network design problem. Its arcs-paths formulation contains much less constraints than its arcs-nodes formulation, both formulations having the same linear relaxation [**9**]. Since the number of paths is exponential in the size of the graph, we should rather generate them dynamically with a branch-and-price algorithm. Other problems, such as the unsplittable multi-commodity flows problem have a structure naturally well suited for column generation, see [**3**].

Independently, Dantzig-Wolfe decomposition has been successfully applied to linearly constrained problems with a pseudo-convex and differentiable objective, yielding the simplicial decomposition [**13**]. Again, this decomposition replaces the possibly complicated constraints by the simple constraints defining the canonical simplex, but require dynamic variable generation..

Up to our knowledge, no such decomposition has yet been applied to Mixed Integer Non Linear Programs, and in particular convex MINLP, although many efficient algorithms have been developed for convex MINLP, see [**7**] for a review. Herein, we reformulate a convex linearly constrained MINLP using Dantzig-Wolfe decomposition. Then, we present a novel branch-and-cut-and-price algorithm based on the NP/NLP algorithm first introduced in [**15**] and implemented in FilMINT [**2**] and Bonmin [**6**]. We are currently implementing our framework for a network design problem with uncertain demand and simple-recourse, using CPLEX 12.1 [**1**] as the LP solver. This framework can easily be extended to handle other convex MINLP, such as the unsplittable multi-commodity flows problem with convex cost, well studied in the splittable case [**12**] and in the linear cost case [**3**].

In the next section, we describe our stochastic network design problem with simple recourse, while our branch-and-cut-and-price algorithm is described in Section 3. Section 4 briefly describes two possible extensions and Section 5 presents preliminary computational results.

2. Oblivious network design with simple recourse

Given an undirected graph (V, E) and a set of demands $d^k \in \mathcal{K}$, with origin $s(k)$ and destination $t(k)$ for every $k \in \mathcal{K}$, the capacitated network design problem aims at installing the cheapest capacities so that the resulting network shall be able to attend each demand. However, in many practical situations, the demands are not known exactly when designing the network, which can be modeled by replacing d^k by continuous random variables, also noted d^k. Moreover, it is often desirable that the routing be planed before the demand is known exactly. Thus, given penalty factors π^k, we formulate the problem as a stochastic program with simple-recourse:

$$
\begin{array}{llll}
 & \min & \displaystyle\sum_{ij\in E} c_{ij}x_{ij} + \sum_{k\in\mathcal{K}} \pi^k \mathbb{E}\left[y^k(\omega)\right] & \\
(26.1) & \text{s.t.} & \displaystyle\sum_{j\in V\setminus i} (f^k_{ji} - f^k_{ij}) = \left\{\begin{array}{rl} d^k(\omega) + z^k(\omega) - y^k(\omega) & i = t(k) \\ -d^k(\omega) - z^k(\omega) + y^k(\omega) & i = s(k) \\ 0 & \mathit{otherwise} \end{array}\right. & \begin{array}{r} \omega \in \Omega, \\ i \in V, \\ k \in \mathcal{K} \end{array} \\
(AN) & & \displaystyle\sum_{k\in\mathcal{K}} \left(f^k_{ij} + f^k_{ji}\right) \leqslant C x_{ij} & ij \in E \\
 & & 0 \leqslant x_{ij} \leqslant \overline{x}_{ij} & ij \in E \\
 & & x \in \mathbb{Z} & \\
 & & f, y, z \geqslant 0, &
\end{array}
$$

where x_{ij} represents the number of batches of capacity with size C installed on edge ij and f^k_{ij} denotes the maximal amount of flow for commodity k that can go from i to j through edge $ij \in E$. Therefore, given the optimal solution (x^*, f^*, y^*) to (AN), demand $d^k(\omega)$ is fully attended only if $y^{k*}(\omega) = 0$. Let

$d^{k*} = \max_{\omega\in\Omega}\{d^k(\omega) \text{ s.t. } y^{k*}(\omega) = 0\}$. Hence, the actual flow on edge $ij \in E$ for commodity $k \in \mathcal{K}$ is equal to $f_{ij}^{k\,*} \min\left(1, \frac{d^k(\omega)}{d^{k\,*}}\right)$.

The simple-recourse structure of (AN) allows us to reformulate (AN) as a non linear problem, substituting $y^k(\omega)$ by $\max\left(0, d^k(\omega) - \sum_{p\in\mathcal{P}^k} f_p^k\right)$, see [**8**]. Then, applying a Dantzig-Wolfe decomposition to (26.1) for each commodity $k \in \mathcal{K}$, and introducing the continuous density probability functions $g^k : \mathbb{R} \to \Xi^k$ for random variable d^k, we obtain our arcs-paths formulation:

$$\begin{aligned}
(26.2)\quad & \min \quad \sum_{ij\in E} c_{ij}x_{ij} + \sum_{k\in\mathcal{K}} \pi^k \int_{\Xi^k} \max\left(0, \xi^k - \sum_{p\in\mathcal{P}^k} f_p^k\right) g^k(\xi^k)d\xi^k \\
(AP)\quad & \text{s.t.} \quad \sum_{k\in\mathcal{K}}\sum_{p\in\mathcal{P}^k} \delta_{ij}^p f_p^k \leqslant C_{ij}x_{ij} && ij \in E \\
& \qquad 0 \leqslant x_{ij} \leqslant \overline{x}_{ij} && ij \in E \\
& \qquad x \in \mathbb{Z} \\
& \qquad f \geqslant 0,
\end{aligned}$$

where $\mathcal{P}^k$ is the set of all paths in G between $s(k)$ and $t(k)$, $\mathcal{P} = \cup_{k\in\mathcal{K}}\mathcal{P}^k$, f_p^k the maximal flow on path p, and δ_{ij}^p is equal to one if $ij \in p$, 0 otherwise. Let $AP(P)$ be (AP) restricted to paths in $P \subseteq \mathcal{P}$. It is well known that (26.2) is convex when each g^k has finite second moments, see [**5**]. Moreover, it is differentiable because each non linear term of (26.2) can be rewritten

$$h^k(f) = \pi^k \int_{\Xi^k:\xi^k \geqslant \sum_{p\in\mathcal{P}^k} f_p^k} \left(\xi^k - \sum_{p\in\mathcal{P}^k} f_p^k\right) g^k(\xi^k)d\xi^k,$$

and $\sum_{p\in\mathcal{P}^k} f_p^k$ is differentiable. Remark that it may be difficult to compare fixed costs c and operating costs π so that we could replace the term $\sum_{ij\in E} c_{ij}x_{ij}$ in (26.2) by a budget constraint $\sum_{ij\in E} c_{ij}x_{ij} \leqslant B$, see [**14**] for examples of network design problems with budget constraints.

3. Algorithm

Formulation (AN) has too many constraints to be solved for real size networks, and requires therefore to be tackled by a decomposition algorithm. However, (AP) considering all paths has too many variables, while only a few of them are required in the optimal solution. Hence, it would be interesting to generate paths only when needed. Once we have chosen a suitable MINLP framework, we must decide how to generate the paths throughout the solution algorithm. Herein, we decided to use the NP/NLP algorithm (remark that similar ideas could be used to couple together the MINLP branch-and-bound algorithm and the simplicial decomposition). Since (AP) has only linear constraints, we do not need to solve feasibility NLP so that the NP/NLP turns out to be a branch-and-cut algorithm. Namely we define a master problem, see (MP) below, accumulating the linearizations of the objective function. Additional linearizations are generated throughout the branch-and-bound algorithm solving (MP), see [**7**]. Although linearizations must be added at each integer node (to test whether we keep the associated incumbent), it is not obvious whether to add them at each fractional node. On one hand, it is important to add enough linearizations early in the tree to avoid exploration of too many infeasible nodes. On the other hand, adding too many unnecessary cuts would slow down the linear relaxation at each node. Our algorithm `B&P-NP/NLP-n` adds linearizations at integer nodes and nodes with a depth less than or equal to a given parameter n.

Let S^k and $P^k \subseteq \mathcal{P}^k$ be the sets of linearizations of h^k and paths variables generated so far for commodity k, respectively, and E^0 and E^1 the sets of edges fixed to 0 and 1, respectively, in the branching constraints. We define now the master problem for $|S^k|$ linearizations of h^k and $|P^k|$ path variables for each commodity k, and $|E^0| + |E^1|$ branching constraints:

$$\begin{array}{llll}
 & \min & \sum_{ij\in E} c_{ij}x_{ij} + \sum_{k\in\mathcal{K}} \theta^k & \\
(26.3) & \text{s.t.} & h^k\left(\overline{f}^s\right) + \sum_{p\in P^k} \frac{\partial h^k}{\partial f_p^k}\left(\overline{f}^s\right)\left(f_p^k - \overline{f}_p^k\right) \leqslant \theta^k & k\in\mathcal{K},\ s\in S^k \\
(26.4) & (MP) & \sum_{k\in\mathcal{K}}\sum_{p\in P^k} \delta_{ij}^p f_p^k \leqslant C_{ij}x_{ij} & ij\in E \\
 & & 0 \leqslant x_{ij} \leqslant \overline{x}_{ij} & ij\in E \\
 & & x_{ij} = 1 & ij\in E^1 \\
 & & x_{ij} = 0 & ij\in E^0 \\
 & & 0 \leqslant x \leqslant 1 & \\
 & & f \geqslant 0. &
\end{array}$$

Note that each h^k satisfies the following property:

$$\frac{\partial h^k}{\partial f_p^k} = \frac{\partial h^k}{\partial f_q^k} \text{ for any } p, q \in \mathcal{P}^k, \tag{26.5}$$

so that we denote (26.5) by $\frac{dh^k}{df}$ in the following. Property (26.5) implies that the reduced cost is easy to compute for any path $p \in \mathcal{P}$.

Lemma 3.1. Let (x^*, f^*) be an optimal solution to (MP), λ^* and μ^* be optimal multipliers associated with constraints (26.3) and (26.4), respectively, and $p \in \mathcal{P}^k$ for some $k \in \mathcal{K}$. The reduced cost of f_p^k is equal to:

$$c_p^{k*} = -\sum_{s\in S^k} \frac{dh^k}{df}\left(f^{s*}\right)\lambda^{s*} - \sum_{ij\in E} \delta_{ij}^p \mu_{ij}^*. \tag{26.6}$$

Since for each k, the piece-wise linear function defined by (26.3) is always smaller to or equal than h^k, we can use c_p^{k*} to know whether new paths can actually improve the objective.

Lemma 3.2. Consider some path $p \in \mathcal{P}\backslash P$ and let $h(x^*, f^*)$ and $h(x', f')$ be the optimal solution of $AP(P)$ and $AP(P \cup p)$, respectively. If $c_p^* \geqslant 0$, then $h(x^*, f^*) = h(x', f')$.

Note that the term $\sum_{s\in S^k} \frac{dh^k}{df}\left(f^{s*}\right)\lambda^{s*}$ depends only on commodity $k \in \mathcal{K}$, not on path p. Thus, the pricing problem turns out to be a shortest path problem for each commodity, with edge costs μ_{ij}^*. This is a well known problem polynomially solvable that we solve by linear programming.

Our algorithm is described on Algorithm 1. Numerical results for Algorithm 1 shall be available very shortly.

```
1  begin                                                    /* Initialization */
2      $T = \{o\}$ where $o$ has no branching constraints;
3      $UB = +\infty$;
4      cut = true;
5      var = true;
6  end
7  while $T$ is nonempty do
8      select a node $o' \in T$;
9      $T \leftarrow T \backslash \{o'\}$;
10     while cut = true or var = true do
11         solve $o'$;
12         let $(\theta^*, x^*, f^*)$ be an optimal solution;
13         let $(\lambda^*, \mu^*)$ be optimal dual multipliers;
14         cut = false, var = false;
15         if $\theta^* < UB$ then
16             if $x$ is integer or $depth(o') \leqslant n$ then
17                 foreach $k \in \mathcal{K}$ do
18                     if $\theta^{k*} < h^k(f^*)$ then
19                         add a cut (26.3) to $S^k$;
20                         cut = true;
21                 if cut = false then
22                     foreach $k \in \mathcal{K}$ do
23                         let $p$ be the shortest paths between $s(k)$ and $t(k)$
                           according to costs $\mu^*_{ij}$;
24                         if $c^{k*}_p < 0$ then
25                             add $p$ to $P^k$;
26                             var = true;
27     if $x$ is integer then define a new upper bound $UB := \theta^*$ and save current
       incumbent;
28     else
29         branch, resulting in nodes $o^*$ and $o^{**}$;
30         $T \leftarrow T \cup \{o^*, o^{**}\}$;
```

Algorithm 1: `B&P-NP/NLP-n`

4. Extensions

The previous scheme extends easily to more general convex and differentiable objective functions $h(x, f)$, as long as we can easily compute the reduced costs for paths in $\mathcal{P} \backslash P$. For instance, if h is separable for each edge,

$$h(x, f) = \sum_{ij \in E} h_{ij} \left(x, \sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}^k} \delta^p_{ij} f^k_p \right), \tag{26.7}$$

the reduced cost for path $p \in \mathcal{P}^k$ simply becomes

$$c_p^{k*} = -\sum_{ij \in E} \delta_{ij}^p \left(\mu_{ij}^* + \sum_{s \in S} \frac{\partial h}{\partial f_p^k} (f^{s*}) \lambda^{s*} \right),$$

when $|S|$ linearizations have been computed for h. Note that various realistic routing costs have structure (26.7), see for example $h_{ij}(f) = \sum_{k \in \mathcal{K}} \frac{1}{2}(q_{ij}^k (f_{ij}^k)^2 + c_{ij} f_{ij}^k)$, and $h_{ij}(f) = \frac{\sum_{k \in \mathcal{K}} f_{ij}^k}{C_{ij} - \sum_{k \in \mathcal{K}} f_{ij}^k}$ taken from [**11**], with $f_{ij}^k = \sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}^k} \delta_{ij}^p f_p^k$.

Algorithm 1 is an "easy" branch-and-cut-and-price algorithm in the sense that we do not branch on the variables that are generated dynamically: we branch on the x variables and generate the f variables. To extend Algorithm 1 to the unsplittable multi-commodity flows problem with convex cost, we should use a more sophisticated branching procedure. We could for instance, adapt the one used in [**3**].

5. Preliminary results

We detail next preliminary results obtained for a randomly generated graph with 50 nodes and 100 edges, with four batches of capacity allowed on each edge and $C = 100$. The number of end-to-end commodities varies between 10 and 20. They follow Gaussian distributions with means and variances uniformly distributed between 1 and 40, and 0 and 9, respectively. Costs c are based on euclidean distances between nodes. All algorithms are coded in JAVA on a HP Compaq 6510b with a processor Intel Core 2 Duo of 2.40 GHz and 2 GB of RAM memory.

The solution times for the subsequent algorithms are shown on Table 1 below.

- `CPLEX`: Formulation (AN) solved by the branch-and-cut framework from CPLEX 12.1; cuts (26.3) are implemented through the *LazyConstraint-Callback*. We check for violated cut at every node of the tree.
- `B&C`: Formulation (AN) solved by a branch-and-cut algorithm (similar to algorithm 1 without the column generation) fully implemented in JAVA using CPLEX 12.1 as the LP solver. We check for violated cut at every node of the tree.
- `B&C&P`: Formulation (AP) solved by algorithm 1 fully implemented in JAVA using CPLEX 12.1 as the LP solver. We check for violated cut and missing path at every node of the tree.

Table 1. CPU times.

	Total time			(MP) time		(MP) and pricing time	time ratio
$\|\mathcal{K}\|$	CPLEX	B&C	B&C&P	B&C	B&C&P	B&C&P	B&C/B&C&P
10	4.5	3.1	2.7	2.3	0.93	1.56	1.1
12	13.7	24.6	15.8	21.2	9	11.4	1.6
14	64	150	82	134	55	64.5	1.8
16	43	171	101	155	69	81	1.7
18	100	581	303	523	205	238	1.9
20	100	1602	742	1455	510	585	2.2

Columns "Total time" present the total amount of CPU time required by each algorithm to solve the problem, columns "(MP) time" provide only the time spent for solving the bounding problem with CPLEX, and column "(MP) and pricing time" sums the times for solving the bounding problem and for pricing out a new variable. Finally, column "time ratio" provides the ratios between total times of `B&C` and `B&C&P`.

Apart from the easiest instance, `CPLEX` is always faster than both `B&C&P` and `B&C`, which was expected because `CPLEX` uses the powerful heuristics and cutting planes from the MIP solver of CPLEX 12.1 while `B&C&P` and `B&C` follow simple branch-and-bound schemes. Then, `B&C&P` is always faster than `B&C` and the ratio rises with the difficulty of the problems.

References

1. IBM-ILOG Cplex, 2009. `http://www.ilog.com/products/cplex/`.
2. K. Abhishek, S. Leyffer, and J. T. Linderoth. Filmint: An outer-approximation-based solver for nonlinear mixed integer programs. Argonne National Laboratory, Mathematics and Computer Science Division, Argonne, IL., 2008.
3. C. Barnhart, C. A. Hane, and P. H. Vance. Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Oper. Res.*, 48(2):318–326, 2000.
4. C. Barnhart, E. L. Johnson, G. L. Nemhauser, Martin W. P. Savelsbergh, and Pamela H. Vance. Branch-and-price: column generation for solving huge integer programs. *Oper. Res.*, 46(3):316–329, 1998.
5. J. R. Birge and F. V. Louveaux. *Introduction to Stochastic programming (2nd edition)*. Springer Verlag, New-York, 2008.
6. P. Bonami, L. T. Biegler, A. R. Conn, G. Cornuéjols, I. E. Grossmann, C. D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, and A. Wächter. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5(2):186–204, 2008.
7. P. Bonami, M. Kilinc, and J. Linderoth. Algorithms and software for convex mixed integer nonlinear programs. Technical Report 1664, Computer Sciences Department, University of Wisconsin-Madison, 2009.
8. B. Fortz, M. Labbé, F. V. Louveaux, and M. Poss. The knapsack problem with gaussian weights. Technical Report 592, GOM, Université Libre de Bruxelles, 2009. `http://www.ulb.ac.be//di/gom/publications/technical/2009/Fortz-Labbe-Louveaux-Poss.html`.
9. A. Lisser, A. Ouorou, J.-P. Vial, and J. Gondzio. Capacity planning under uncertain demand in telecommunication networks. Technical report, 1999.
10. M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Oper. Res.*, 53(6):1007–1023, 2005.
11. A. Ouorou. A proximal subgradient projection algorithm for linearly constrained strictly convex problems. *Optimization Methods and Software*, 22(4):617–636, August 2007.
12. A. Ouorou, P. Mahey, and J.-P. Vial. A survey of algorithms for convex multicommodity flow problems. *Management Science*, 46(1):126–147, 2000.
13. M. Patriksson. Simplicial decomposition algorithms. In *Encyclopedia of Optimization*, pages 3579–3585. 2009.

14. M. Pióro and D. Medhi. *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.
15. I. Quesada and I. E. Grossman. An LP/NLP based branch and bound algorithm for convex MINLP optimization problems. *Comput. Chem. Eng.*, 16(10/11):937–947, 1992.

Projected Perspective Reformulations for MIQP problems

Antonio Frangioni[1] **Claudio Gentile**[2] **Enrico Grande**[3] **Andrea Pacifici**[3]

[1] Dipartimento di Informatica, Università di Pisa
Polo Universitario della Spezia, Via dei Colli 90,
19121 La Spezia, Italy
`frangio@di.unipi.it`

[2] IASI-CNR
Viale Manzoni 30,
00185 Roma, Italy
`gentile@iasi.cnr.it`

[3] Dipartimento di Ingegneria dell'Impresa,
Università degli Studi di Roma "Tor Vergata"
via del Politecnico 1, 00133 Rome – Italy
`grande@disp.uniroma2.it,pacifici@disp.uniroma2.it`

Abstract

The *Perspective Relaxation* (PR) is a general approach for constructing tight approximations to Mixed-Integer NonLinear Problems with semicontinuous variables. The PR of a MINLP can be formulated either as a Mixed-Integer Second-Order Cone Program, provided that the original objective function is SOCP-representable, or as a Semi-Infinite MINLP. We show that under some further assumptions—rather restrictive, but satisfied in several practical applications—the PR of Mixed-Integer Quadratic Programs can also be reformulated as a piecewise linear-quadratic problem of roughly the same size of the standard continuous relaxation. Furthermore, if the original problem has some exploitable structure, this is typically preserved in the reformulation, allowing to construct specialized approaches for solving the PR. We report on implementing these ideas on two MIQPs with appropriate structure: a sensor placement problem and a Quadratic-cost (single-commodity) network design problem.

Keywords: mixed-integer quadratic problems, perspective relaxation.

1. Introduction

Semi-continuous variables are very often found in models of real-world problems such as distribution and production planning problems [**7, 10**], financial trading and planning problems [**8**], and many others [**1, 11, 12**]. These are variables which are constrained to either assume the value 0, or to lie in some given polyhedron $\mathcal{P}$; when 0 belongs to $\mathcal{P}$, one incurs in a *fixed cost* to allow the variable to have a nonzero value. We will consider Mixed-Integer NonLinear Programs (MINLP) with n semi-continuous variables $x_i \in \mathbb{R}^{m_i}$ for each $i \in N = \{1, \dots, n\}$. Assuming that each $\mathcal{P}_i = \{x_i : A_i x_i \leqslant b_i\}$ is compact, and therefore $\{x_i : A_i x_i \leqslant 0\} = \{0\}$, each x_i can be modeled by using an associated binary variable y_i, leading to problems of the form

$$\text{(27.1)} \qquad \min \qquad g(z) + \textstyle\sum_{i \in N} f_i(x_i) + c_i y_i$$

$$\text{(27.2)} \qquad A_i x_i \leqslant b_i y_i \qquad i \in N$$

$$\text{(27.3)} \qquad (x, y, z) \in \mathcal{O} \quad , \quad y \in \{0,1\}^n \quad , \quad x \in \mathbb{R}^m \quad , \quad z \in \mathbb{R}^q$$

where all f_i and g are closed convex functions, z is the vector of all the "other" variables, and $\mathcal{O} \subseteq \mathbb{R}^{m+n+q}$ (with $m = \sum_{i \in N} m_i$) represents all the "other" constraints of the problem.

It is known that the convex hull of a (possibly disconnected) domain such as $\{0\} \cup \mathcal{P}$ can be conveniently represented in a higher-dimensional space, which allows to derive *disjunctive cuts* for the problem [**14**]; this leads to defining the *Perspective Reformulation* of (27.1)—(27.3) [**5, 7**]

$$\text{(27.4)} \qquad \min \left\{ \; g(z) + \textstyle\sum_{i \in N} y_i f_i(x_i / y_i) + c_i y_i \;\; : \;\; (27.2) \; , \; (27.3) \; \right\}$$

whose continuous relaxation is significantly stronger than that of (27.1)—(27.3), and that therefore is a more convenient starting point to develop exact and approximate solution algorithms [**7, 8, 10, 12**]. However, an issue with (27.4) is the high nonlinearity in the objective function due to the added fractional term. Two alternative reformulations of (27.4) have been proposed: one as a Mixed-Integer Second-Order Cone Program [**15, 3, 12**] (provided that the original objective function is SOCP-representable), and the other as a Semi-Infinite MILP [**7**]. In several cases, the latter outperforms the former in the context of exact or approximate enumerative solution approaches [**9**], basically due to the much higher reoptimization efficiency of active-set (simplex-like) methods for Linear and Quadratic Programs w.r.t. the available Interior Point methods for Conic Programs. However, both reformulations of (27.4) require the solution of substantially more complex continuous relaxations than the original formulation of (27.1)—(27.3); furthermore, they may spoil the valuable structure of the problem, such as the presence of network constraints. We show that, under some further assumptions, the PR of a Mixed-Integer *Quadratic* Program can also be reformulated as a piecewise linear-quadratic problem of roughly the same size of the standard continuous relaxation; this new reformulation is obtained by projecting each pair of variables (x_i, y_i) onto the subspace of the variables x_i, as discussed in Section 2. Moreover, if the original problem has some exploitable structure, then this structure is preserved in the reformulation, thus allowing to construct specialized approaches for solving the PR. We apply this approach to a Sensor Placement problem (Section 3) and to a Quadratic-cost (single-commodity) network design problem (Section 4), reporting numerical experiments comparing state-of-the-art, off-the-shelf MIQP solvers with the new specialized solution approach (Section 5).

2. A piecewise description of the convex envelope

Here we analyze the properties of the Perspective Reformulation under three further assumptions on the data of the original problem (27.1)—(27.3):

A1) each x_i is a *single variable* (i.e., $m_i = 1$) and each $\mathcal{P}_i$ is a bounded real interval $[0, u_i]$;

A2) the variables y_i *only* appear each in the corresponding constraint (27.2), i.e., the "other" constraints $\mathcal{O}$ do not concern the y_i;

A3) all functions are *quadratic*, i.e., $f_i(x_i) = a_i x_i^2 + b_i x_i$ (and since they are convex, $a_i \geqslant 0$).

While these assumptions are indeed restricting, they are in fact satisfied by most of the applications of the PR reported so far [**7, 8, 11, 3, 12**]. Since in this paragraph we will only work with *one* block at a time, to simplify the notation in the following we will drop the index "i". We will therefore consider the (fragment of) Mixed-Integer Quadratic Program (MIQP)

$$\min \left\{ ax^2 + bx + cy \ : \ 0 \leqslant x \leqslant uy \ , \ y \in \{0,1\} \right\} \tag{27.5}$$

and its Perspective Relaxation

$$\min \left\{ f(x,y) = (1/y)ax^2 + bx + cy \ : \ 0 \leqslant x \leqslant uy \ , \ y \in \{0,1\} \right\} \ . \tag{27.6}$$

The basic idea behind the approach is to recast (27.6) as the minimization over $x \in [0,u]$ of

(27.7)

$$z(x) = \min_y f(x,y) = bx + \min_y \left\{ (1/y)ax^2 + cy \ : \ 0 \leqslant x \leqslant uy \ , \ y \in [0,1] \right\} \ .$$

It is well-known that $z(x)$ (partial minimization of a convex function) is convex; furthermore, due to the specific structure of the problem $z(x)$ can be algebraically characterized. In particular, due to convexity of $f(x,y)$, the optimal solution $y^*(x)$ of the inner optimization problem in (27.7) is easily obtained by the solution $\tilde{y} = \tilde{y}(x)$ (if any) of the first-order optimality conditions of the unconstrained version of the problem, i.e., $\partial f(x,y)/\partial y = c - ax^2/y^2 = 0$. In fact, if $\tilde{y}$ is feasible for the problem, then it is optimal ($y^*(x) = \tilde{y}$); otherwise, $y^*(x)$ is the projection of $\tilde{y}$ over the feasible region, i.e., the extreme of the interval nearer to $\tilde{y}$ (this is where assumption A1 is used). Thus, by developing the different cases, one can construct an explicit algebraic description of $z(x) = f(x, y^*(x))$.

We start by rewriting the constraints in (27.7) as

$$(0 \leqslant)\ x/u \leqslant y \leqslant 1 \tag{27.8}$$

(since $u \geqslant x \geqslant 0 \Rightarrow x/u \geqslant 0$). We must now proceed by cases:

1) If $c \leqslant 0$, then $\tilde{y}$ is undefined: the derivative is always negative. Thus, there is no global minima of the unconstrained problem, and therefore $y^*(x) = 1$, yielding

$$z(x) = ax^2 + bx + c \tag{27.9}$$

2) Instead, if $c > 0$ then $\tilde{y} = x\sqrt{a/c}$ (note that we have used $x \geqslant 0$, $c > 0, a \geqslant 0$). In general, two cases can arise:

2.1) $\tilde{y} \leqslant x/u \Leftrightarrow u \leqslant \sqrt{c/a} \Leftrightarrow y^*(x) = x/u \Rightarrow$

$$z(x) = \big(\, b + au + c/u \,\big)x \tag{27.10}$$

2.2) $0 \geqslant \tilde{y} \geqslant x/u \Leftrightarrow u \geqslant \sqrt{c/a}$. This gives two further subcases

$*$ $(u \geqslant)\; x \geqslant \sqrt{c/a}\; (\geqslant 0) \Rightarrow \tilde{y} \geqslant 1 \Rightarrow y^*(x) = 1$,

$*$ $0 \leqslant x \leqslant \sqrt{c/a}\; (\leqslant u) \Rightarrow \tilde{y} \leqslant 1 \Rightarrow y^*(x) = \tilde{y}$,

finally showing that $z(x)$ is the piecewise linear-quadratic function

$$z(x) = \begin{cases} \big(\, b + 2\sqrt{ac} \,\big)x & \text{if } 0 \leqslant x \leqslant \sqrt{c/a} \\ ax^2 + bx + c & \text{if } \sqrt{c/a} \leqslant x \leqslant u. \end{cases} \tag{27.11}$$

Note that (27.11) is continuous and differentiable even at the (potential) breakpoint $x = \sqrt{c/a}$, and therefore convex (as expected).

In all the cases, $z(x)$ is a convex differentiable piecewise-quadratic function with at most 2 pieces.

3. A sensor placement problem

Consider the problem of optimally placing a set $N = \{1, \dots, n\}$ of sensors to cover a given area, where deploying one sensor has a fixed cost plus a cost that is quadratic in the radius of the surface covered [**1**]. The problem, which is shown to be $\mathcal{NP}$-hard in [**2**], can be written as

$$\min \Big\{ \textstyle\sum_{i \in N} c_i y_i + \sum_{i \in N} a_i x_i^2 \;:\; \sum_{i \in N} x_i = 1, 0 \leqslant x_i \leqslant y_i, y_i \in \{0,1\} \quad i \in N \Big\} \tag{27.12}$$

Since we can assume $c_i > 0$ (for otherwise y_i can surely be fixed to 1), in the continuous relaxation of this problem the "design" variables y_i can be *projected* onto the x_i; that is, the y_i variables can be eliminated since at optimality $y_i = x_i$. Such a problem can be solved in $O(n \log n)$ by Lagrangian relaxation [**1**]; however, the bound can be weak, yielding to a large number of nodes in the enumeration tree and to a large computational time. We can improve on the bound by using the convex envelope of the single blocks of the objective function; as outlined in Section 2, we can compute this bound by means of a *single* minimization involving the piecewise-linear-quadratic functions (27.10)-(27.11). Hence, we can rewrite the problem in the form

$$\min \Big\{ \textstyle\sum_{j=1}^m b_j \chi_j + \sum_{j=1}^m d_j \chi_j^2 \;:\; \sum_{j=1}^m \chi_j = 1 \;\;,\;\; \chi_j \in [0, \alpha_j] \quad j = 1, \dots, m \Big\} \tag{27.13}$$

where $m \leqslant 2n$ and the coefficients b_j and d_j are as follows:

- if $\sqrt{c_i/a_i} \geqslant 1$ then only one new variable χ_j is generated with coefficients $b_j = a_i u_i + c_i/u_i$, $d_j = 0$, and $\alpha_j = u_i$;
- if $\sqrt{c_i/a_i} < 1$ then two new variables χ_{j_1} and χ_{j_2} are generated such that $x_i = \chi_{j_1} + \chi_{j_2}$ with $b_{j_1} = 2\sqrt{a_i c_i}$, $d_{j_1} = 0$, $\alpha_{j_1} = \sqrt{c_i/a_i}$ for the first variable and $b_{j_1} = 2\sqrt{a_i c_i}$, $d_{j_1} = a_i$, $\alpha_{j_1} = 1 - \sqrt{c_i/a_i}$ for the second variable.

This problem can be easily solved in $O(m \log m) = O(n \log n)$ with the same algorithm mentioned for the continuous relaxation of (27.12).

4. Quadratic-cost network design

A directed graph $G = (N, A)$ is given; for each node $i \in N$ a deficit $b_i \in \mathbb{R}$ is given indicating the amount of flow that the node demands (negative deficits indicate source nodes). Each arc $(i, j) \in A$ can be used up to a given maximum capacity u_{ij} paying a fixed cost c_{ij}. Otherwise, no cost is due if (i, j) is not installed but flow cannot pass through the arc. Additionally, if x_{ij} units of flow are sent through an installed arc (i, j), a quadratic flow cost $b_{ij}x_{ij} + a_{ij}x_{ij}^2$ is also incurred. The problem is to decide which arcs to install and how to route the flow in such a way that demands are satisfied and the total (installing + routing) cost is minimized. The problem can be written as

$$\text{(27.14)} \qquad \begin{array}{lll} \min & \sum_{(i,j)\in A} c_{ij}y_{ij} + b_{ij}x_{ij} + a_{ij}x_{ij}^2 & \\ & \sum_{(j,i)\in A} x_{ji} - \sum_{(i,j)\in A} x_{ij} = b_i & i \in N \\ & 0 \leqslant x_{ij} \leqslant u_{ij}y_{ij} \;\;,\;\; y_{ij} \in \{0,1\} & (i,j) \in A \end{array}$$

This network design problem is $\mathcal{NP}$-hard, since it is a generalization of the sensor placement problem described in Section 3. A recent application of this general model in a Facility Location setting is given in [**11, 12**].

Again, since $c_{ij} > 0$ (for otherwise y_{ij} can surely be fixed to 1), in the continuous relaxation of (27.14) the design variables y_{ij} can be projected onto the x_{ij}; that is, at optimality $y_{ij} = x_{ij}/u_{ij}$. The resulting problem can be efficiently solved by means of (convex) Quadratic Min-Cost Flow (QMCF) algorithms; however, the bound provided by the continuous relaxation is usually weak.

Applying the results of Section 2 to (27.14), a *Separable Convex-cost NonLinear MCF* problem is obtained, where the flow cost function on each arc is a piecewise linear-quadratic convex cost function. In turn, this can be rewritten as a QMCF problem

$$\text{(27.15)} \qquad \begin{array}{lll} \min & \sum_{(i,j)\in A'} b'_{ij}\chi_{ij} + a'_{ij}\chi_{ij}^2 & \\ & \sum_{(j,i)\in A'} \chi_{ji} - \sum_{(i,j)\in A'} \chi_{ij} = b_i & i \in N \\ & 0 \leqslant \chi_{ij} \leqslant u'_{ij} & (i,j) \in A' \end{array}$$

on a graph $G' = (N, A')$ with the same node set and *at most* 2 times the number of arcs. For each of the original arcs (i, j), at most two "parallel" copies are constructed. If $u_{ij} \leqslant \sqrt{c_{ij}/a_{ij}}$ (case 2.1), then only one representative of (i, j) is constructed in G', with $b'_{ij} = b_{ij} + a_{ij}u_{ij} + c_{ij}/u_{ij}$, $a'_{ij} = 0$ and $u'_{ij} = u_{ij}$. Instead, if $u_{ij} \geqslant \sqrt{c_{ij}/a_{ij}}$ (case 2.2) then two parallel copies of the arc (i, j) have to be constructed in G': the first has $b'_{ij} = b_{ij} + 2\sqrt{a_{ij}c_{ij}}$, $a'_{ij} = 0$, and $u'_{ij} = \sqrt{c_{ij}/a_{ij}}$, while the second has $b'_{ij} = b_{ij} + 2\sqrt{a_{ij}c_{ij}}$, $a'_{ij} = a_{ij}$, and $u'_{ij} = u_{ij} - \sqrt{c_{ij}/a_{ij}}$. For this kind of "partitioned" NonLinear MCF problems—where some of the arcs have strictly convex cost functions, while the other have linear cost functions—specialized algorithms have been proposed [**6**]. In general, any algorithm for Convex (Quadratic) MCF problems (e.g., [**4**]) can be used. While codes implementing these algorithms are either not available or not very efficient in practice, the off-the-shelf solver `Cplex` turns out to be quite efficient in solving these convex QMCFs.

5. Computational Results

In order to assess the behaviour of the Projected Perspective Reformulation technique we implemented it on the two problems discussed in sections 3 and 4 within a specialized B&B where the perspective relaxation is solved by computing the projection $z(p)$ as in (27.10)-(27.11). We considered the reformulations (27.13) and (27.15) and, for their solution, we applied the specialized $O(n \log n)$ algorithm for the Sensor Placement problem and the `Cplex` quadratic solver, respectively. We compared the new approach (denoted as P^2/R) against the following ones:

- a B&C on the PR (27.6) using the Semi-Infinite MILP formulation (denoted as P/C for Perspective Cut method);
- a B&C on the PR (27.6) using the MI-SOCP formulation (denoted as CPLEX-SOCP);
- a standard B&C on the continuous relaxation (27.5) (denoted as CPLEX).

These three alternative methods have all been implemented by means of `Cplex` B&C solver. In particular, the P/C method has been coded with a `cut-callback` function. All the algorithms have been coded in `C++`, compiled with GNU `g++ 4.0.1` (with -O3 optimization option) and ran on an Opteron 246 (2 GHz) computer with 2 Gb of RAM, under Linux Fedora Core 3.

We generated 180 random instances of the Sensor Placement problem, grouped in 6 classes with 30 instances each. The first 4 classes contain instances with either 2000 or 3000 sensors and have either high or low quadratic costs. In the former ("h") , fixed costs are uniformly chosen in the interval $[1, n]$ while quadratic costs are uniformly chosen in the interval $[n, C_{max}]$, where $C_{max} \in \{10n, 20n, 30n\}$. In the latter ("l"), fixed costs are randomly generated in the interval $[n, B_{max}]$, where $B_{max} \in \{10n, 20n, 30n\}$, while quadratic costs are randomly generated in the interval $[1, n]$. The last two classes are generated starting from random instances of the PARTITION problem, according to the NP-hardness proof for the Sensor Placement problem in [**2**]. We considered 2000 and 3000 PARTITION items ranging in the intervals [100,1000], [500,1000], [1,100000]. Table 1 reports the obtained results.

name	P^2/R			CPLEX			
	time	nodes	av. t/n	time	nodes	av. t/n	gap
2000-h	0.39	1	0.39	1020.51	223293	0.01	4.03
2000-l	0.09	1	0.09	101.58	3713	0.03	0.00
3000-h	0.92	1	0.92	1057.09	144406	0.01	7.18
3000-l	0.21	1	0.21	270.49	5724	0.05	0.00
PTN-2000	0.43	1	0.43	1018.13	4149	0.25	2.98
PTN-3000	1.02	1	1.02	1008.42	568	1.79	3.14
name	P/C			CPLEX - SOCP			
	time	nodes	av. t/n	time	nodes	av. t/n	gap
2000-h	47.74	924	30.43	1066.02	507	2.11	207.04
2000-l	17.02	1	17.02	49.32	38	7.60	0.00
3000-h	91.24	88	74.09	1069.73	332	3.24	412.54
3000-l	40.27	1	40.27	135.95	72	12.08	0.00
PTN-2000	94.30	6	56.93	23.79	1	23.80	0.00
PTN-3000	202.63	6	114.72	53.74	1	53.74	0.00

Table 1. Results for the Sensor Placement problem

For the Network Design Problem we generated 360 problems, grouped into 12 classes with 30 instances each, as follows:

- the underlying flow networks with 1000, 2000, or 3000 nodes have been generated by `netgen` [**13**], where: (*i*) the minimum arc cost is 1 and the maximum is randomly generated between 10 and 100, (*ii*) the total supply b_s is randomly generated between 100 and 1000, and (*iii*) the minimum arc capacity is $0.05b_s$ and the maximum arc capacity is randomly generated in the interval $[0.2b_s, 0.4b_s]$;
- the fixed costs which are either low or high with respect to the linear costs generated by `netgen`, i.e., c_{ij} is uniformly generated either in $[0.5b_{ij}, b_{ij}]$ ("l") or in $[3b_{ij}, 10b_{ij}]$ ("h");
- the quadratic costs which are either low or high with respect to the linear costs generated by `netgen`, i.e., a_{ij} is uniformly generated either in $[3b_{ij}, 10b_{ij}]$ ("l") or in $[100b_{ij}, 1000b_{ij}]$ ("h").

Table 2 reports the obtained results.

name	P^2/R			CPLEX			
	time	nodes	av. t/n	time	nodes	av. t/n	gap
1000-h-h	0.05	1	0.05	108.80	35630	0.28	0.00
1000-h-l	0.31	5	0.05	1037.63	324447	0.01	0.02
1000-l-h	0.05	1	0.05	163.67	46685	0.18	0.00
1000-l-l	0.32	5	0.05	1046.89	304305	0.01	0.01
2000-h-h	0.10	1	0.10	690.09	101868	0.11	0.00
2000-h-l	45.42	278	1.10	1031.75	141485	0.01	0.06
2000-l-h	0.09	1	0.09	858.22	131954	0.03	0.00
2000-l-l	8.78	63	0.10	1036.79	140877	0.01	0.04
3000-h-h	0.15	1	0.15	1041.96	88541	0.01	0.00
3000-h-l	71.02	269	0.17	1051.93	73591	0.01	0.12
3000-l-h	0.15	1	0.15	988.74	89209	0.12	0.00
3000-l-l	19.05	79	0.16	1062.45	85878	0.01	0.04
name	P/C			CPLEX - SOCP			
	time	nodes	av. t/n	time	nodes	av. t/n	gap
1000-h-h	17.03	3	10.14	967.30	26	62.86	0.01
1000-h-l	5.89	25	0.38	79.17	46	16.98	0.00
1000-l-h	8.89	4	4.60	620.77	21	38.62	0.00
1000-l-l	4.68	22	0.33	30.46	63	17.37	0.00
2000-h-h	57.09	7	13.84	895.70	8	207.60	0.01
2000-h-l	51.60	348	0.72	252.98	36	27.65	0.00
2000-l-h	42.3	6	16.57	525.35	9	63.35	0.00
2000-l-l	20.60	131	0.51	252.82	193	40.02	0.00
3000-h-h	117.30	11	18.90	564.41	2	407.97	0.01
3000-h-l	140.47	584	1.39	366.95	27	36.76	0.00
3000-l-h	101.18	12	12.01	372.16	4	89.53	0.01
3000-l-l	45.43	153	0.89	292.41	83	62.39	0.00

Table 2. Results for Network Design problems

For our experiments we fixed a time limit of 1000 seconds. All problems where solved at optimality within this time limit with the P^2/R and the P/C methods, therefore we do not report the gap at termination for them. For all methods, we report the running time in seconds, the number of B&B nodes and the average time for node. As expected from previous results [**7, 9**], the P/C method overcomes CPLEX B&C algorithm both with standard and SOCP formulations. However, the newly proposed P^2/R approach significantly overcomes the P/C method. This is mainly because of the much faster specialized solution methods used for the relaxations, which significantly reduces the effort required at each node. Furthermore, P/C approximates the true perspective relaxations by means of a finite number of cutting planes, thereby introducing some (small) approximation errors; these seem to cause the generation of more B&C nodes w.r.t. the "exact" solutions provided by P^2/R.

References

1. A. Agnetis, E. Grande, P.B. Mirchandani, and A. Pacifici. Covering a line segment with variable radius discs. *Computers & Operations Research*, 36(5):1423–1436, 2009.
2. A. Agnetis, E. Grande, and A. Pacifici. Demand allocation with latency cost functions. *CoRR*, abs/0810.1650, 2008.
3. S. Aktürk, A. Atamtürk, and S. Gürel. A strong conic quadratic reformulation for machine-job assignment with controllable processing times. *Operations Research Letters*, 37(3):187–191, 2009.
4. J. Castro and N. Nabona. An Implementation of Linear and Nonlinear Multicommodity Network Flows. *European J. of Operational Research*, 92:37–53, 1996.
5. S. Ceria and J. Soares. Convex programming for disjunctive convex optimization. *Mathematical Programming*, 86:595–614, 1999.
6. R. De Leone, R.R. Meyer, and A. Zakarian. A Partitioned ε-Relaxation Algorithm for Separable Convex Network Flow Problems. *Computational Optimization and Applications*, 12:107–126, 1999.
7. A. Frangioni and C. Gentile. Perspective Cuts for 0-1 Mixed Integer Programs. *Mathematical Programming*, 106(2):225–236, 2006.
8. A. Frangioni and C. Gentile. SDP Diagonalizations and Perspective Cuts for a Class of Nonseparable MIQP. *Operations Research Letters*, 35(2):181 – 185, 2007.
9. A. Frangioni and C. Gentile. A Computational Comparison of Reformulations of the Perspective Relaxation: SOCP vs. Cutting Planes. *Operations Research Letters*, 37(3):206–210, 2009.
10. A. Frangioni, C. Gentile, and F. Lacalandra. Tighter Appro-ximated MILP Formulations for Unit Commitment Problems. *IEEE Transactions on Power Systems*, 24(1):105–113, 2009.
11. O. Günlük, J. Lee, and R. Weismantel. MINLP Strengthening for Separable Convex Quadratic Transportation-Cost UFL. IBM Research Report RC24213, IBM Research Division, 2007.
12. O. Günlük and J. Linderoth. Perspective relaxation of MINLPs with indicator variables. In A. Lodi, A. Panconesi, and G. Rinaldi, editors, *Proceedings 13th IPCO*, volume 5035 of *Lect. N. Comp. Sc.*, pages 1–16, 2008.

13. D. Klingman, A. Napier, and J. Stutz. NETGEN: A program for generating large scale capacitated assignment, transportation, and minimum cost flow network problems. *Management Science*, pages 814–821, 1974.
14. R.A. Stubbs and S. Mehrotra. A branch-and-cut method for 0-1 mixed convex programming. *Mathematical Programming*, 86:515–532, 1999.
15. M. Tawarmalani and N.V. Sahinidis. Convex extensions and envelopes of lower semi-continuous functions. *Mathematical Programming*, 93:515–532, 2002.

The Convex Hull Relaxation for Nonlinear Integer Programs with Convex Objective and Linear Constraints

Monique Guignard[1] **Aykut Ahlatcioglu**[2]

[1] Operations and Information Management Department
The Wharton School
University of Pennyslvania
Philadelphia, PA, 19104, USA
guignard@wharton.upenn.edu

[2] Department of Economics
Princeton University
Princeton, NJ, 08544, USA
aahlatci@princeton.edu

Keywords: nonlinear integer programming, bounds, relaxation.

1. Introduction

In this paper we introduce a relaxation method for computing both a lower bound on the optimal value of a nonlinear integer minimization program (NLIP), and good integer feasible solutions. For a *linear* integer program (LIP), an optimal integer solution is also optimal over the convex hull of all integer feasible solutions, but this is not usually the case for NLIPs. Rather, the minimization over this convex hull yields a *relaxation* of the NLIP, which we will call the Convex Hull (CH) Relaxation. While we define this relaxation for arbitrary NLIPs, for computational reasons, we restrict our attention to *convex* minimization problems with *linear* constraints, and we show that the lower bound can then be computed using any version of simplicial decomposition, with sub-problems that have the same constraints as the NLIP, but with linear objective functions. If these are easier to solve than their nonlinear counterpart, as would be the case for instance for nonlinear 0-1 knapsack problems, the bound may be tight and relatively inexpensive to compute.

A side product of this procedure is the generation of feasible integer points, which provide a tight upper bound to the optimal value of the problem.

What makes this relaxation very special is that contrary to Lagrangean relaxation or to primal relaxation (Guignard, 1994), this relaxation does not dualize or treat separately any constraints. While for nonlinear integer problems, it cannot be directly compared with Lagrangean relaxation, it always provides a bound at least as good as any primal relaxation. However, if (1) the linear subproblems are difficult to solve, (2) the objective function is nonconvex, and/or (3) there are nonlinear constraints, then one has to consider using a primal relaxation instead. In other cases, this new approach appears very attractive.

In the paper, we first define the CH relaxation (CHR) in section 2, analyze the application of simplicial decomposition to the CHR problem in section 3, give details on the algorithm in section 4, and finally present some computational results.

2. Preliminaries and notation

Consider the following nonlinear integer program (NLIP)

$$\text{(NLIP)} \qquad \min_{x \in S} f(x)$$

where
$f(x)$ is a nonlinear convex function of x, a vector of $\mathbb{R}^n$,
$S = \{x \in Y : Ax \leqslant b\}$,
A is an $m \times n$ constraint matrix,
b is a resource vector in $\mathbb{R}^m$,
Y is a subset of $\mathbb{R}^n$ specifying integrality restrictions on x.

Definition 28.1. We define the Convex Hull Relaxation of (NLIP) to be

$$\text{(CHR)} \qquad \min_{x \in \mathrm{Co}(S)} f(x)$$

The problem (CHR) is not in general equivalent to (NLIP) when $f(x)$ is nonlinear, because an optimal solution of (CHR) may not be integer, and therefore not feasible for (NLIP). However, it is easy to see that (CHR) is indeed a relaxation to (NLIP).

This relaxation is a primal relaxation, in the x-space, and it is related to the primal relaxation for nonlinear integer problems introduced in Guignard (1994). It is actually a primal relaxation that does not "relax" any constraint. A similar idea was used independently by Albornoz in his 1998 unpublished dissertation.

The difficulty in solving (CHR) comes from the implicit formulation of the convex hull. However the idea of decomposing the problem into a sub-problem and a master-problem, first introduced by Frank & Wolfe (1956), and furthered by Von Hohenbalken with Simplicial Decomposition (1973), and Hearn et al. with Restricted Simplicial Decomposition (1987), does provide an efficient way to solve (CHR) to optimality, solving a sequence of linear integer problems and of essentially unconstrained nonlinear problems. Primal relaxations that also relax constraints require a more complicated scheme, such as that described in Contesse and Guignard (1996, 2007), which use an augmented Lagrangean approach, with simplicial decomposition used at each iteration. By contrast, here, due to the absence of relaxed constraints, only one call to simplicial decomposition is needed.

3. Applying simplicial decomposition to the CHR problem

3.1. Assumption

There are several assumptions which should be imposed to the (NIP) formulation in order for the simplicial decomposition technique to effectively solve (CHR) to optimality. These are:

(i) Compactness and convexity of the feasible region
(ii) Convexity of the objective function
(iii) Linearity of the constraint set.

3.2. Subproblem

The first part of the decomposition problem is the sub-problem, and can also be considered as feasible descent direction finding problem. Assume we are at a feasible point of (CHR), call it x^k. For the kth iteration of simplicial decomposition, we can find a feasible descent direction for $\mathrm{Co}\,\{\mathrm{Ax} \leq \mathrm{b}, \mathrm{x} \in \mathrm{X}\}$, a polyhedron, by solving the following problem

$$\text{(CHS)} \qquad \min_y \nabla f(x^k)^T \times (y - x^k) \ \text{ s.t. } \ y \in \mathrm{Co}\{Ax \leqslant b, x \in Y\}$$

We will call this the Convex Hull Sub-problem (CHS), and x^k a linearization point. Note that CHS is a linear program. Therefore unlike nonlinear problems, it has an equivalent integer program, which we will call Integer Program Subproblem (IPS)

$$\text{(IPS)} \qquad \min_y \nabla f(x^k)^T \times (y - x^k) \ \text{ s.t. } \ Ay \leqslant b, y \in Y.$$

For many types of integer programming problems, solving (IPS) is considerably easier than solving (NLIP). The solution to (IPS) will yield an extreme point of the convex hull, unless x^k is optimal for the convex hull relaxation (CHR) problem. Therefore, at each iteration we obtain a feasible point to the original (NLIP) problem. Convergence to the optimal solution will be discussed in section 4. If x^k is not optimal, we proceed to the master problem.

3.3. Master Problem

Consider the following nonlinear programming problem with one simple constraint, which we call the Master Problem (MP).

$$\text{(MP)} \qquad \min f(X\beta) \ \text{ s.t. } \ \sum_{i=1}^{r} \beta_i = 1, \quad \beta_i \geqslant 0, \quad i = 1, 2, ..., r.$$

X is the $n \times r$ matrix comprised of a subset of extreme points of the convex hull, along with one of the current iterates x^k or a past iterate. There are r such points in X. Note that in the hypothetical case where we know all the extreme points of the convex hull, (MP) would have been equivalent to (CHR). Naturally, if the method required such equivalence, there would be no point in using it to solve (CHR). Luckily, any point within the convex hull of a set can be described as a convex combination of at most $n + 1$ points within that set, a result of Caratheodory Theorem. Therefore, the optimal point can be written as a convex combination of a subset of extreme points. Simplicial decomposition takes advantage of this observation, introducing only one extreme point obtained from the subproblem per iteration. Then at the master problem stage, (MP) is solved, which is a minimization problem over a $r-1$ dimensional simplex. If the optimal solution of (CHR) is within this simplex, then the algorithm terminates. If not, the optimal solution β^* of (MP) will be used to compute the next iterate, x^{k+1}, which can be found using the following formula:

$$x^{k+1} = \sum_{i=1}^{r} \beta_i^* \times X_i$$

Then we go back to the subproblem, find another extreme point and increase the dimension of the simplex for (MP). It may seem as if a considerable number of extreme points have to be included in X, before finding the optimal solution. Fortunately, this is not the case. This can be perhaps explained by the way extreme points are introduced to the X matrix. At each sub-problem, the extreme point chosen y^* is the one which yields the steepest descent direction as $\nabla f(x^k)^T \times (y^* - x^k)$ is minimal among all such directions. Therefore at each iteration, we are quickly progressing toward the optimal solution, in contrast what would happen if extreme points were to be chosen arbitrarily.

For some pathological cases, putting no restriction on r could potentially pose computational problems. Restricted simplicial decomposition, introduced by Hearn et al. (1987) puts a restriction on the number of extreme points that can be kept. However, even for such pathological cases, there are certain trade-offs between restricted simplicial decomposition and unrestricted simplicial decomposition. Discussing these tradeoffs is beyond the scope of this paper.

3.4. Convergence to the Optimal Solution of CHR

Because the objective function is convex, the necessary and sufficient optimality condition for x^k to be the global minimum is

$$\nabla f(x^k)^T(y^* - x^k) \geqslant 0$$

Lemma 2 of Hearn et al. (1987) proves that if x^k is not optimal, then $f(x^{k+1}) < f(x^k)$, so that the sequence $\{x^k\}$ is monotonically decreasing. Finally Lemma 3 of Hearn et al. (1987) shows that any convergent subsequence of $\{x^k\}$ will converge to the global minimum. The result is proved using contradiction that one cannot have a subsequence such that $\nabla f(x^\infty)^T(y^\infty - x^\infty) \geqslant 0$ where $x^k \to x^\infty$, $y^* \to y^\infty$.

4. Algorithm

The algorithm used in this study follows the restricted simplicial decomposition (Hearn et al. 1987). The parameter R denotes the maximum number of extreme points allowed to be used in solving the master problem. In the test runs done for this paper, the number of extreme points stored in the matrix X was manageable, so that we put no limit on R, making the algorithm below equivalent to the unrestricted simplicial decomposition method of von Hohenbalken (1977). The stopping condition for the algorithm is taken from Contesse & Guignard (1996, 2007).

Note that in this notation:

(1) $[\mathrm{W_s}]^\mathrm{k}$ is the collection of extreme points stored at iteration k,
(2) $[\mathrm{W_x}]^\mathrm{k}$ stores one point, a current or a past linearization point.

Then, with this notation the master problem introduced in section 3.3 will be:

$$\min f(X\beta) \text{ s.t. } \sum_{i=1}^{r} \beta_i = 1, \quad \beta_i \geqslant 0, \;\; i = 1,2,...,r, \;\; X = [\mathrm{W_s}]^{\mathrm{k}} \cup [\mathrm{W_x}]^{\mathrm{k}} = [\mathrm{W}]^{\mathrm{k}}.$$

An important point to note is that we discard those points within $[\mathrm{W_s}]^{\mathrm{k+1}}$ and $[\mathrm{W_x}]^{\mathrm{k+1}}$ with $\beta_i = 0$ after solving the master problem. This prevents an excessive increase in the number of extreme points stored in $[\mathrm{W_s}]^{\mathrm{k+1}}$.

Step 0: Take a feasible point x^0. Set $k = 0$, $[\mathrm{W_s}]^0 = \emptyset$, $[\mathrm{W_x}]^0 = \left\{x^0\right\}$

Step 1: Solve $\min\left\{\nabla \mathrm{f}\left(\mathrm{x^k}\right)\mathrm{y} : \mathrm{y} \in \mathrm{S}\right\}$ and let $\mathrm{y^k} = \operatorname{argmin}\left\{\nabla \mathrm{f}\left(\mathrm{x^k}\right)\mathrm{y} : \mathrm{y} \in \mathrm{S}\right\}$.

(1) If $\left|[\mathrm{W_s}]^{\mathrm{k}}\right| < R$, set $[\mathrm{W_s}]^{\mathrm{k+1}} = [\mathrm{W_s}]^{\mathrm{k}} \cup \left\{\mathrm{y^k}\right\}$, and $[\mathrm{W_x}]^{\mathrm{k+1}} = [\mathrm{W_x}]^{\mathrm{k}}$

(2) If $\left|[\mathrm{W_s}]^{\mathrm{k}}\right| = R$, take the element of $[\mathrm{W_s}]^{\mathrm{k}}$ with the minimal weight out and put y^k in instead to obtain $[\mathrm{W_s}]^{\mathrm{k+1}}$, and let $[\mathrm{W_x}]^{\mathrm{k+1}} = \{x^k\}$. Set $[\mathrm{W}]^{\mathrm{k+1}} = [\mathrm{W_s}]^{\mathrm{k+1}} \cup [\mathrm{W_x}]^{\mathrm{k+1}}$, and go to step 2.

Step 2: Let $x^{k+1} = \operatorname{argmin}\left\{f(x) : \;\; x \in H\left([\mathrm{W}]^{\mathrm{k+1}}\right)\right\}$, where $H\left([\mathrm{W}]^{\mathrm{k+1}}\right)$ is the convex hull of the points corresponding to the columns of $[\mathrm{W}]^{\mathrm{k+1}}$. Write $x^{k+1} = \sum_{i=1}^{r} \beta_i W_i$. Then take all elements with weights $\beta_i = 0$ out of [W]. Set $k = k + 1$ and go to step 3.

Step 3: If $\left|x^k - x^{k-1}\right| < \varepsilon_x \max\left\{\left|x^k\right|, \left|x^{k-1}\right|\right\}$, or $\left|f(x^k) - f(x^{k-1})\right| < \varepsilon_f \left|f(x^{k-1})\right|$, x^k is a solution, then terminate. Otherwise, go to step 1.

5. Calculating lower and upper bounds for convex GQAP problems

As stated in Definition 28.1, (CHR) is a relaxation to the (NLIP). Simplicial Decomposition finds an optimal solution, say, x^*, to (CHR), and this provides a lower bound on v(NLIP):

$$\mathrm{LB_{CHR}} = f(x^*)$$

On the other hand, at each iteration k of the subproblem an extreme point, y_*^k, of the convex hull is found , which is an integer feasible point of (NLIP). Each point y_*^k yields an Upper Bound (UB) to the optimal value of the (NLIP), and the best upper bound on v(NLIP) can be computed as

$$\mathrm{UB_{CHR}} = \min\{f(y_*^1), f(y_*^2), ..., f(y_*^k)\}$$

To demonstrate the ability of the CHR approach to compute bounds often significantly better than the continuous relaxation bounds, we implemented CHR to find a lower bound on the optimal value of convex GQAPs (Generalized Quadratic Assignment Problems).

We used two types of data. First we adapted data from GQAPs from the literature, with instances of size 30×15 , $35{\times}15$ and $30{\times}20$, and measured the improvement over the continuous bound by computing the ratio of

$$\frac{\text{CHR bound - NLP bound}}{\text{best feasible value found - NLP bound}}$$

as a measure of how much the gap is reduced when one replaces the NLP bound by the CHR bound. We computed the matrix of the objective function by premultiplying the original matrix Q by its transpose, where the entries of Q are products of a flow by a distance as given in the original GQAP instances. These problems tend to have moderate integrality gaps. The results were in the range 43 to 99 %. The largest runtime on a fast workstation was 12 seconds.

The second data set uses again data from the GQAP literature, and generates the objective function matrix as the product of a matrix by its transpose, but this matrix is now randomly generated with coefficients between -500 and +500. These tend to have large duality gaps and to be considerably more difficult to solve. We will describe in the next paragraph how one can reduce this gap.

6. A priori Improvement of Lower Bounds

For some of the convex data sets generated for GQAP, the gaps between the continuous and the CHR bound on the one hand, and the optimal value on the other, are very large. Since CHR computes a bound based on the convex hull of all 0-1 feasible solutions, there is nothing that can be done to improve that part of the model, like adding cuts or tightening inequalities. The only place where an improvement remains possible is then the objective function.

The reason we mention the continuous relaxation bound is that we know that the CHR bound must be at least as good or better. If we can manage then to increase the continuous bound, we must be able to increase the CHR bound as well.

Consider the convex function $f(x) = ux(x-1)$, $x \in \{0,1\}$, u a positive scalar. The problem is to minimize $f(x)$ subject to $x \in \{0,1\}$. The function is zero for $x = 0$ or 1, but it is negative in between. If one computes the continuous bound on $f(x)$ for $x \in [0,1]$, one gets $u(1/2)(-1/2) = -u/4$, and if u is large, so is the integrality gap. Notice that one could however replace $f(x)$ by $g(x) = ex(x-1)$ with $e > 0$, e very close to 0, it would produce an equivalent problem $\min\{g(x) | x \in \{0,1\}\}$. Indeed $g(x)$ coincides with $f(x)$ over the feasible set $\{0,1\}$, yet it gives a much better lower bound equal to $-e/4$, and it is a convex function as long as $e > 0$, no matter how small.

If one has a convex objective function of n variables, the same kind of difficulty may occur, i.e., the continuous bound, and thus the integrality gap, may be very large because the value of the objective function drops substantially when the variables are allowed to be between 0 and 1.

Convexification (see for instance Billionnet, Elloumi and Plateau 2008) in its simplest form adds terms of the form $u_{ij}(x_{ij}^2 - x_{ij})$, with u_{ij} real, to the quadratic objective function. To convexify a nonconvex quadratic function, one tends to <u>add</u> positive terms to the diagonal of the matrix, to make it positive semidefinite. Here one will tend to <u>subtract</u> positive terms from the diagonal, as long as the objective function remains convex. This will not change the objective function for x_{ij} 0 or 1. We will call this backward process de-convexification, even though it leaves the problem convex.

De-convexification through SDP is possible (see for instance the GAMS website link

`http://www.gams.com/modlib/libhtml/gqapsdp.htm`

showing an application to GQAP using CSDP) but somewhat expensive. Some commercial programs for quadratic optimization, on the other hand, detect non-convexity of an MIQCP objective function. CPLEX is one of them. If one is willing to get results somewhat inferior to those of the SDP approach, one can set all u_{ij} equal, say, to some $u > 0$, and one can determine the largest value of u that keeps the function $g(x) = f(x) - u\sum_{ij}(x_{ij}^2 - x_{ij})$ convex. We call this the diagonal de-convexification. One can then replace $f(x)$ by $g(x)$ in the model to compute improved lower bounds and hopefully solve it more easily.

7. Computational results

7.1. Instances with large integrality gaps.

To illustrate the diagonal de-convexification process, we will use generalized quadratic assignment instances with a large duality gap. We present results for two instances of size 16×7 and 20×10 respectively. 16×7 is the largest problem size that we were able to solve using CPLEX as a mixed-integer quadratic problem. Its objective function has 0 linear terms.

Table 1 concerns the 16×7 GQAP problem. It shows how the continuous relaxation bound improves (increases) as u increases, until the problem becomes nonconvex.

Table 1. u vs. continuous bound de-convexified problem

0	114,187,401	7,850,000	209,000,689
100	114,188,627	7,875,000	209,295,092
1,000	114,197,260	7,895,000	209,530,577
10,000	114,297,630	7,897,500	209,560,048
100,000	115,417,877	7,898,500	209,571,775
1,000,000	126,470,436	7,898,750	209,574,722
5,000,000	175,074,749	7,899,949	209,589,926
7,500,000	204,873,059	7,950,000	210,177,957
7,600,000	206,053,510	7,975,000	210,972,127
7,700,000	207,233,060	7,977,000	210,495,656
7,800,000	208,411,709	7,977,050	210,496,246

Figure 1 plots the values in Table 1.

Table 2 below compare bounds obtained before and after the "diagonal" de-convexification. Bounds from SDP are even better (290,083,674 for the 16×7 instance, and 147,817,573 for the 20×10 instance), and the CHR bound could of course be computed from that final model, implementation is in progress.

7.2. Instances with smaller integrality gaps

Table 3 shows results for relatively large GQAP instances with smaller integrality gaps. The last two digits of an instance name are for identification, while the first four specify problem size, for instance 30×20 or 40×10. The gaps are computed

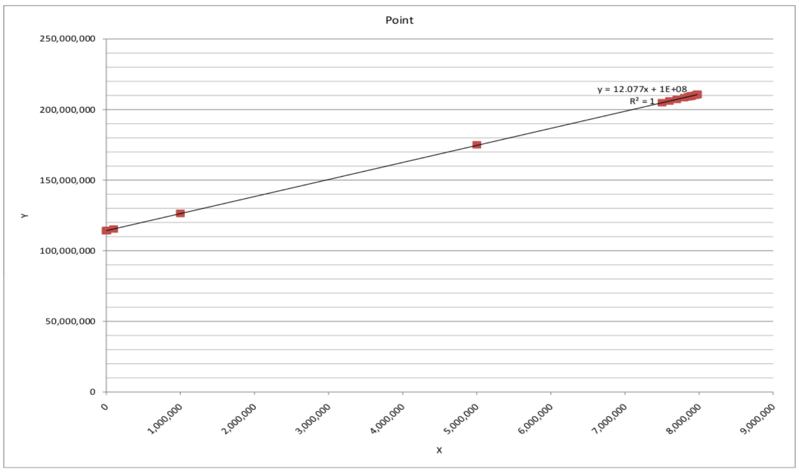


Table 2. Instances 1 and 2

16×7	Continuous bound	CHR bound	RLT-1 bound	Optimal value or best feasible value	Largest u s.t. problem convex
Original convex function	114,187,401	139,645,033	185,918,839		
De-convexified function	210,434,185	232,950,970	185,918,839	577,900,705	7,977,050

20×10	Continuous bound	CHR bound	RLT-1 bound	Optimal value or best feasible value	Largest u s.t. problem convex
Original convex function	56,255,405	56,342,009	-418,102,042		
De-convexified function	91,498,550	91,592,623	-418,102,042	469,555,787	2,126,110

from the best known integer feasible solution values (IFV) to-date. Replacing the continuous bound by the CHR bound almost bridges the gap in most cases.

8. Conclusion

The Convex Hull Relaxation (CHR), possibly combined with de-convexification, provides tight lower and upper bounds by (1) transforming a nonlinear integer optimization problem in one over the convex hull of all integer feasible solutions, and (2) replacing this problem by a sequence of integer linear programs and simple nonlinear continuous programs. The potential strength of the proposed algorithm is that the difficulty of the problems solved at each iteration stays relatively unchanged from iteration to iteration. It will be most suitable for those nonlinear integer problem types that would be much easier to solve with a linear objective function. One should expect that CHR will have a robust performance for large-scale problems if

Table 3. Continuous vs. CHR bound

Instance	Continuous bound	CHR bound	Best IFV	Improvement
30-20-35	28,996,191	29,210,393	29,267,194	79.04%
30-20-55	18,721,636	18,925,414	18,932,878	96.47%
30-20-75	24,882,925	25,273,194	25,299,161	93.76%
30-20-95	21,162,833	23,787,852	23,809,571	99.18%
35-15-35	32,750,712	32,769,891	32,795,216	43.09%
35-15-55	27,443,328	27,605,620	27,621,169	91.26%
35-15-75	30,638,516	30,920,476	30,928,923	97.09%
35-15-95	34,722,239	35,825,436	35,886,295	94.77%
50-10-75	56,103,056	56,606,460	56,615,187	98.30%
50-10-95	71,684,990	72,082,091	72,099,812	95.73%

one has access to solvers able to handle large integer linear programs and simple nonlinear programs efficiently. Current experiment seem to confirm this behavior.

References

1. V. Albornoz. Diseno de Modelos y Algoritmos de Optimizacion Robusta y su Aplicacion a la Planificacion Agregada de la Produccion. *Doctoral Dissertation, Universidad Catolica de Chile*, Santiago, Chile, 1998.
2. A. Billionnet, S. Elloumi, M.-C. Plateau. Quadratic 0-1 Programming: Tightening Linear or Quadratic Convex Reformulation by Use of Relaxations, *RAIRO-RO* 42:103-121, 2008.
3. L. Contesse, M. Guignard. An Augmented Lagrangean Relaxation for Nonlinear Integer Programming Solved by the Method of Multipliers, Part II: Application to Nonlinear Facility Location. *Working Paper*, 1996 (latest revision 2007).
4. M. Guignard. Primal Relaxation in Integer Programming. *VII CLAIO Meeting*, Santiago, Chile, 1994, also *Operations and Information Management Working Paper 94-02-01*, University of Pennsylvania, 1994.
5. D.W. Hearn, S. Lawphongpanich, J.A. Ventura. Restricted Simplicial Decomposition: Computation and Extensions. *Mathematical Programming Study* 31, 99-118, 1987.
6. B. Von Hohenlbalken. Simplicial decomposition in non linear programming algorithms. *Mathematical Programming*, 13, 49-68, 1997.

Acknowledgments

Research partially supported under NSF Grant DMI-0400155.

29

Rounding-based heuristics for nonconvex MINLPs

Giacomo Nannicini[1] **Pietro Belotti**[2]

[1] Tepper School of Business
Carnegie Mellon University
Pittsburgh, PA, 15217, USA
nannicin@andrew.cmu.edu

[2] Department of Industrial & Systems Engineering
Lehigh University
Bethlehem, PA, 18015, USA
belotti@lehigh.edu

Abstract

We propose two primal heuristics for possibly nonconvex MINLPs, based on the idea of rounding the solution to a continuous NLP subject to linear constraints. The linear constraints consist in the linearization of the feasible region of the problem, and in local branching cuts [**3**]. Our heuristics use the same algorithmic scheme, but they differ in the continuous NLP problem which is solved to compute the point that is to be rounded, and in the use of local branching cuts. We design a heuristic that tries to find a first feasible solution, and an improvement heuristic which starts at a feasible solution and tries to compute one with a better objective function value. We mainly deal with MINLPs with binary variables, but we also discuss the extension to general integer variables. Computational results show the effectiveness in practice of these simple ideas.

Keywords: primal heuristics, nonconvex MINLPs, local branching.

1. Introduction

Mixed-Integer Nonlinear Programs (MINLPs) are mathematical programs that are very difficult to solve in practice. In this paper we focus on nonconvex MINLPs, that is, mathematical programs which involve both integer and continuous variables, and where the objective function and constraints can be nonconvex. Difficulties in the solution of nonconvex MINLPs arise because they inherit all the problems related to solving large-scale Nonlinear Programs (NLPs) and Mixed-Integer Linear Programs

(MILPs). In recent years, some attempts have been made in developing exact solvers for nonconvex MINLPs [**1, 5**]. These solvers are based on the Branch-and-Bound (BB) algorithm, where lower bounds (we assume a minimization problem) at each node of the BB tree are typically obtained by solving a Linear Program (LP) that defines a relaxation of the corresponding MINLP.

Within the context of a BB algorithm for nonconvex MINLPs, obtaining good upper bounds as quickly as possible is of great practical importance; in this paper we study two heuristics for this purpose. Our primal heuristics are based on the same algorithm scheme: first, we obtain a feasible point for the continuous relaxation (we denote such a point *constraint feasible* in the following) by solving a NLP, then we round the solution to this NLP, subject to linear constraints, to obtain an integral feasible point. This rounding is accomplished through the solution of a MILP. Finally, we fix the integer variables and act on the continuous variables to generate a MINLP feasible point.

We make an extensive use of local branching cuts, i.e. the linear cuts which define a neighbourhood of a solution employed in the local branching heuristic for MILPs [**3**]. We apply these cuts both to define a neighbourhood of a solution and to cut off integer solutions which we have already visited during the previous iterations of the heuristic. The use of local branching cuts is the reason why we focus on MINLPs with binary variables. However, we also discuss the extension to general integer variables, which will be included in the full paper.

2. Notation and Preliminaries

Consider the following mathematical program:

$$\left.\begin{array}{rr} \min & f(x) \\ \forall j \in M & g_j(x) \leq 0 \\ \forall i \in N & x_i^L \leq x_i \leq x_i^U \\ \forall i \in N_I & x_i \in \mathbb{Z}, \end{array}\right\} \mathcal{P} \tag{29.1}$$

where f and g are possibly nonconvex functions, $n = |N|$ is the number of variables, and $x = (x_i)_{i \in N}$ is the vector of variables. Difficulties arise from the integrality of some of the variables, as well as nonconvexities (if present). Solution methods typically require that the functions f and g are factorable, that is, they can be expressed as $\sum_i \prod_j h_{ij}(x)$ [**5**]. Branch-and-Bound methods for MINLPs attempt to closely mimick their MILP counterparts. Lower bounds are typically obtained by computing a convex relaxation (simply *convexification* from now on) of the feasible region of the problem.

In the following, we will assume that the Branch-and-Bound method that we address computes a linear convexification of the original problem; that is, the objective function f and all constraints g_j are replaced by suitable linear terms which underestimate and overestimate the original functions over all the feasible region. The accuracy of the convexification greatly depends on the variable bounds.

Local branching [**3**] is an improvement heuristic for BB algorithms which relies on exploring a neighbourhood of the incumbent, looking for a better solution. For problems with only continuous and binary variables, the neighbourhood is defined by adding a *local branching constraint* to the original problem, obtaining the local branching problem. Let $B \subseteq N_I$ be the set of binary variables, $0 < k \in \mathbb{N}$, and let

$\bar{x}$ be any feasible solution; then the local branching constraint is:

$$\sum_{i \in B: \bar{x}_i = 1} (1 - x_i) + \sum_{i \in B: \bar{x}_i = 0} x_i \leq k. \tag{29.2}$$

This constraint has the effect of allowing only k binary variables to flip their value from 0 to 1 or vice versa. Typically, k is a small value; experiments in [**3**] suggest $k \approx 10$. As a consequence, the number of feasible solutions of the local branching problem is very small, and an efficient BB code requires little time to find its optimal solution. Note that if we reverse the sense of the inequality (29.2), then we obtain a linear inequality that cuts off $\bar{x}$ and requires solutions to differ from $\bar{x}$ on at least k binary variables:

$$\sum_{i \in B: \bar{x}_i = 1} (1 - x_i) + \sum_{i \in B: \bar{x}_i = 0} x_i \geq k. \tag{29.3}$$

Following [**2**], we call an inequality of the type (29.3) with $k = 1$ a *no-good cut.*

3. Main algorithmic ideas

The heuristics presented in the paper follow a common scheme, whose main component is the rounding of a constraint feasible solution, subject to linear constraints. These linear constraints always include the linearization of the original feasible region, and typically also local branching cuts (both (29.2) and (29.3)). We introduce some notation.

Let $\mathcal{Q}$ be the continuous relaxation of $\mathcal{P}$, that is, $\mathcal{P}$ without integrality requirements on the variables. Let x' be a constraint feasible point, i.e. a feasible solution to $\mathcal{Q}$. In the following, we denote by F the linearization of the feasible region of $\mathcal{P}$ with the integrality requirements for the variables in N_I, by $LB(\bar{x}, k) = \{x \in \mathbb{R}^n : \sum_{i \in B: \bar{x}_i = 1}(1 - x_i) + \sum_{i \in B: \bar{x}_i = 0} x_i \leq k\}$, and by $NG(\bar{x}) = \{x \in \mathbb{R}^n : \sum_{i \in B: \bar{x}_i = 1}(1 - x_i) + \sum_{i \in B: \bar{x}_i = 0} x_i \geq 1\}$. F is the intersection of a polyhedron with $\mathbb{R}^{|N \setminus N_I|} \times \mathbb{Z}^{|N_I|}$. With a slight abuse of notation, for a problem $\mathcal{P}$ and $S \subset \mathbb{R}^n$ we denote by $\mathcal{P} \cap S$ the problem which is obtained by intersecting the feasible region of $\mathcal{P}$ with S. Given a point $y \in \mathbb{R}^n$, we denote by $\mathcal{Q}(y)$ the problem:

$$\left.\begin{array}{rr} \min & f(x) \\ \forall j \in M & g_j(x) \leq 0 \\ \forall i \in N \setminus N_I & x_i^L \leq x_i \leq x_i^U \\ \forall i \in N_I & x_i = y_i. \end{array}\right\} \mathcal{Q}(y) \tag{29.4}$$

In other words, $\mathcal{Q}(y)$ is the NLP that we obtain from $\mathcal{P}$ by fixing the integer variables of y; therefore, if y has integer components $y_j \in \mathbb{Z}\ \forall j \in N_I$ and $\mathcal{Q}(y)$ has a feasible solution z, z is feasible for the original problem $\mathcal{P}$.

Our idea is to start from a feasible solution x' to $\mathcal{Q}$, and look for an integral solution x^I which is close to x' and is in F. This can be achieved by solving the MILP:

$$\min_{x \in F} \|x - x'\|_1 \tag{29.5}$$

which is equivalent to rounding x' subject to the linear constraints that overapproximate the feasible region of the original problem. The point $x^I = \arg\min_{x \in F} \|x - x'\|_1$ obtained this way is certainly integral feasible, but it is not necessarily constraint feasible with respect to the original problem $\mathcal{P}$. Therefore, we solve $\mathcal{Q}(x^I)$

and hopefully obtain a feasible solution x^* to $\mathcal{P}$. If a termination criterion is not satisfied (which may depend on the feasibility of x^* or on different conditions), we iterate the algorithm by setting $F = F \cap NG(x^I)$, and resolving (29.5), i.e. we try a different rounding of x'.

So far, we have not defined some elements which are necessary for our algorithm: in particular, we need to decide how to compute the initial point x' which is to be rounded, and the termination criterion. Moreover, we may impose additional constraints for the rounding phase. Depending on these details, we obtain different heuristics; in the rest of this paper, we will specialize this basic scheme into two different heuristics (Section 5 and Section 6). The basic scheme is given in Algorithm 2. Note that all subproblems which have to be solved to apply Algo-

```
1:  Initialization: stop ← false
2:  Compute point $x'$
3:  Set $R \leftarrow F$
4:  while ¬stop do
5:      Compute $x^I = \arg\min_{x \in R} \|x - x'\|_1$ with a MILP solver
6:      Solve $\mathcal{Q}(x^I)$ with a NLP solver and obtain point $x^*$
7:      if Termination condition is satisfied then
8:          Set stop ← true
9:      else
10:         Set $R \leftarrow R \cap NG(x^*)$
11:     end if
12: end while
13: return $x^*$
```

Algorithm 2: Basic algorithm

rithm 2 need not be solved to optimality: in principle, only feasibility is required. However, typically $\mathcal{Q}(x^I)$ is always solved to optimality, because early termination can provide a solution with a worse objective function value.

It now becomes clear why we mainly focus on MINLPs with binary variables: whenever the rounding of x' fails, i.e. yields a point x^* which does not satisfy the termination criterion, we want to make sure that at the following iteration we obtain a different point. Since we assumed that $B \neq \emptyset$, this can be easily done by adding a no-good cut, that is, a single linear inequality with integer coefficients that cuts off x^* from the feasible region of the MILP which is solved at line 5 of Algorithm 2. For MINLPs without binary variables, this is not always straightforward. We briefly discuss the extension of this scheme to general integer variables in the next session.

4. Dealing with General Integer Variables

Algorithm 2 needs an efficient way to cut off a point x^* from the feasible region of a MILP. If binary variables are present in the formulation, we accomplish this task through a no-good cut. If there are no binary variables, we propose the following method. If there is a sufficient number of integer variables at one of their bounds in the point x^* that we want to exclude (where "sufficient" depends on the total number of integer variables), then we can easily write a no-good cut:

$$\sum_{i \in B' : \bar{x}_i = x_i^U} (x_i^U - x_i) + \sum_{i \in B' : \bar{x}_i = x_i^L} x_i \geq 1,$$

where B' is the set of integer variables at one of their bounds. Otherwise, we choose an integer variable x_j at random, and change the corresponding bounds, setting $x_j^L = x_j + 1$ or $x_j^U = x_j - 1$ (we want to exclude the smallest interval between $[x_j^L, x_j - 1]$ and $[x_j + 1, x_j^U]$). This ensures that we cannot obtain x^* as the solution of the rounding MILP at the following iteration.

A detailed description and evaluation of these heuristics on instances with only general integer variables is left to the full paper. Through the rest of this extended abstract, we focus on MINLPs with binary variables.

5. Local Branching for MINLPs

In principle, the original local branching for MILPs [**3**] could be applied to nonconvex MINLPs without modifications, by simply employing a BB solver for nonconvex MINLPs instead of a BB solver for MILPs. However, as pointed out in Section 2, BB solvers for nonconvex MINLPs are much slower than their counterparts for MILPs. This motivates our idea for a local branching scheme which does not employ a solver for nonconvex MINLPs to solve the local branching problem, in order to be faster. We apply the scheme presented in Section 3, with small modifications which are intended to favour speed.

First, we specify the missing details for Algorithm 2, namely lines 2 and 7. Let $\bar{x}$ be the solution which we want to improve, and let k be the rhs of (29.2); both $\bar{x}$ and k are inputs for the heuristics. We solve $\mathcal{Q} \cap LB(\bar{x}, k)$, which is a NLP, and obtain the initial constraint feasible point x' which is to be rounded; we use $\bar{x}$ as the starting point for the NLP solver. The stopping condition on line 7 is set to be: $((\forall j \in M \; g_j(x) \leq 0) \wedge (f(x^*) \leq f(\bar{x}))) \vee (\texttt{NumIterations} \geq \mathit{MaxIterations})$, where *MaxIterations* is a parameter; in other words, we exit from the main loop as soon as we find a feasible point which improves over the incumbent $\bar{x}$, or after a maximum number of iterations. Furthermore, on line 3 we replace $R \leftarrow F$ with $R \leftarrow F \cap LB(\bar{x}, k)$; this way, the MILP which is solved on line 5 is a local branching problem, therefore it can be solved efficiently. Summarizing, the local branching heuristic proceeds as indicated in Algorithm 3. We call this heuristic *Local-Branching-based Iterative Rounding* (LB-IR).

6. Feasibility-based Iterative Rounding

In order to provide a starting point for the local branching heuristic, we tailor a second heuristic based on Algorithm 2, aimed at discovering a first feasible solution. For this purpose, we initially try to generate a feasible point with a good objective function, and in case of failure, we iteratively focus more on feasibility than objective function value. In particular, we compute the point x' which is to be rounded by considering a linear combination of the objective function and a measure of feasibility, i.e. the amount by which the point satisfies the inequality constraints $g_j \leq 0 \; \forall j \in M$. Given a parameter $\mu \in [0, 1]$, the problem that we solve to obtain x' is the following:

$$(29.6) \qquad \left.\begin{array}{rr} \min & \mu f(x) + (1-\mu)\sum_{j \in M}(g_j(x)) \\ \forall j \in M & g_j(x) \leq 0 \\ \forall i \in N \setminus N_I & x_i^L \leq x_i \leq x_i^U . \end{array}\right\} \mathcal{F}(\mu)$$

```
1: **Input**: incumbent $\bar{x}$, parameters $k, \mathit{MaxIterations}$
2: **Output**: improved solution $x^*$
3: Initialization: stop $\leftarrow$ false, NumIterations $\leftarrow 0$
4: Solve $\mathcal{Q} \cap LB(\bar{x}, k)$ with a NLP solver to obtain $x'$
5: Set $R \leftarrow F \cap LB(\bar{x}, k)$
6: while $\neg$stop do
7:    Compute $x^I = \arg\min_{x \in R} \|x - x'\|_1$ with a MILP solver
8:    Solve $\mathcal{Q}(x^I)$ with a NLP solver and obtain point $x^*$
9:    if $((x^* \text{ feasible} \wedge f(x^*) < f(\bar{x})) \vee (\text{NumIterations} \geq \mathit{MaxIterations}))$ then
10:       Set stop $\leftarrow$ true
11:    else
12:       Set $R \leftarrow R \cap NG(x^*)$
13:    end if
14:    Set NumIterations $\leftarrow$ NumIterations $+ 1$
15: end while
16: return $x^*$
```

Algorithm 3: Local branching algorithm

This provides a constraint feasible point x' which is a local minimum of $\mathcal{Q}$ if $\mu = 1$ (recall that $\mathcal{Q}$ may be nonconvex), and is more in the interior of the feasible region as μ goes from 1 to 0. For $\mu = 0$, the problem boils down to a minimization of $\sum(g_j(x))$, subject to $g_j \leq 0$ and the box constraints on the variables: that is, we try to obtain the g_j's as negative as possible. Our intuition is that, if the constraints are satisfied by a large amount, then there are more chances to obtain a feasible integer point when rounding x'. This clarifies line 2 of Algorithm 2.

For the termination criterion on line 7, we stop as soon as we find a feasible solution to $\mathcal{P}$, or after a maximum number of iterations. We call this heuristic *Feasibility-based Iterative Rounding* (F-IR); we give a description in Algorithm 4.
In practice, it is natural to run F-IR several times, starting with $\mu = 1$ (which

```
1: **Input**: parameters $\mu, \mathit{MaxIter}$
2: **Output**: feasible solution $x^*$
3: Initialization: stop $\leftarrow$ false, NumIter $\leftarrow 0$
4: Solve $\mathcal{F}(\mu)$ with a NLP solver to obtain $x'$
5: Set $R \leftarrow F$
6: while $\neg$stop do
7:    Compute $x^I = \arg\min_{x \in R} \|x - x'\|_1$ with a MILP solver
8:    Solve $\mathcal{Q}(x^I)$ with a NLP solver and obtain point $x^*$
9:    if $(x^* \text{ feasible}) \vee (\text{NumIter} \geq \mathit{MaxIter})$ then
10:       Set stop $\leftarrow$ true
11:    else
12:       Set $R \leftarrow R \cap NG(x^*)$
13:    end if
14:    Set NumIter $\leftarrow$ NumIter $+ 1$
15: end while
16: return $x^*$
```

Algorithm 4: Feasibility-based Iterative Rounding

is likely to find a solution with better objective value, if one is found) and then decreasing μ until it reaches zero.

7. Computational Experiments

In this section, we provide computational experiments to assess the practical usefulness of the proposed heuristics. Both LB-IR and F-IR where implemented within `Couenne` [**1**], an open-source BB solver for nonconvex MINLPs. Through the rest of this section, the linearization of the feasible region of the original problem $\mathcal{P}$ is assumed to be the one provided by `Couenne` at the root node of the BB tree, after the bound tightening phase which is carried out with the default parameters. We try at most 10 different roundings of a point x' (see Algorithm 2) before reporting a failure of the heuristic. We used `Cplex 11.0` [**4**] as the MILP solver, and `Ipopt` [**6**] as the NLP solver. We set a time limit of 5 seconds for the solution of all MILPs arising during the application of the heuristics, whereas no time limit was given for the solution of NLPs. We set the parameter `MIP_EMPHASIS` of Cplex to `HIDDENFEAS`. All experiments were run on one core of a machine equipped with an Intel Xeon 2.4 GHz and 8 GB of RAM, running Linux. For space reasons, in this section we only report average values instead of full tables of results.

We tested LB-IR and F-IR on 23 challenging instances with binary variables taken from MINLPLib. Our full test set consists in the instances reported in Table 1; we report the number of variables, the number of integer variables (which includes binary and general integer variables), the number of constraints. In the computational experiments, we consider two subsets of the full test set: one consisting in the instances for which the default configured version of `Couenne` found a feasible solution in less than 2 hours of CPU time, and one consisting of the instances for which the default configured version of `Couenne` did not find a solution at the root node. We call these two subsets, respectively, TS1 and TS2; the list of instances in each of them is provided in Table 1.

Instance	# Cons	# Vars (Ints)	TestSet	Instance	# Cons	# Vars (Ints)	TestSet
`cecil_13`	899	841 (180)	TS2	`space25a`	202	384 (240)	TS2
`csched2a`	138	233 (140)	TS1-TS2	`space25`	236	894 (750)	TS2
`csched2`	138	401 (308)	TS1-TS2	`synheat`	65	57 (12)	TS1
`deb10`	130	183 (22)	TS2	`tls12`	155	346 (296)	TS2
`deb6`	508	476 (20)	TS2	`tls4`	65	106 (89)	TS1-TS2
`eniplac`	190	142 (24)	TS1-TS2	`tls5`	91	162 (136)	TS1-TS2
`fo7`	212	115 (42)	TS1-TS2	`tls6`	121	216 (179)	TS1-TS2
`fo8`	274	147 (56)	TS1-TS2	`tls7`	155	346 (296)	TS2
`fo9`	344	183 (72)	TS1-TS2	`water4`	138	196 (126)	TS1-TS2
`lop97icx`	88	987 (899)	TS1-TS2	`waterx`	55	71 (14)	TS2
`m7`	212	115 (42)	TS1-TS2	`waterz`	138	196 (126)	TS1-TS2
`qap`	31	226 (225)	TS1				

Table 1. Details on the test set.

In the following, we will denote by `Couenne` the default configured version of the solver, by `CouenneLB` the version which includes the LB-IR heuristic, by `CouenneF` the version which includes the F-IR heuristic, and by `CouenneIR` the version which includes both F-IR and LB-IR.

In the first set of experiments, we want to assess the effectiveness of LB-IR. To this aim, we focus on TS1, that is, the instances for which `Couenne` finds a feasible solution in less than two hours. The aim of an improvement heuristic such as LB-IR is to keep a good incumbent through the BB algorithm. We propose the following measure: we run `Couenne` and `CouenneLB` in parallel, and we record for how much time one of the two solvers has an incumbent which is strictly better than the incumbent of the other one. We then compute a percentage value by dividing by the maximum time (2 hours) if the instance is not solved, or by the time taken for the fastest of the two algorithms to prove optimality. We obtain a value between 0 and 1 for each of the two algorithms. This can be seen as a measure of the probability of one of the two algorithms having a better incumbent at a given time instant. In our experiments, we obtain the best results setting $k = 15$. With this value, `CouenneLB` holds a better incumbent with respect to `Couenne` for 62.0% of the time on average, and for an additional 26.4% they have an incumbent with the same objective value. Moreover, for $k = 15$ `CouenneLB` finds a better solution at the end of the two hours time limit for 9 instances, and the same solution for an additional 4 (`Couenne` wins in the remaining 2 instances).

Next, we perform experiments to determine the usefulness of F-IR; thus, we consider the test set TS2, i.e. the instances for which `Couenne` does not find a solution at the root node, and we analyse the solutions computed by F-IR at the root node. Recall that F-IR has a parameter μ which determines the objective function, so that using a different μ yields a different point x'; in each run, we apply F-IR with differents value of μ, starting from $\mu = 1$ and decreasing by equal steps so that the last application has $\mu = 0$. We stop as soon as we find a feasible solution; if no solution is found, we report a failure. Formally, let h be the maximum number of different values of μ that we are willing to try; we set $\mu_1 = 1$ and for $i \geq 2$ we set $\mu_i = 1 - (i - 1)/(h - 1)$. In our experiments, we obtain the best results for $h = 5$. With this value of the parameter, we find a feasible solution through F-IR for 14 instances, with an average CPU time of 67.87 seconds, and the objective value of these solutions is on average 27.15% away from the best known objective value for those instances.

Finally, we combine LB-IR (with $k = 15$) and F-IR (with $h = 5$) together into `CouenneIR`. On the complete test set of 23 instances, `CouenneIR` holds an incumbent with an objective value strictly better than the incumbent held by `Couenne` for 67.94% of the time, whereas `Couenne` has a strictly better incumbent for 12.21% of the time. Furthermore, `CouenneIR` returns a better solution on 12 instances, `Couenne` returns a better solution on 2 instances, and on the remaining 9 instances they return the same solution. Clearly, `CouenneIR` (i.e. `Couenne` plus both heuristics) performs better than default `Couenne`. Not only it finds feasible solutions to a larger set of instances, but it also find better solutions on average, and typically has a better incumbent during the BB algorithm. Furthermore, the computational overhead is small, so that on those instances where the heuristics fail and there is no improvement in terms of objective function, the CPU time lost is negligible with respect to the 2 hours of time limit.

References

1. P. Belotti. Couenne: a user's manual. Technical report, Lehigh University, 2009.
2. C. D'Ambrosio. *Application Oriented Mixed Integer Nonlinear Programming.* PhD thesis, DEIS, Università di Bologna, 2009.
3. M. Fischetti and A. Lodi. Local branching. *Mathematical Programming*, 98:23–37, 2003.

4. ILOG. *ILOG CPLEX 11.0 User's Manual*. ILOG S.A., Gentilly, France, 2007.
5. M. Tawarmalani and N. Sahinidis. Global optimization of mixed integer nonlinear programs: A theoretical and computational study. *Mathematical Programming*, 99:563–591, 2004.
6. A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

Combinatorial Integral Approximation for Mixed-Integer Nonlinear Optimal Control

Sebastian Sager **Michael Jung** **Christian Kirches**

Interdisciplinary Center for Scientific Computing
University of Heidelberg
INF 368, 69120 Heidelberg, Germany

sebastian.sager@iwr.uni-heidelberg.de

Keywords: MINLP, MIOCP, MILP, optimal control, integer programming.

We are interested in structures of and efficient methods for mixed-integer nonlinear programs (MINLP) that arise from a *first discretize, then optimize* approach to time-dependent mixed-integer optimal control problems (MIOCPs). In this study we focus on combinatorial constraints, in particular on restrictions on the number of switches on a fixed time grid

$$0 = t_1 < \ldots < t_{n_\mathrm{t}+1} = t_\mathrm{f}, \tag{30.1}$$

which we will use for a discretization of the control in a first discretize, then optimize approach. We consider a problem with piecewise constant binary control functions,

$$\omega_k(t) = p_{k,i} \in \{0,1\}, \quad t \in [t_i, t_{i+1}],\ k = 1 \ldots n_\omega,\ i = 1 \ldots n_\mathrm{t} \tag{30.2}$$

with $p_{k,i} \in \{0,1\}$. Special emphasis is on the switching constraints

$$\sum_{i=1}^{n_\mathrm{t}-1} |p_{k,i+1} - p_{k,i}| \leq \sigma_{k,\max}, \quad k = 1 \ldots n_\omega. \tag{30.3}$$

We propose a novel approach that is based on a decomposition of the MINLP into a NLP and a MILP. We discuss the relation of the MILP solution to the MINLP solution and formulate bounds for the gap between the two, depending on Lipschitz constants and the control discretization grid size. The MILP solution can also be used for an efficient initialization of the MINLP solution process.

Previously obtained results on integer gaps have been used in several ways. Most importantly they imply that, if the control discretization grid is fine enough, no integer gap exists [**2**], because Δt can be chosen arbitrarily small and the estimation carries over to continuous objective and constraint functions. Also, the specific way of constructing a binary solution can be used, e.g., in the adaptive algorithm MINTOC, [**2, 1**]. However, both uses require that the constructed binary control is feasible for the original problem. This is not a problem if only constraints on the differential states are present when $\Delta t \to 0$, but switching constraints will typically be violated if Δt is small.

Therefore we propose to change the point of view: while before it was argued that the difference between integer and relaxed solution will become arbitrarily small if $\Delta t \to 0$, we now consider Δt to be fixed and allow a larger constant to obtain a feasible solution.

To be able to include constraint (30.3) we determine p as the solution of the MILP

$$\begin{array}{llll} & \min_{p} \max_{k=1\dots n_\omega} \max_{i=1\dots n_\mathrm{t}} \left| \sum_{j=1}^{i} (q_{k,j} - p_{k,j}) \Delta t_j \right| & & \\ (30.4) & \text{subject to} & & \\ & \sigma_{k,\max} \geq \sum_{i=1}^{n_\mathrm{t}-1} \left| p_{k,i} - p_{k,i+1} \right|, & k = 1 \dots n_\omega, & \\ & p_i \in \{0,1\}, & i = 1 \dots n_\mathrm{t}, & \end{array}$$

for fixed control values q that stem from the solution of the relaxed nonlinear problem and given upper bounds on the number of switches, $\sigma_{k,\max}$. The objective function is related to higher level approximation results, and hence allows for error estimates between the MILP and the MINLP solution.

The speedup of the solution of the MILP compared to the MINLP solution is considerable already for general purpose MILP solvers. We analyze the structure of the MILP that takes switching constraints into account, discuss properties of the feasible polytope, and propose a tailored Branch and Bound strategy that outperforms CPlex and Gurobi on a numerical case study.

References

1. S. Sager. Reformulations and algorithms for the optimization of switching decisions in nonlinear optimal control. *Journal of Process Control*, 19(8):1238–1247, 2009.
2. S. Sager, G. Reinelt, and H.G. Bock. Direct methods with maximal lower bound for mixed-integer optimal control problems. *Mathematical Programming*, 118(1):109–149, 2009.

PART 3
Posters

On a mixed 0–1 separable nonlinear model for water irrigation scheduling

Marc Almiñana[1] Laureano Escudero[2]
Mercedes Landete[1] Juan Monge[1] Alejandro Rabasa[1]
Joaquín Sánchez-Soriano[1]

[1] Centro de Investigación Operativa,
Universidad Miguel Hernández
Elche (Alicante), Spain
{marc,landete,monge,a.rabasa,joaquin}@umh.es

[2] Depto. de Estadística e Investigación Operativa
Universidad Rey Juan Carlos
Móstoles (Madrid), Spain
laureano.escudero@urjc.es

Abstract

We present a mixed 0–1 separable nonlinear model for the optimization of the management of water resources for agricultural irrigation usage on a daily basis. It provides dynamic scheduling of the daily irrigation for a given land area by considering the irrigation network topography, water flow technical conditions and logistical operation constraints in order to optimize the usage of the water stored in a reservoir. We also present a solution procedure that iteratively solves a mixed 0–1 linear approximation of the model. A large-scale real-life problem is reported.
Keywords: water resource scheduling, agricultural irrigation, mixed 0–1 separable nonlinear program.

1. Problem description

Consider the problem of an agricultural area to be irrigated on a daily basis where the water flows from a reservoir, that is connected with the hydrants in an arborescent form. There are three types of elements in the pipe system: sector head nodes, hydrant nodes and bifurcation nodes. The last two types of nodes are also called non-head nodes, and they are grouped in sectors which start from the sector heads. These other sectors are inter-connected in the opposite direction to the reservoir, and constitute the set of root nodes of the sector subtrees. Figure 1 illustrates the topology of the system.

Each sector head has a given elevation and a given water pressure. The irrigation is performed in a certain set of consecutive daily time periods (in our case, 5 periods of 4 hours each). Depending upon the dimension of the area to be irrigated by a hydrant, it has the right to be irrigated during a certain number of consecutive time periods. The set of hydrants must satisfy the demand for water and some technical constraints, such as the minimum water pressure and a maximum speed in their upstream tree pipe segments.

The water pressure at a given hydrant is a function of the pressure at its sector head, plus the elevation difference between the sector head and the given hydrant, minus the load loss to be described below. Furthermore, two additional pieces of information are taken into account in the irrigation scheduling problem. First, the sector heads and the bifurcation nodes allow the appropriate water flow distribution, but they do not have their own water needs. Secondly, the time periods for irrigation should be non-preempted (i.e, once the irrigation begins through a hydrant it can not be interrupted).

Given an irrigation network, the system operator assigns a priority to each hydrant for each time period. Hence, the problem involves of selecting the hydrants for irrigation at each time period, such that the sum of priorities is maximized and the technical and logistical constraints are satisfied in the given time frame. An alternative function given by the maximum pressure in pipe segments is minimized. The constraints are the satisfaction of the daily water demand of the hydrants for given consecutive time periods, the maximum water speed and the minimum pressure in the pipe segments.

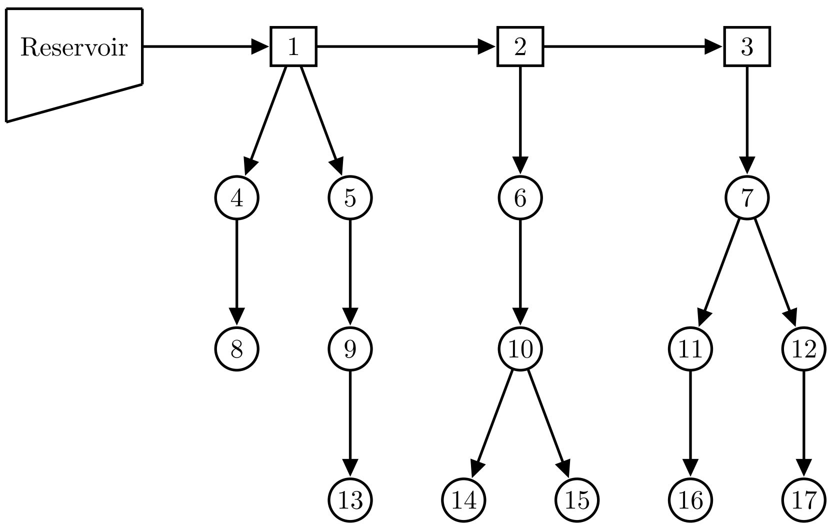


Figure 1. Problem topology. Squared nodes are sector heads and circled nodes are non-headed (hydrants and bifurcation) nodes.

Moreover, the decision support system based on the proposed model and algorithm allows the system operator to irrigate some fields partially on a given day. This feature, plus the tuning of the priority coefficients, allows us to get high priority for irrigation on the next day, in the event that some water users do not obtain satisfaction for their demand on a given day.

In order to describe mathematically the irrigation scheduling problem, we need to take into account the technical, topographical, logistical and irrigation scheduling parameters. The technical and topographical parameters are related to physical properties of the fluids. The logistical parameters are related to the management of the irrigation system and the irrigation scheduling parameters are related to constraints on how to assign the time periods of irrigation. Finally, the decision variables are related to the water discharge in each node for each set of a given number of consecutive time periods at which the hydrants are irrigated.

The organization of the remainder of the note is as follows: Section 2 introduces the notation of the elements of the problem. Section 3 presents the mixed 0-1 separable nonlinear model. Section 4 gives the main ideas of the iterative procedure for problem solving. And section 5 describes the pilot case.

2. Notation of the elements

The notation for all relevant elements and parameters in the system is the following:
Sets of general elements:

- **T:** set of time periods for water irrigation purposes.
- **I:** set of hydrants and bifurcation nodes in the geographical area under consideration.
- **I_0:** set of bifurcation nodes.
- **$I^1 \subset I - I_0$:** subset of hydrants whose irrigation schedule has been partially or completely fixed.
- **$T_i^1 \subset T$:** subset of time periods whose irrigation variables have already been fixed by the system operator for hydrant i, for $i \in I^1$.
- **Γ:** set of sector heads, and $\gamma(i) \in \Gamma$ is the root node (sector head) of the subtree to which hydrant i belongs.
- **R_i:** set of upstream nodes to hydrant i, in its path back to its sector head, including the same hydrant i, for $i \in I - I_0$.
- **S_i:** set of successor nodes to node i, including the same hydrant i, for $i \in I$. Notice that the nodes in set S_i can belong to different successor paths (it is the case where the successor path has bifurcation nodes).

Technical and topographical parameters:

- **$\overline{f}_{it}$:** friction factor for obtaining the pressure in the immediate upstream pipe segment of hydrant i at time period t, to be updated iteratively, for $i \in I - I_0, t \in T$. (For details on the computing of this coefficient see Section 4.1).
- **E_i:** elevation of node i, for $i \in I \cup \Gamma$.
- **L_i:** length of the immediate upstream pipe segment of node i, for $i \in I \cup \Gamma$.
- **D_i:** diameter of the immediate upstream pipe segment of node i, for $i \in I \cup \Gamma$.
- **g:** gravity acceleration coefficient.

Pr_γ: pressure at sector head γ, for $\gamma \in \Gamma$.

$\hat{Pr}$: minimum pressure required by any hydrant at any time period.

$\hat{V}$: maximum water flow speed allowed along the immediate upstream pipe segment of any node at any time period.

k: constant to obtain the water flow volume to irrigate the land area through any hydrant at any time period.

Irrigation scheduling parameters:

$\hat{N}_i$: number of consecutive time periods that hydrant i must irrigate its respective land area, based on its dimension (has), for $i \in I - I_0$.

Logistic parameters provided by the system operator:

c_{it}: priority coefficient for selecting hydrant i to begin a non-preempted irrigation at time period t, for $i \in I - I_0$.

F_i: effective land area (has) to be irrigated by hydrant i, for $i \in I - I_0$.

$\hat{Y}_{it}$: fixed value to 0 or 1 for the variable Y_{it} due to logistic considerations, for $t \in T_i^1, i \in I^1$.

Variables:

Q_{it}: water discharge to flow through node i at time period t to satisfy its own needs, if any, and the water needs of its successor nodes, for $t \in T, i \in I \cup \Gamma$.

Y_{it}: 0–1 variable, such that its value is 1 if the irrigation in hydrant i begins at time period t and, otherwise, it is zero, for $t \in T, i \in I - I_0$. Notice that the irrigation is carried out in periods $t, \ldots, t + \hat{N}_i - 1$ such that $Y_{it} = 1$.

3. Modelization

In order to achieve the aim mentioned in the previous section, the model for the water irrigation scheduling is as follows,

$$(31.1)\qquad (P1)\qquad \max \sum_{i\in I-I_0}\sum_{t\in T} c_{it}Y_{it}$$

$$\text{s.t.}\quad Pr_{\gamma(i)} + E_{\gamma(i)} - E_i$$

$$(31.2)\qquad - \sum_{j\in R_i}\left(\frac{8\overline{f}_{jt}L_j}{\pi^2 g D_j^5}\,Q_{jt}^2\right) \geqslant \hat{Pr} \qquad \forall t\in T, i\in I-I_0$$

$$(31.3)\qquad Q_{it} = \sum_{j\in S_i - I_0 - \Gamma} kF_j\Big(\sum_{\tau=t-\hat{N}_j+1,\ldots,t} Y_{j\tau}\Big) \qquad \forall t\in T, i\in I\cup\Gamma$$

$$(31.4)\qquad \left(\frac{4}{\pi D_i^2}\right)Q_{it} \leqslant \hat{V} \qquad \forall t\in T, i\in I\cup\Gamma$$

$$(31.5)\qquad \sum_{t\in T} Y_{it} = 1 \qquad \forall i\in I-I_0$$

$$(31.6)\qquad Y_{it} = \hat{Y}_{it} \qquad \forall t\in T_i^1, i\in I^1$$

$$(31.7)\qquad Y_{it}\in\{0,1\} \qquad \forall t\in T, i\in I-I_0$$

The program (31.1)–(31.7) has three blocks, namely, the technical and topographical subsystem that comprises the constraints (31.2)–(31.4), the irrigation scheduling block that comprises the constraints (31.5), and the logistic block that comprises the objective function, the fixing bound (31.6) and the parameter F_j in the constraints (31.3).

The objective function (31.1) maximizes the priority given to the irrigation scheduling through the hydrants. The constraints (31.2) are technical expressions to lower bound the pressure in each hydrant node. The equations (31.3) define the water discharge through each node at each time period, and define the number of time periods to irrigate. The constraints (31.4) upper bound the water flow speed allowed along the immediate upstream pipe segment of any node. The special ordered sets (31.5) impose the condition that the hydrant irrigation must be non-preempted. The constraints (31.6) fix (totally or partially) the irrigation schedule for some hydrants, due to logistic considerations by the system operator.

Another interesting objective function consists of the minimization of the maximum pressure of water in the irrigation network at any time period.

$$(31.8)$$

$$(P2)\min \qquad Pr_{\max}$$

$$(31.9)\qquad \text{s.t.}(31.2)-(31.7)$$

$$(31.10)\qquad Pr_{\gamma(i)} + E_{\gamma(i)} - E_i - \sum_{j\in R_i}\left(\frac{8\overline{f}_{jt}L_j}{\pi^2 g D_j^5}\,Q_{jt}^2\right) \leqslant Pr_{\max}, \quad \forall i\in I-I_0, t\in T$$

where $Pr_{\max}$ is the variable that gives the maximum pressure of water along the time horizon.

4. Algorithmic framework

4.1. Computing the friction factor $\overline{f}_{ht}$

The mixed 0-1 nonlinear model (31.1)–(31.7) has an additional difficulty. It is the computation of the friction factor $\overline{f}_{it}$ in the calculation of the load loss for obtaining the pressure in the immediate upstream pipe segment of the hydrant i, for $i \in I - I_0$. This friction factor can be calculated by using the Colebrook formula (see e.g., [**2, 3**]), it has the following expression, where the subindices t and i have been omitted,

$$\frac{1}{\sqrt{\overline{f}}} = -2\log\left(\frac{\epsilon}{3.71D} + \frac{2.51\nu}{VD\sqrt{\overline{f}}}\right), \tag{31.1}$$

where ϵ_i is a constant related to the roughness of the immediate upstream pipe segment of hydrant i, ν is the constant related to the water kinematic viscosity, and V gives the water flow speed that has the following technical expression,

$$V = \left(\frac{4}{\pi D^2}\right) Q. \tag{31.2}$$

The computation of $\overline{f}$ will be iteratively performed at each outer iteration of the algorithm for a given water discharge through the hydrants. Considering (31.1) as a nonlinear equation in $\overline{f}$, we make use of the Newton-Raphson procedure for obtaining its roots. This procedure starts with a value of $\overline{f}$ obtained by using only the first term of the right hand side of the Colebrook formula, and approaches the root by steps which are proportional to the quotient of the function (31.1) and its derivative. (An explicit calculation of the friction factor in a set of special pipes is presented in [**4**]).

4.2. Iterating a mixed 0-1 linear problem

The constraints (31.2) have the quadratic term Q_{jt}^2, for $j \in R_i, t \in T, i \in I - I_0$. It can be approximated by the Taylor series expansion

$$Q_{jt}^2 \approx \overline{Q}_{jt}^2 + 2\overline{Q}_{jt}(Q_{jt} - \overline{Q}_{jt}), \tag{31.3}$$

where $\overline{Q}_{jt}$ is the value from the optimization of the model in the previous iteration.

The algorithm for solving the original problem (31.1)–(31.7) in an outer iterative way is as follows.

Step 1. Compute the friction factor $\overline{f}_{it}$ in expression (31.1), such that Q is replaced by $\overline{Q}_{it}$ in expression (31.2).

Step 2. Solve the mixed 0–1 model (31.1)–(31.7), where the function Q_{jt}^2 is substituted by its linear approximation (31.3). Let Q_{it}^* denote the optimal value of the variable Q_{it} $\forall i \in I, t \in T$.

Step 3. Optimality test. If condition (31.4) is satisfied then stop, the quasi-optimal solution has been obtained. Otherwise, go to Step 4.

$$\sum_{t\in T}\sum_{i\in I}(Q_{it}^* - \overline{Q}_{it})^2 \leqslant \sigma, \tag{31.4}$$

where σ is a positive tolerance.

Step 4. Update

$$\overline{Q}_{it} := \overline{Q}_{it} + \varphi(Q^*_{it} - \overline{Q}_{it}), \quad \forall i \in I, t \in T \tag{31.5}$$

where $0 < \varphi \leqslant 1$, and go to Step 1.

5. Pilot case description

The approach presented in this work has been implemented for solving a real-life problem presented by "La Comunidad de Regantes, Riegos de Levante, Canal 2nd", which is located in Eastern Spain. Its irrigation area comprises 2188 Has and is distributed in 20 pipe sectors (i.e, 20 head nodes) with a total number of 2831 nodes (2025 of them are hydrants with their own water demand needs). The irrigation is needed in a daily basis for $|T| = 5$ time periods (4 hours each, and the additional time period is devoted to maintenance operations). The topography of this problem is depicted in Figure 2. Depending on the dimension of the land area to irrigate, the consecutive time periods are 1 (for 1791 hydrants), 2 (for 191 hydrants), 3 (for 43 hydrants), and none hydrant for 4 and 5 time periods. The water flows from a reservoir with a capacity of $13Hm^3$, and the full system has an arborescence structure. The technical and topographic parameters are as follows: $k = 2.3l/sec/ha$, $\hat{Pr} = 25mca$ $(2.5Kg/cm^3)$, $\hat{V} = 2.5m/sec$, $Pr_\gamma = 40$ $\forall \gamma \in \Gamma$; $\epsilon = 0.003$ mm; and $\nu = 1.08 \times 10^{-6} m^2/s$. The pipe segment parameters L_i and D_i vary from 0 to 691 meters and from 66 to 800 millimeters, respectively, for $i \in I \cup \Gamma$.

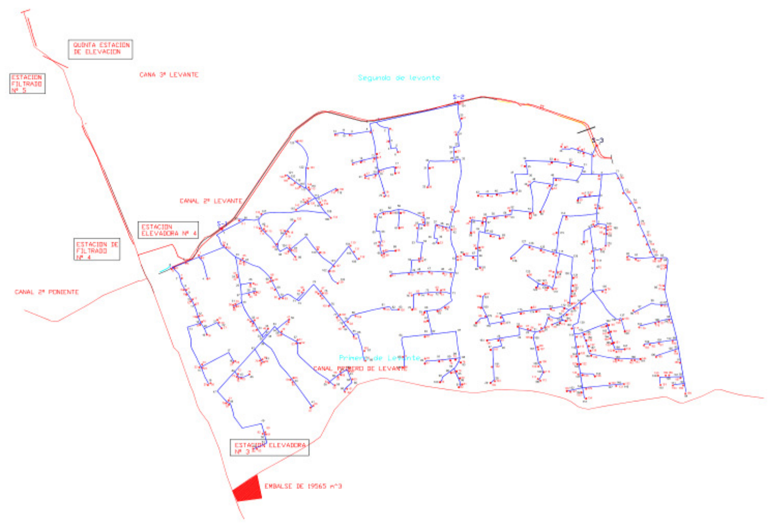


Figure 2. Pilot case topography

See the numerical results in [**1**]. The testbed is available upon request.

Acknowledgments

This research has been partially supported by the grant PLANIN MTM2009-14087-C04-01 from the Spanish Ministry of Science and Innovation, and the grant RIESGOS CM from Comunidad de Madrid.

References

1. M. Almiñana, L.F. Escudero, M. Landete, J.F. Monge, A.Rabasa and J. Sánchez-Soriano. On a mixed 0–1 separable nonlinear approach for water irrigation scheduling *IIE Tansactions* 40:398-405, (2008).
2. C.F. Colebrook. Turbulent flow in pipes, with particular reference to the transmition region between the smooth and rough pipe laws. *Journal of the Institute of Civil Engineers* 11. 133–156 (1938).
3. T.G. Lester Calculating Pressure Drops in Piping Systems. *ASHRAE Journal* 44, 41–43 (2002).
4. D.H. Yoo and V.P. Singh. Explicit Design of Commercial Pipes with no Secundary Losses. *Journal of Irrigation and Drainage Engineering* 130, 437–440 (2004).

Extending SCIP for solving MIQCPs

Timo Berthold[1] **Ambros M. Gleixner**[1] **Stefan Heinz**[1] **Stefan Vigerske**[2]

[1] Zuse Institute Berlin
Takustraße 7
14195 Berlin, Germany
`{berthold,gleixner,heinz}@zib.de`

[2] Humboldt-Universität zu Berlin
Unter den Linden 6
10099 Berlin, Germany
`stefan@math.hu-berlin.de`

1. Introduction

In recent years, substantial progress has been made in the solvability of generic *mixed integer programs (MIPs)* [**2, 9**] and in the extension of successful MIP solving techniques to the more general case of *mixed integer nonlinear programs (MINLPs)* [**1, 4, 10**]. Similarly, integrating *constraint programming (CP)* and mixed integer programming has proven to be effective for the solution of optimization problems that were intractable with either of the two methods alone, for an overview see [**15, 23**]. The paradigm of *constraint integer programming (CIP)* [**2, 3**] combines modeling and solving techniques from the fields of CP, MIP, and *satisfiability testing (SAT).*

The goal of this poster is to show, how a framework for CIPs can be extended towards a competitive solver for *mixed integer quadratically constrained programs (MIQCPs).* This allows to utilize the complete power of already existing MIP and CP technologies for handling the linear and the discrete parts of the problem. Also an experimental support for convex MINLPs is discussed shortly. We use the branch-cut-and-price framework SCIP (Solving Constraint Integer Programs), which incorporates the idea of CIP and implements several state-of-the-art techniques for MIP solving. Due to its design, it can be easily customized, e.g., by adding problem specific separation, presolving, or domain propagation algorithms.

SCIP solves CIPs by a branch-and-bound algorithm. The problem is recursively split into smaller subproblems, thereby creating a branching tree and implicitly enumerating all potential solutions. At each subproblem, domain propagation may exclude further values from the variables' domains, and a relaxation may be solved to achieve a local lower bound (assuming minimization problems). The relaxation may be strengthened by adding further valid constraints. In case of an infeasible subproblem, conflict analysis is performed to learn additional valid constraints. Primal heuristics are used as supplementary methods to improve the upper bound. In the context of this article, the relaxation employed in SCIP is a linear program (LP).

2. Algorithm

In the following we sketch some extensions to SCIP that target on MIQCPs and convex MINLPs.

A constraint handler defines the semantics and the algorithms to process constraints of a certain class. Each constraint handler has to implement an enforcement method. In enforcement, the handler has to decide whether a given solution, e.g., the optimum of a relaxation, satisfies all of its constraints. If the solution violates one or more constraints, the handler may resolve the infeasibility by adding another constraint, performing a domain reduction, or a branching. For speeding up the computation, a constraint handler may further implement additional features like presolving, cut separation, and domain propagation for its particular class of constraints.

2.1. A constraint handler for quadratic constraints

In the following, we discuss the presolving, separation, propagation, and enforcement algorithms that are used in our handler for quadratic constraints.

Presolving. During the presolving phase, bounds on the variables are tightened and a set of reformulations and simplifications are tried. For products of a *binary variable* with a linear term, a well-known linearization technique based on adding auxiliary variables and linear constraints is applied. Constraints of the form $\sum_{i=1}^{k}(\alpha_i x_i)^2 \leqslant (\beta y)^2$, $y \geqslant 0$, where $\alpha_i \in \mathbb{Q}$, $i = 1\ldots,k$, and $\beta \in \mathbb{Q}$ are recognized as *second-order cone constraints* and handled by the corresponding constraint handler, cf. Section 2.2. Constraints of the form $\ell \leqslant xy \leqslant u$, $\ell, u \in \mathbb{Q} \cup \{\pm\infty\}$, with either $y > 0$ or $y < 0$, are reformulated by division by y and handled as *univariate nonlinear constraints* (linearization for convex side, secants for concave side). After the presolving phase is finished, each quadratic constraint is checked for *convexity* by computing the sign of the minimal eigenvalue of the coefficient matrix A.

Separation. If the current LP solution $\tilde{x}$ violates some constraints, a constraint handler may add valid cutting planes in order to strengthen the formulation. For a violated convex constraint, this is always possible by linearizing the constraint function at $\tilde{x}$. For a violated nonconvex constraint, we currently underestimate each term of a quadratic function $\sum_{i,j} a_{i,j}x_i x_j$ separately by a linearization for $a_{i,i}x_i^2$ with $a_{i,i} > 0$, a secant for $a_{i,i}x_i^2$ with $a_{i,i} < 0$, and a McCormick underestimator [**17**] for a bilinear term, respectively. If a linear inequality generated by this method does not cut off the current LP solution $\tilde{x}$, the infeasibility has to be resolved in enforcement.

Propagation. In the domain propagation call, the constraint handler may infer deductions of the variables' local domains. Domain deductions can yield stronger linear underestimators in the separation procedures, they may cut off nodes due to infeasibility of a constraint, and can result in further domain deductions on other constraints. For quadratic constraints, we implemented an interval-arithmetic based method similar to [**14**].

Enforcement. In the enforcement call, the constraint handler has to check whether the current LP solution $\tilde{x}$ is feasible for the constraints of the constraint handler. If it is not feasible, it can resolve this infeasibility. We have configured SCIP to call the enforcement method of the quadratic constraint handler with a lower priority than the enforcement method for the handler of integrality constraints. Thus, at the point where quadratic constraints are enforced, all integer variables take an integral value in the LP optimum $\tilde{x}$. For a violated quadratic constraint, we perform a spatial branching operation if a domain propagation step does not cut off the current node and the separation method does not cut off $\tilde{x}$ from the LP relaxation. We use a pseudo-cost based branching rule as suggested in [**4**] to select the branching variable.

2.2. A constraint handler for second-order cone constraints

Constraints of the form $\sqrt{\sum_{i=1}^{k}(\alpha_i x_i)^2} \leqslant \beta y$, where $\beta y \geqslant 0$, are handled by the constraint handler for second-order cone (SOC) constraints. These constraints are enforced by separation only. If the current LP solution $\tilde{x}$ violates some SOC constraints, then we add the gradient-based cut $\gamma + \gamma^{-1}(\sum_{i=1}^{k} \alpha_i \tilde{x}_i (x_i - \tilde{x}_i)) - \beta y \leqslant 0$, where $\gamma := \sqrt{\sum_{i=1}^{k}(\alpha_i \tilde{x}_i)^2}$, which is violated by $\tilde{x}$. We also experimented with adding a linear outer-approximation of the SOC as suggested in [**5**] during presolve, but did not observe a computational benefit for the instances in our testset.

2.3. A constraint handler for convex nonlinear constraints

Experimental support for convex nonlinear constraints consists of separating the solution of the LP relaxation by generating gradient-based linear outer-approximations of convex functions.

2.4. Primal Heuristics

Three *large neighborhood search* heuristics for MINLPs have been implemented in SCIP so far.

NLP local search (Fix-and-NLP). When solving MIQCPs or convex MINLPs, we still make use of all default MIP primal heuristics of SCIP [**6**]. Since most of them aim at finding good integer-feasible solutions for the LP relaxation, they usually construct a point $\hat{x}$ which is feasible for the MIP relaxation, but violate some of the nonlinear constraints. However, we utilize $\hat{x}$ as starting point for a local search in the NLP obtained from the MINLP by fixing all integer variables to the values of $\hat{x}$. Each feasible solution of this NLP also is a feasible solution of the MINLP.

RENS (Fix-and-MINLP). Furthermore, we implemented an extended form of the relaxation enforced neighborhood search (RENS) heuristic [**7**]. At some node of the branch-and-bound-tree, this heuristic creates a sub-MINLP problem by fixing all integer variables which take an integral value in an optimal solution of the LP relaxation and restricts the bounds of all integer variables with fractional LP solution value to the two nearest integral values. This – hopefully much easier –

sub-MINLP is then partially solved by a separate SCIP instance. Obviously, each feasible solution of the sub-MINLP is a feasible solution of the original MINLP.

Undercover (Fix-and-MIP). The heuristic "undercover" [**8**] is based on the observation that it often suffices to fix only a comparatively small number of variables in an MINLP such as to yield a MIP subproblem. Every solution of such a sub-MIP is then a feasible solution for the original MINLP. The variables to fix are chosen by solving a set covering problem, which aims at minimizing the number of variables to fix. The values for the fixed variables are taken from the solution of the LP relaxation.

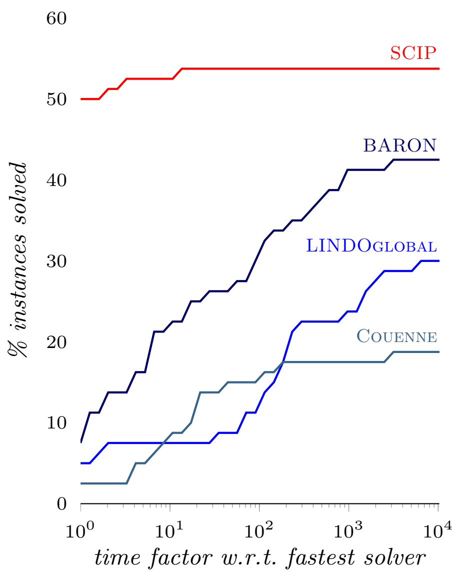
3. Numerical Results

We conducted numerical experiments on several different testsets. The first is a testset of H. Mittelmann of *mixed integer quadratic programs (MIQPs)* [**18**], i.e., problems with a quadratic objective function and linear constraints. Second, we have selected a testset of *mixed integer conic programs (MICPs)*, which have been formulated as MIQCP. They represent three different classes of portfolio optimization problems and have been introduced in [**20**]. Third, we have assembled a testset of general MIQCPs from the MINLPLib [**12**], an IBM-CMU project on MINLP [**13**], and truss structure design problems from [**23**]. Finally, we used a testset of *convex MINLPs* from [**11**].

For our benchmark, we ran SCIP 1.2.0.7 using CPLEX 11.2.1 [**16**] as LP solver and IPOPT 3.8 [**21**] (with MA27 as linear solver) as NLP solver for the heuristics. For comparison we used solvers as provided by GAMS 23.3.2 (except for COUENNE and BONMIN, which we compiled by ourself). For MIQCPs, we ran BARON 9.0.2 [**19**] (with CPLEX as LP solver and MINOS as NLP solver), COUENNE 0.3 [**4**] (with CLP as LP solver and IPOPT with MA27 as NLP solver), CPLEX 12.1, LINDOGLOBAL 6.0.1 (with a limited license), and Mosek 6.0.0.55. For convex MINLPs, we ran AlphaECP 1.75.04 [**22**] (with CPLEX as MIP solver and CONOPT as NLP solver), BONMIN 1.3 [**10**] (with CLP as LP solver and IPOPT with MA27 as NLP solver), DICOPT 2x-C (with CPLEX as MIP solver and CONOPT as NLP solver), and SBB (with CONOPT as NLP solver).

All solvers were run with a time limit of one hour, a final gap tolerance of 10^{-4}, and a feasibility tolerance of 10^{-6} on a 2.5 GHz Intel Core2 Duo CPU with 4 GB RAM and 6 MB Cache. Further, for AlphaECP the option `ECPstrategy` was set to 1, for DICOPT the option `stop` was set to 1 and `maxcycles` was set to 10000, for SBB the option `memnodes` was set to 9999999, and for SCIP when solving convex MINLPs, an option was set to indicate that these models are convex. BONMIN was run with three algorithmic options. BONMIN-BB is an NLP-based branch-and-bound similar to SBB, BONMIN-QG is an LP-based branch-and-bound algorithm similar to the one that SCIP implements, and BONMIN-Hyb is an hybrid of an NLP-based and an LP-based branch-and-bound algorithm.

Figure 1 shows performance profiles for MIQCPs and convex MINLPs. The performance profile for MIQCPs was created for all 80 instances from the first three testsets where at least two solvers took more than 10 seconds and which fit into the license limit of LINDOGLOBAL. Detailed problem statistics and results can be found in the appendix.



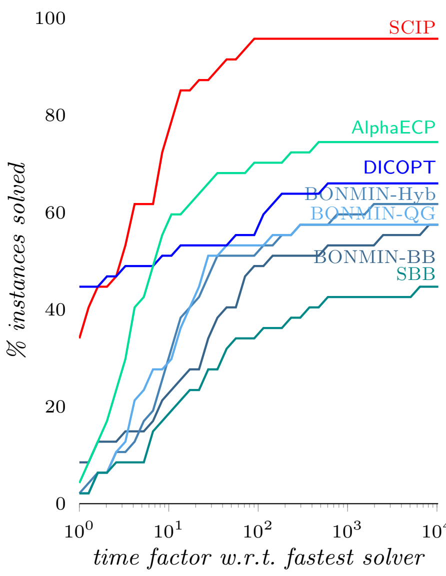


Figure 1. Performance profiles for MIQCPs (left) and convex MINLPs (right).

Acknowledgments

This research was funded by the DFG Research Center MATHEON, Project B20. We thank GAMS Development Corp. for providing us with evaluation licenses.

References

1. K. Abhishek, S. Leyffer, and J.T. Linderoth. FilMINT: An outer-approximation-based solver for nonlinear mixed integer programs. Technical Report ANL/MCS-P1374-0906, Argonne National Laboratory, Mathematics and Computer Science Division, 2006.
2. T. Achterberg. *Constraint Integer Programming*. PhD thesis, TU Berlin, 2007.
3. T. Achterberg, T. Berthold, T. Koch, and K. Wolter. Constraint integer programming: A new approach to integrate CP and MIP. In L. Perron and M.A. Trick, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 5th International Conference, CPAIOR 2008*, volume 5015 of *LNCS*, pages 6–20. Springer, 2008.
4. P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter. Branching and bounds tightening techniques for non-convex MINLP. *Optimization Methods and Software*, 24(4-5):597–634, 2009.
5. A. Ben-Tal and A. Nemirovski. On polyhedral approximations of the second-order cone. *Math. Oper. Res.*, 26(2):193–205, 2001.
6. T. Berthold. Primal heuristics for mixed integer programs. Master's thesis, TU Berlin, 2006.
7. T. Berthold. RENS – relaxation enforced neighborhood search. ZIB-Report 07-28, Zuse Institute Berlin, 2007.

8. T. Berthold and A.M. Gleixner. Undercover – a primal heuristic for MINLP based on sub-MIPs generated by set covering. ZIB-Report 09-40, Zuse Institute Berlin, December 2009.
9. R.E. Bixby, M. Fenelon, Z. Gu, E. Rothberg, and R. Wunderling. MIP: theory and practice – closing the gap. In M.J.D. Powell and S. Scholtes, editors, *System Modelling and Optimization: Methods, Theory and Applications*, pages 19–50. Kluwer, 2000.
10. P. Bonami, L.T. Biegler, A.R. Conn, G. Cornuéjols, I.E. Grossmann, C.D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, and A. Wächter. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5:186–204, 2008.
11. P. Bonami, M. Kilinç, and J. Linderoth. Algorithms and software for convex mixed integer nonlinear programs. available online `http://www.optimization-online.org/DB_HTML/2009/10/2429.html`, 2009.
12. M.R. Bussieck, A.S. Drud, and A. Meeraus. MINLPLib - A Collection of Test Models for Mixed-Integer Nonlinear Programming. *INFORMS Journal on Computing*, 15(1):114–119, 2003.
13. CMU-IBM MINLP Project. `http://egon.cheme.cmu.edu/ibm/page.htm`.
14. F. Domes and A. Neumaier. Constraint propagation on quadratic constraints. available online at `http://www.mat.univie.ac.at/~dferi/publ`, 2008.
15. J. Hooker. *Integrated Methods for Optimization*. International Series in Operations Research & Management Science. Springer, New York, 2007.
16. IBM. CPLEX. `http://www-01.ibm.com/software/integration/optimization/cplex/`.
17. G.P. McCormick. Computability of global solutions to factorable nonconvex programs: Part I-Convex Underestimating Problems. *Math. Prog.*, 10:147–175, 1976.
18. H. Mittelmann. MIQP test instances. `http://plato.asu.edu/ftp/miqp.html`.
19. M. Tawarmalani and N.V. Sahinidis. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*. Kluwer Academic Publishers, 2002.
20. J.P. Vielma, S. Ahmed, and G.L. Nemhauser. A lifted linear programming branch-and-bound algorithm for mixed integer conic quadratic programs. *INFORMS Journal on Computing*, 20(3):438–450, 2008.
21. A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Math. Prog.*, 106(1):25–57, 2006.
22. T. Westerlund and K. Lundquist. Alpha-ECP, version 5.04. an interactive MINLP-solver based on the extended cutting plane method. Technical Report 01-178-A, Process Design Laboratory, Åbo Akademi University, Åbo, Finland, 2003. `http://abo.fi./~twesterl/A-ECPManual.pdf`.
23. T. Yunes, I.D. Aron, and J. Hooker. An integrated solver for optimization problems. Technical report, School of Business Administration, University of Miami, 2008.

Appendix: Detailed numerical experiments

In the tables on problem statistics, the "presolved problem" columns show statistics about the MIQCP or MINLP after SCIP has applied its presolving routines. The columns "vars", "int", and "bin" show the number of all variables, the number of integer variables, and the number of binary variables, respectively. The columns "linear", "quad", "soc", "nonlin" show the number of linear, quadratic, second-order cone, and nonlinear (other than quadratic or soc) constraints, respectively. The column "conv" indicates whether all nonlinear constraints of the presolved MIQCP or MINLP are convex or whether at least one of them is nonconvex.

In the tables with computational results, each entry shows the number of seconds to solve a problem, or the lower and upper bounds at termination, if the problem was not solved.

Results on Mittelmann MIQPs

Table 1 presents the 25 instances from the MIQP testset [**18**]. Note that we consider the `clay*` instances in the general MIQCP testset, since they are not MIQPs. We observe, that due to the reformulation for products with binary variables, 15 instances could be reformulated as mixed integer linear programs in the presolving state.

Table 2 compares the performance of BARON, Couenne, CPLEX, LINDOGlobal (as far as our license allowed), and SCIP on this testset. Note that some of the instances are nonconvex before applying the reformulation described in Section 2.1, so that we did not apply solvers which have only been designed for convex problems. Instance `ivalues` is the only instance that cannot be handled by CPLEX due to nonconvexity.

Results on second-order cone problems

The MICP testset consists of three types of optimization problems, see Table 3. The instances `classical_XXX_YY` contain only one convex quadratic constraint of the form $\sum_{j=1}^{k} x_j^2 \leqslant u$ for some $u \in \mathbb{Q}$, where `XXX` stand for the dimension k and `YY` is a problem index. The instances `robust_XXX_YY` and `shortfall_XXX_YY` additionally contain one, respectively two, SOC constraints of dimension k.

Table 4 compares the performance of BARON, Couenne, CPLEX, LINDOGlobal, Mosek, and SCIP on this testset.

Results on general MIQCPs

For the general MIQCP testset, we took 42 instances from the MINLPLib [**12**], six constrained layout problems (`clay*`) from [**13**], and twelve truss structure design problems (`*bar*`) from [**15**], see Table 5. The instances `lop97ic`, `lop97icx`, `pb302035`, `pb351535`, `qap`, and `qapw` were transformed into MIPs during presolving. For the instances `200bar`, `nuclear14a`, and `nuclear14b`, a simple bilinear constraint of the form $\ell \leqslant xy \leqslant u$ with $y > 0$ or $y < 0$ was reformulated by division by y.

Table 6 compares the performance of BARON, Couenne, LINDOGlobal, and SCIP on this testset. The additional column "PP" indicates which instances participated in the generation of the performance profile in Figure 1.

Results on convex MINLPs

For the convex MINLP testset, we reproduced the testset from the recent paper [**11**], see Table 7.

Table 8 compares the performance of AlphaECP, BONMIN in three variants, DICOPT, SBB, and SCIP on this testset.

Table 1. Problem statistics for MIQP testset from Mittelmann.

instance	original problem					presolved problem					
	vars	int	bin	linear	quad	vars	int	bin	linear	quad	conv
`iair04`	8905	0	8904	823	1	12848	0	7362	17464	0	✓
`iair05`	7196	0	7195	426	1	10574	0	6117	14218	0	✓
`ibc1`	1752	0	252	1913	1	866	0	252	1438	0	✓
`ibell3a`	123	29	31	104	1	129	29	31	161	1	✓
`ibienst1`	506	0	28	576	1	473	0	28	592	0	✓
`icap6000`	6001	0	6000	2171	1	7323	0	5865	6362	0	✓
`icvxqp1`	10001	10000	0	5000	1	10003	9998	2	5006	1	✓
`ieilD76`	1899	0	1898	75	1	2685	0	1898	3168	0	✓
`ilaser0`	1003	151	0	2000	1	1003	151	0	1000	1	✓
`imas284`	152	0	150	68	1	228	0	150	299	0	✓
`imisc07`	261	0	259	212	1	360	0	238	598	0	✓
`imod011`	10958	1	96	4480	1	8963	1	96	2730	1	✓
`inug06-3rd`	2887	0	2886	3972	1	3709	0	2886	7779	0	✓
`inug08`	1633	0	1632	912	1	2217	0	1632	3076	0	✓
`iportfolio`	1201	192	775	201	1	1201	192	775	201	1	✓
`iqap10`	4151	0	4150	1820	1	5879	0	4150	9047	0	✓
`iqiu`	841	0	48	1192	1	871	0	48	1285	0	✓
`iran13x13`	339	0	169	195	1	468	0	169	585	0	✓
`iran8x32`	513	0	256	296	1	651	0	256	713	0	✓
`isqp0`	1001	50	0	249	1	1001	50	0	249	1	✓
`isqp1`	1001	0	100	249	1	1068	0	100	480	1	
`isqp`	1001	50	0	249	1	1001	50	0	249	1	✓
`iswath2`	6405	0	2213	483	1	8007	0	2213	5631	0	✓
`itointqor`	51	50	0	0	1	51	50	0	0	1	✓
`ivalues`	203	202	0	1	1	203	202	0	1	1	

Table 2. Results for MIQP testset from Mittelmann.

instance	BARON	COUENNE	CPLEX	LINDOGLOBAL	SCIP
`iair04`	$[-\infty, \infty]$	fail	**37.52**	license limit	206.75
`iair05`	$[-\infty, \infty]$	$[25886, \infty]$	**30.71**	license limit	103.56
`ibc1`	$[1.792, 3.72]$	$[1.696, 3.98]$	895.54	$[-\infty, \infty]$	**49.65**
`ibell3a`	58.95	198.90	**3.96**	$[869872, 878785]$	14.58
`ibienst1`	1048.04	1477.19	2836.05	$[15.47, 51.89]$	**32.62**
`icap6000`	$[-2448496, -2441852]$	fail	**6.28**	license limit	6.34
`icvxqp1`	$[324603, 613559]$	fail	$[\mathbf{327522}, \mathbf{410439}]$	license limit	$[0, 4451398]$
`ieilD76`	$[729.5, 1081]$	$[808.3, 898.5]$	**13.50**	$[-\infty, \infty]$	40.92
`ilaser0`	$[-\infty, \infty]$	fail	$[2409925, \mathbf{2412734}]$	license limit	fail
`imas284`	$[89193, 92241]$	3139.12	**4.36**	$[-\infty, \infty]$	30.38
`imisc07`	$[2432, 2814]$	$[1696, 3050]$	70.02	$[0, 2815]$	**49.18**
`imod011`	$[-3.823, -\mathbf{3.843}]$	fail	$[-\infty, \infty]$	license limit	**328.60**
`inug06-3rd`	$[177.8, \mathbf{1434}]$	fail	$[527.2, \mathbf{1434}]$	license limit	$[\mathbf{1114}, \infty]$
`inug08`	$[1451, 14696]$	$[683.1, \infty]$	2126.68	$[-\infty, \infty]$	**23.50**
`iportfolio`	$[-\infty, 0]$	$[-\mathbf{0.4944}, \infty]$	$[-\mathbf{0.4944}, -\mathbf{0.4937}]$	$[-\infty, \infty]$	$[-0.5252, 0]$
`iqap10`	$[-\infty, \infty]$	$[329.8, \infty]$	1411.26	license limit	**894.97**
`iqiu`	$[-402.5, -108.6]$	$[-357.5, -126.3]$	91.77	$[-773.6, -113.5]$	**70.02**
`iran13x13`	$[2930, 3355]$	$[3014, 3476]$	**20.02**	$[2743, 3347]$	49.53
`iran8x32`	$[5013, 5454]$	$[5034, 5629]$	25.24	$[4956, 5326]$	**17.14**
`isqp0`	$[-\infty, \infty]$	$[-\infty, -20137]$	$[-\mathbf{20338}, -\mathbf{20320}]$	$[-\infty, \infty]$	$[-\infty, -19895]$
`isqp1`	$[-\infty, \infty]$	$[-\infty, -18801]$	$[-\mathbf{19028}, -\mathbf{18993}]$	$[-\infty, \infty]$	$[-\infty, -17883]$
`isqp`	$[-\infty, \infty]$	**1312.56**	$[-21071, -\mathbf{21001}]$	$[-\infty, \infty]$	$[-\infty, \infty]$
`iswath2`	$[335.6, 661.9]$	$[335.9, 411.8]$	212.15	license limit	**210.00**
`itointqor`	$[-\infty, -1146]$	fail	$[-1150, -\mathbf{1147}]$	**394.10**	$[-\infty, 0]$
`ivalues`	$[-12.88, -0.4168]$	$[-\mathbf{6.054}, -\mathbf{1.056}]$	fail	$[-\infty, \infty]$	$[-169.4, 0]$

Table 3. Problem statistics for MICP testset.

instance	original problem					presolved problem					
	vars	int	bin	linear	quad	vars	int	bin	linear	quad	SOC
`classical_40_0`	120	0	40	82	1	120	0	40	82	1	0
`classical_40_1`	120	0	40	82	1	120	0	40	82	1	0
`classical_50_0`	150	0	50	102	1	150	0	50	102	1	0
`classical_50_1`	150	0	50	102	1	150	0	50	102	1	0
`classical_200_0`	600	0	200	402	1	600	0	200	402	1	0
`classical_200_1`	600	0	200	402	1	600	0	200	402	1	0
`robust_40_0`	163	0	41	124	2	163	0	41	124	1	1
`robust_40_1`	163	0	41	124	2	163	0	41	124	1	1
`robust_50_0`	203	0	51	154	2	203	0	51	154	1	1
`robust_50_1`	203	0	51	154	2	203	0	51	154	1	1
`robust_100_0`	403	0	101	304	2	403	0	101	304	1	1
`robust_100_1`	403	0	101	304	2	403	0	101	304	1	1
`robust_200_0`	803	0	201	604	2	803	0	201	604	1	1
`robust_200_1`	803	0	201	604	2	803	0	201	604	1	1
`shortfall_40_0`	164	0	41	125	2	164	0	41	125	0	2
`shortfall_40_1`	164	0	41	125	2	164	0	41	125	0	2
`shortfall_50_0`	204	0	51	155	2	204	0	51	155	0	2
`shortfall_50_1`	204	0	51	155	2	204	0	51	155	0	2
`shortfall_100_0`	404	0	101	305	2	404	0	101	305	0	2
`shortfall_100_1`	404	0	101	305	2	404	0	101	305	0	2
`shortfall_200_0`	804	0	201	605	2	804	0	201	605	0	2
`shortfall_200_1`	804	0	201	605	2	804	0	201	605	0	2

Table 4. Results for MICP testset.

instance	BARON	COUENNE	CPLEX	LINDOGLOBAL	Mosek	SCIP
`classical_40_0`	207.24	178.72	**1.60**	38.25	12.79	22.90
`classical_40_1`	7.09	114.49	**1.01**	217.55	22.28	6.55
`classical_50_0`	[−0.09191, −0.09074]	[−0.09447, −0.09054]	**41.33**	[−0.09572, −0.09074]	161.17	2866.56
`classical_50_1`	250.46	[−0.09595, −0.09459]	**7.38**	[−0.09593, −0.09476]	30.62	200.31
`classical_200_0`	[−0.1247, −0.1077]	[−0.1255, −0.0951]	[−**0.1231**, −**0.1106**]	[−0.1256, −0.08574]	[−0.124, −0.1103]	[−0.1285, −0.1081]
`classical_200_1`	[−0.1269, −0.1149]	[−0.1283, −0.1036]	[−**0.1257**, −**0.1164**]	[−0.1284, −0.1093]	[−0.1266, −0.1162]	[−0.13, −**0.1164**]
`robust_40_0`	3473.03	[−0.09706, −0.07602]	**0.67**	3600.95	1.37	4.04
`robust_40_1`	1752.70	[−0.116, −0.07646]	**0.64**	249.72	2.88	2.25
`robust_50_0`	[−0.08615, −0.08609]	[−0.1263, −0.0861]	1.88	165.45	**0.90**	1.17
`robust_50_1`	[−0.08574, −0.08569]	[−0.1274, −**0.08578**]	**2.32**	506.36	3.18	9.00
`robust_100_0`	[−0.1013, −0.0932]	[−0.1514, −**0.09753**]	[−0.1043, −0.09721]	[−0.1542, −0.08833]	1210.86	**1140.34**
`robust_100_1`	[−0.07501, −0.0703]	[−0.1257, −0.0677]	525.14	[−0.1269, 0]	**292.21**	628.80
`robust_200_0`	[−0.1722, −0.1363]	fail	[−**0.145**, −**0.1411**]	[−1, 0]	[−0.1468, −**0.1411**]	[−0.1456, −**0.1411**]
`robust_200_1`	[−0.1477, −0.1424]	[−0.1995, −0.1377]	[−**0.1454**, −0.1425]	[−1, 0]	[−0.1456, −**0.1427**]	[−0.1473, −0.1421]
`shortfall_40_0`	102.17	333.76	247.99	1027.02	45.71	**15.44**
`shortfall_40_1`	5.99	133.49	5.71	288.12	13.72	**3.02**
`shortfall_50_0`	[−1.098, −1.095]	[−1.102, −1.095]	1913.00	[−1.104, −1.095]	**405.93**	1494.87
`shortfall_50_1`	91.84	[−1.103, −1.099]	**13.13**	[−1.104, −1.102]	21.73	14.63
`shortfall_100_0`	[−1.12, −**1.114**]	[−1.126, −1.102]	[−1.132, −1.112]	[−1.125, −1.114]	[−**1.116**, −**1.114**]	[−1.12, −**1.114**]
`shortfall_100_1`	[−1.109, −1.106]	[−1.113, −1.091]	3301.75	[−1.113, −1.105]	[−1.111, −1.106]	**2223.76**
`shortfall_200_0`	[−1.149, −1.12]	[−1.15, −1.094]	[−**1.146**, −1.125]	[−1.479, −1.08]	[−**1.146**, −**1.126**]	[−1.148, −1.125]
`shortfall_200_1`	[−1.15, −1.131]	[−1.152, −1.101]	[−1.15, −1.133]	[−1.361, −1.089]	[−1.151, −**1.135**]	[−**1.149**, −1.134]

Table 5. Problem statistics for general MIQCP testset.

instance	original problem					presolved problem						
	vars	int	bin	linear	quad	vars	int	bin	linear	quad	nonlin	convex
10bar1A	148	0	110	38	10	137	0	110	27	10	0	
10bar1B	148	0	110	38	10	137	0	110	27	10	0	
10bar1C	148	0	110	38	10	137	0	110	27	10	0	
10bar1D	148	0	110	38	10	137	0	110	27	10	0	
10bar2	176	0	110	56	20	154	0	110	34	20	0	
10bar3	148	0	110	38	10	138	0	110	28	10	0	
10bar4	176	0	110	56	20	156	0	110	36	20	0	
25bar	436	0	275	323	50	248	0	87	119	50	0	
72bar	1248	0	792	1000	144	623	0	175	320	144	0	
90bar	1560	0	990	1250	180	781	0	219	402	180	0	
108bar	1872	0	1188	1500	216	939	0	263	484	216	0	
200bar	7850	0	6000	4570	600	4532	0	2880	1175	593	7	
clay0203m	30	0	18	30	24	27	0	15	27	24	0	✓
clay0204m	52	0	32	58	32	48	0	28	54	32	0	✓
clay0205m	80	0	50	95	40	75	0	45	90	40	0	✓
clay0303m	33	0	21	30	36	31	0	19	29	36	0	✓
clay0304m	56	0	36	58	48	54	0	34	57	48	0	✓
clay0305m	85	0	55	95	60	81	0	51	93	60	0	✓
du-opt	21	13	0	9	1	21	13	0	5	1	0	✓
du-opt5	21	13	0	9	1	19	11	0	4	1	0	✓
ex1263	92	0	72	51	4	91	0	71	47	4	0	
ex1264	88	0	68	51	4	82	0	62	47	4	0	
ex1265	130	0	100	69	5	122	0	92	65	5	0	
ex1266	180	0	138	89	6	168	0	126	81	6	0	
ex4	37	0	25	5	26	37	0	25	4	26	0	
fac3	67	0	12	33	1	67	0	12	39	1	0	✓
feedtray2	88	0	36	137	147	300	0	12	1001	147	0	
lop97ic	1754	831	831	52	40	5228	708	708	11521	0	0	✓
lop97icx	987	831	68	48	40	488	68	68	1138	0	0	✓
meanvarx	36	0	14	44	1	30	0	12	36	1	0	✓
netmod_dol1	1999	0	462	3137	1	1993	0	462	3131	1	0	✓
netmod_dol2	1999	0	462	3080	1	1592	0	454	2637	1	0	✓
netmod_kar1	457	0	136	666	1	453	0	136	662	1	0	✓
netmod_kar2	457	0	136	666	1	453	0	136	662	1	0	✓
nous1	51	0	2	15	29	47	0	2	11	29	0	
nous2	51	0	2	15	29	47	0	2	11	29	0	
nuclear14a	992	0	600	49	584	1568	0	600	2377	368	192	
nuclear14b	1568	0	600	1225	560	1568	0	600	1225	368	192	
nuclear14	1562	0	576	624	602	986	0	576	48	602	0	
nuclearva	351	0	168	50	267	327	0	144	24	267	0	
nvs19	9	8	0	0	9	9	8	0	0	9	0	
nvs23	10	9	0	0	10	10	9	0	0	10	0	
pb302035	601	0	600	50	1	1199	0	600	1847	0	0	✓
pb351535	526	0	525	50	1	1048	0	525	1619	0	0	✓
product	1553	0	107	1793	132	446	0	92	450	82	0	
product2	2842	0	128	2597	528	480	0	128	338	128	0	
qap	226	0	225	30	1	449	0	225	702	0	0	✓
qapw	451	0	225	255	1	675	0	225	930	0	0	✓
sep1	29	0	2	25	6	19	0	2	15	6	0	
space25	893	0	750	210	25	767	0	716	118	25	0	
space25a	383	0	240	176	25	308	0	240	101	25	0	
spectra2	70	0	30	65	8	68	0	30	30	8	0	
tln5	35	30	5	25	5	35	30	5	20	5	0	
tln6	48	42	6	30	6	48	42	6	24	6	0	
tln7	63	56	7	35	7	63	56	7	28	7	0	
tln12	168	156	12	60	12	180	144	24	85	11	0	
tloss	48	42	6	47	6	46	42	4	39	6	0	
tltr	48	36	12	51	3	56	27	20	73	2	0	
util	146	0	28	164	4	32	0	12	13	4	0	
waste	2484	0	400	623	1368	1238	0	400	516	1230	0	

Table 6. Results for general MIQCP testset. The column "PP" indicates which instances participated for the performance profile in Figure 1.

instance	BARON	Couenne	LINDOGlobal	SCIP	PP
`10bar1A`	1.82	2.51	61.43	**1.15**	
`10bar1B`	1.69	2.73	58.26	**1.44**	
`10bar1C`	3.59	2.77	$[0, 1623]$	**2.75**	
`10bar1D`	13.83	13.58	$[0, 1623]$	**3.57**	✓
`10bar2`	121.22	**5.51**	$[0, 1976]$	10.29	✓
`10bar3`	**450.01**	2123.14	$[0, \mathbf{5157}]$	$[-\infty, 5181]$	✓
`10bar4`	3449.80	3702.63	$[2028, 6547]$	**941.70**	✓
`25bar`	707.04	fail	$[268.5, 387.1]$	**633.93**	✓
`72bar`	**949.26**	fail	$[71.58, 127.8]$	$[70.38, \infty]$	✓
`90bar`	$[90.13, \mathbf{104.3}]$	fail	$[-\infty, \infty]$	$[\mathbf{90.34}, \infty]$	✓
`108bar`	$[103.3, \infty]$	fail	$[-\infty, \infty]$	$[\mathbf{110.7}, \mathbf{249.3}]$	✓
`200bar`	$[\mathbf{7681}, \infty]$	fail	license limit	$[7353, \infty]$	✓
`clay0203m`	1.56	2.03	43.13	**0.13**	
`clay0204m`	48.25	4.79	85.50	**0.54**	✓
`clay0205m`	971.83	27.73	1162.00	**4.80**	✓
`clay0303m`	1.29	2.85	62.17	**0.37**	
`clay0304m`	14.77	16.31	187.38	**0.88**	✓
`clay0305m`	3584.73	27.65	1112.42	**7.30**	✓
`du-opt`	137.39	$[-8727, \infty]$	2204.70	**0.97**	✓
`du-opt5`	150.54	$[-2437, 9.012]$	697.10	**0.46**	✓
`ex1263`	**0.86**	2.83	249.39	0.97	
`ex1264`	0.97	1.49	325.47	**0.51**	
`ex1265`	1.19	11.57	$[0, 15.1]$	**0.66**	✓
`ex1266`	0.39	7.91	1.50	**0.12**	
`ex4`	1.54	8.98	28.15	**0.34**	
`fac3`	**0.20**	$[24208886, 31985546]$	44.26	0.57	✓
`feedtray2`	**0.12**	18.86	181.45	1.47	✓
`lop97ic`	$[2549, \infty]$	$[\mathbf{3826}, \infty]$	$[-\infty, \infty]$	$[2576, \mathbf{4358}]$	✓
`lop97icx`	$[2812, 4415]$	$[\mathbf{3903}, 4272]$	$[0, 5259]$	$[3612, \mathbf{4124}]$	✓
`meanvarx`	0.11	1.59	0.13	**0.08**	
`netmod_dol1`	$[-0.7817, -0.08079]$	$[-0.8333, -0.2579]$	license limit	$[\mathbf{-0.607}, \mathbf{-0.56}]$	
`netmod_dol2`	$[-0.6162, -0.1459]$	$[-0.6504, -0.1254]$	license limit	**50.85**	
`netmod_kar1`	508.78	$[-0.6209, -0.3697]$	$[-0.5549, -0.4198]$	**5.63**	✓
`netmod_kar2`	508.71	$[-0.621, -0.3697]$	$[-0.6626, -0.4198]$	**5.64**	✓
`nous1`	641.19	$[1.345, 1.567]$	**41.44**	$[0.9966, 1.567]$	✓
`nous2`	0.97	2.69	**0.36**	265.50	
`nuclear14a`	$[\mathbf{-12.26}, \infty]$	$[\mathbf{-12.26}, \infty]$	$[-\infty, \infty]$	$[-247, \mathbf{-1.074}]$	✓
`nuclear14b`	$[\mathbf{-2.078}, \mathbf{-1.107}]$	$[-2.234, \infty]$	$[-\infty, \infty]$	$[-197.8, -1.101]$	✓
`nuclear14`	$[-\infty, \infty]$	$[-\infty, -1.12]$	$[-\infty, \mathbf{-1.126}]$	$[-\infty, -1.122]$	✓
`nuclearva`	$[-\infty, \infty]$	$[-\infty, \mathbf{-1.005}]$	$[-\infty, \infty]$	$[-\infty, \infty]$	✓
`nvs19`	12.14	778.31	457.04	**0.26**	✓
`nvs23`	44.54	$[-1380, -1109]$	2533.14	**0.43**	✓
`pb302035`	$[-\infty, \infty]$	fail	$[-\infty, \infty]$	$[\mathbf{1133674}, \mathbf{3993292}]$	✓
`pb351535`	$[-\infty, \infty]$	fail	fail	$[\mathbf{1666134}, \mathbf{5069392}]$	✓
`product`	fail	fail	$[-2200, -2092]$	**89.14**	✓
`product2`	fail	fail	license limit	$[\mathbf{-2106}, \mathbf{-2099}]$	
`qap`	$[\mathbf{103040}, \mathbf{388250}]$	$[0, \infty]$	$[-\infty, \infty]$	$[26372, 414268]$	✓
`qapw`	$[\mathbf{265372}, \mathbf{391210}]$	$[0, \infty]$	$[0, 405354]$	$[30350, 405088]$	✓
`sep1`	**0.04**	0.33	0.14	0.08	
`space25a`	$[99.99, 490.2]$	$[35.09, \infty]$	$[\mathbf{330.6}, \mathbf{489.2}]$	$[73.42, \infty]$	✓
`space25`	$[\mathbf{84.91}, \mathbf{520.9}]$	$[42.68, \infty]$	$[33.07, 638.8]$	$[72.46, \infty]$	✓
`spectra2`	3.33	$[-1072, 19.2]$	109.91	**0.55**	✓
`tln5`	798.56	$[6.592, 10.3]$	174.02	**129.53**	✓
`tln6`	$[13.75, 15.3]$	$[7.801, 15.3]$	**182.23**	$[9.88, 15.3]$	✓
`tln7`	$[12.38, 15.6]$	$[5.038, 16.1]$	$[\mathbf{14.2}, 15.6]$	$[6.303, \mathbf{15.2}]$	✓
`tln12`	$[32.73, \infty]$	$[16.19, \infty]$	$[\mathbf{85.8}, 139.1]$	$[16.41, \mathbf{91.6}]$	✓
`tloss`	**0.10**	2.99	0.27	0.14	
`tltr`	0.14	1.61	1.43	**0.11**	
`util`	**0.28**	1.32	6.12	0.31	
`waste`	$[306.7, 712.3]$	fail	**1532.71**	$[308.2, 681.7]$	✓

Table 7. Problem statistics for convex MINLP testset.

instance	original problem					presolved problem					
	vars	int	bin	linear	nonlin	vars	int	bin	linear	quad	nonlin
BatchS101006M	279	0	129	1018	2	298	0	120	690	0	29
BatchS121208M	407	0	203	1510	2	430	0	192	1044	0	35
BatchS151208M	446	0	203	1780	2	472	0	192	1215	0	38
BatchS201210M	559	0	251	2326	2	590	0	240	1595	0	43
clay0303h	99	0	21	114	36	99	0	21	114	0	36
clay0304h	176	0	36	210	48	176	0	36	210	0	48
clay0304m	56	0	36	58	48	54	0	34	57	48	0
clay0305h	275	0	55	335	60	275	0	55	335	0	60
clay0305m	85	0	55	95	60	81	0	51	93	60	0
FLay04H	234	0	24	278	4	234	0	24	278	0	4
FLay05H	382	0	40	460	5	382	0	40	460	0	5
FLay05M	62	0	40	60	5	62	0	40	60	0	5
FLay06M	86	0	60	87	6	86	0	60	87	0	6
fo7_2	114	0	42	197	14	82	0	42	137	0	14
fo7	114	0	42	197	14	82	0	42	137	0	14
fo8	146	0	56	257	16	102	0	56	173	0	16
m6	86	0	30	145	12	62	0	30	101	0	12
m7	114	0	42	197	14	80	0	42	133	0	14
o7_2	114	0	42	197	14	90	0	42	153	0	14
RSyn0805H	308	0	37	426	3	141	0	36	169	0	3
RSyn0805M02M	360	0	148	763	6	194	0	106	563	0	6
RSyn0805M03M	540	0	222	1275	9	296	0	176	937	0	9
RSyn0805M04M	720	0	296	1874	12	406	0	246	1405	0	12
RSyn0810M02M	410	0	168	854	12	223	0	119	615	0	12
RSyn0810M03M	615	0	252	1434	18	342	0	198	1031	0	18
RSyn0820M	215	0	84	357	14	105	0	48	209	0	14
RSyn0830M04H	2344	0	496	4156	80	977	0	405	2067	0	80
RSyn0830M	250	0	94	405	20	132	0	57	249	0	20
RSyn0840M	280	0	104	456	28	157	0	65	291	0	28
SLay06H	343	0	60	435	1	343	0	60	435	1	0
SLay07H	477	0	84	609	1	477	0	84	609	1	0
SLay08H	633	0	112	812	1	633	0	112	812	1	0
SLay09H	811	0	144	1044	1	811	0	144	1044	1	0
SLay09M	235	0	144	324	1	235	0	144	324	1	0
SLay10M	291	0	180	405	1	291	0	180	405	1	0
sssd-10-4-3	68	0	52	30	12	68	0	52	30	0	12
sssd-12-5-3	95	0	75	37	15	95	0	75	37	0	15
sssd-15-6-3	132	0	108	45	18	132	0	108	45	0	18
Syn15M04M	340	0	120	762	44	192	0	80	409	0	44
Syn20M03M	315	0	120	657	42	190	0	82	375	0	42
Syn20M04M	420	0	160	996	56	259	0	115	572	0	56
Syn30M02M	320	0	120	564	40	185	0	77	320	0	40
Syn40M03H	1146	0	240	1914	84	344	0	173	637	0	84
Syn40M	130	0	40	198	28	104	0	33	162	0	28
uflquad-20-150	3021	0	20	3150	1	3021	0	20	3150	1	0
uflquad-30-100	3031	0	30	3100	1	3031	0	30	3100	1	0
uflquad-40-80	3241	0	40	3280	1	3241	0	40	3280	1	0

Table 8. Results for convex MINLP testset.

instance	AlphaECP	BONMIN-BB	BONMIN-Hyb	BONMIN-QG	DICOPT	SBB	SCIP
BatchS101006M	67.51	25.90	21.38	22.28	**9.30**	90.04	13.46
BatchS121208M	159.89	52.35	60.39	60.16	**5.08**	217.35	70.88
BatchS151208M	279.57	235.90	117.35	144.64	**10.38**	274.89	126.27
BatchS201210M	124.19	218.52	140.83	169.00	**9.15**	220.50	82.19
clay0303h	1.70	31.31	10.67	3.11	30.82	4.42	**0.68**
clay0304h	16.90	193.47	70.82	27.68	$[7088, 58357]$	228.26	**1.94**
clay0304m	12.78	63.50	48.29	8.61	$[7180, 58357]$	11.76	**0.85**
clay0305h	94.13	$[8070, 8092]$	70.47	14.41	1075.33	464.75	**13.08**
clay0305m	20.07	723.15	42.80	$[-\infty, 8289]$	2227.20	27.60	**7.26**
FLay04H	99.99	52.47	50.23	15.17	494.68	6.07	**4.84**
FLay05H	$[-\infty, 64.5]$	1993.43	1308.71	1080.98	$[57.96, 64.5]$	1809.22	**294.48**
FLay05M	$[-\infty, 64.5]$	593.36	284.01	246.33	$[61.99, 64.5]$	**1.00**	67.83
FLay06M	$[-\infty, 66.93]$	$[60, 66.93]$	$[60.88, 66.93]$	$[59.87, 66.93]$	$[55, 66.93]$	$[56.16, 66.93]$	**3254.95**
fo7_2	**16.76**	$[6.844, 19.96]$	81.40	68.38	$[13.66, 29.15]$	$[9.9, 17.75]$	49.43
fo7	**104.70**	$[8.346, 25.84]$	153.74	217.86	$[11.75, 33.68]$	$[8.195, 29.95]$	129.52
fo8	474.99	$[7.067, 30.22]$	961.07	2451.36	$[14.75, \infty]$	$[6.232, 46.72]$	**236.46**
m6	0.66	1057.83	13.67	3.14	**0.14**	868.92	4.90
m7	4.37	$[75.93, 106.8]$	73.39	20.79	**0.47**	$[83.48, 106.8]$	18.06
o7_2	773.09	$[23.89, 134.4]$	**632.51**	2344.03	$[67.55, \infty]$	$[55.67, 138.2]$	790.18
RSyn0805H	0.09	3.37	30.35	1.09	**0.05**	0.55	0.37
RSyn0805M02M	7.54	$[-2385, -2238]$	$[-\infty, -2153]$	$[-\infty, -2148]$	**3.45**	$[-2418, -2198]$	12.99
RSyn0805M03M	16.51	$[-3388, -3065]$	$[-\infty, -2625]$	$[-\infty, -2966]$	**7.99**	$[-3900, -2704]$	18.84
RSyn0805M04M	13.67	$[-7846, -7141]$	72.00	$[-\infty, -7140]$	**8.59**	$[-8191, -7163]$	23.12
RSyn0810M02M	10.81	$[-3194, -1710]$	$[-\infty, -1711]$	$[-\infty, -1682]$	**5.03**	$[-3327, -1585]$	44.82
RSyn0810M03M	38.24	$[-4143, -2704]$	$[-\infty, -2633]$	$[-\infty, -2655]$	**11.37**	$[-5826, -2486]$	46.01
RSyn0820M	0.62	$[-1228, -1150]$	$[-1150, -1033]$	$[-1317, -1150]$	**0.18**	$[-1479, -1116]$	1.53
RSyn0830M04H	11.42	398.73	122.20	75.31	5.39	676.27	**3.22**
RSyn0830M	1.31	$[-610, -509.9]$	$[-\infty, -497.9]$	$[-684, -506.7]$	**0.46**	$[-1567, -504.1]$	3.34
RSyn0840M	0.94	$[-822.4, -318.6]$	537.66	$[-664.8, -308.4]$	**0.31**	$[-1764, -115]$	2.74
SLay06H	602.70	4.20	26.99	**3.07**	33.41	143.73	16.74
SLay07H	2573.58	8.79	44.51	**7.52**	42.60	2988.40	47.88
SLay08H	$[-\infty, 84960]$	**13.71**	154.02	48.86	2220.36	$[82526, 85728]$	323.05
SLay09H	$[-\infty, 107806]$	**27.70**	268.53	555.13	$[105649, 108645]$	$[103146, 114342]$	1483.84
SLay09M	$[-\infty, 108521]$	**10.29**	46.16	202.57	797.99	10.57	91.21
SLay10M	$[-\infty, 131168]$	118.87	382.70	$[126913, 129580]$	$[121630, 132180]$	**37.01**	$[106361, 129580]$
sssd-10-4-3	1.23	14.59	1.11	$[-\infty, 174854]$	2.30	$[78151, 183184]$	**0.91**
sssd-12-5-3	14.04	58.70	12.63	23.08	21.64	$[100794, 315015]$	**2.02**
sssd-15-6-3	321.45	240.46	877.70	62.87	715.62	$[155130, 517747]$	**3.58**
Syn15M04M	0.54	356.32	$[-\infty, -4892]$	$[-\infty, -4924]$	**0.16**	52.90	1.17
Syn20M03M	0.51	$[-\infty, -2598]$	$[-2647, -2596]$	$[-\infty, -2615]$	**0.08**	$[-3207, -2647]$	1.46
Syn20M04M	1.17	$[-\infty, -3512]$	21.96	$[-3544, -3503]$	**0.13**	$[-5120, -3452]$	9.45
Syn30M02M	0.67	$[-681, -378.7]$	$[-\infty, -389.7]$	$[-\infty, -362.6]$	**0.12**	$[-2541, -356.9]$	1.56
Syn40M	0.17	1407.02	$[-\infty, -55.71]$	$[-205.4, -60.76]$	**0.04**	$[-492.8, -67.71]$	0.62
Syn40M03H	2.99	25.85	$[-\infty, -0]$	5.85	**1.08**	15.04	1.19
uflquad-20-150	$[-\infty, 630.2]$	**495.58**	$[258.4, 653.2]$	$[241.1, 596.9]$	$[240.8, 727.5]$	$[359.4, 1026]$	$[132.3, 4313]$
uflquad-30-100	$[-\infty, 415]$	**721.82**	$[163.6, 399.6]$	$[144.2, 430]$	$[147, 441.8]$	$[249.9, 558.5]$	$[97.56, 2953]$
uflquad-40-80	$[-\infty, 393.6]$	$[\mathbf{304.3}, \mathbf{348.1}]$	$[103.1, 1896]$	$[103.1, 1896]$	$[119.8, 440]$	$[221.5, 402.7]$	$[97.28, 905.5]$

Solving a general mixed-integer quadratic problem through convex reformulation: a computational study

Alain Billionnet Sourour Elloumi Amélie Lambert

CEDRIC-ENSIIE
1, square de la résistance
Evry, 91026 Cedex, France

{alain.billionnet,sourour.elloumi,amelie.lambert}@ensiie.fr

Abstract

Let (QP) be a mixed integer quadratic program that consists of minimizing a quadratic function subject to linear constraints. In this paper, we present a convex reformulation of (QP), i.e. we reformulate (QP) into an equivalent program, with a convex objective function. Such a reformulation can be solved by a standard solver that uses a branch and bound algorithm. This reformulation, that we call `MIQCR (Mixed Integer Quadratic Convex Reformulation)`, is the best one within a convex reformulation scheme, from the continuous relaxation point of view. It is based on the solution of an SDP relaxation of (QP). Computational experiences were carried out with instances of (QP) with one equality constraint. The results show that most of the considered instances, with up to 60 variables, can be solved within 1 hour of CPU time by a standard solver.

Keywords: mixed integer quadratic programming, convex reformulation, semidefinite programming, experiments.

1. Introduction

Consider the following linearly-constrained mixed-integer quadratic program:

$$
(QP)\left\{\begin{array}{llll}
Min & f(x) & & \\
s.t. & \sum_{i=1}^{N} a_{ri}x_i = b_r & r = \{1,\ldots,m\} & (33.1)\\
 & \sum_{i=1}^{N} d_{si}x_i \leqslant e_s & s = \{1,\ldots,p\} & (33.2)\\
 & 0 \leqslant x_i \leqslant u_i & i \in I & (33.3)\\
 & 0 \leqslant x_i \leqslant u_i & i \in J & (33.4)\\
 & x_i \in \mathbb{N} & i \in I & (33.5)\\
 & x_i \in \mathbb{R} & i \in J & (33.6)
\end{array}\right.
$$

where $I = \{1,\ldots,n\}$ is the sub-set of integer variable indices, $J = \{n+1,\ldots,N\}$ is the sub-set of real variable indices,

$$f(x) = x^T Qx + c^T x = \sum_{(i,j)\in I^2} q_{ij}x_ix_j + \sum_{(i,j)\in I\times J} 2q_{ij}x_ix_j + \sum_{(i,j)\in J^2} q_{ij}x_ix_j + \sum_{i\in I\cup J} c_ix_i$$

and $Q \in \mathbf{S}_N$ (space of symmetric matrices of order N), $c \in \mathbb{R}^N$, $A \in \mathbf{M}_{m,N}$ (space of $m \times N$ matrices), $b \in \mathbb{R}^m$, $D \in \mathbf{M}_{p,N}$, $e \in \mathbb{R}^p$, $u \in \mathbb{N}^N$.

We suppose that the sub-function of the products of real variables of $h(x)$: $\sum_{(i,j)\in J^2} q_{ij}x_ix_j$ is convex.

(QP) belongs to the class of $\mathcal{NP}$-hard problems [**3**]. Standard solvers [**5, 2**] can efficiently solve Mixed Integer Quadratic Programs (MIQP), but only in the specific case where $f(x)$ is convex. Thus, to solve (QP) by use of a standard solver, we choose to reformulate it into another program with a convex objective function. By convex reformulation, we mean to design a program, that is equivalent to (QP), and that has a convex objective function. In concrete terms, that will consist of perturbing the Q matrix of $f(x)$ in order to obtain a positive semidefinite matrix. In this work, we first define a convex reformulation scheme, and then we compute, within this scheme, the optimal convex reformulation in terms of continuous relaxation bound. To do it, we introduce new variables y_{ij}, and new linear constraints to enforce the equality $y_{ij} = x_ix_j$. These new variables will allow the perturbation of each term of matrix Q. In a sense, our approach mixes ideas of linearization and convexification.

In the rest of the paper, we present our approach, that we denote by `MIQCR` (`Mixed Integer Quadratic Convex Reformulation`). In Section 2, we propose a reformulation scheme of (QP) into an equivalent mixed-integer quadratic program $(QP_{\alpha,\beta})$ depending on a scalar parameter α, and on a matrix parameter β. In Section 3, we show how to compute α^* and β^*, the values of α and β that maximize the optimal value of the continuous relaxation of $(QP_{\alpha,\beta})$. We show that α^* and β^* can be deduced from the solution of a semidefinite relaxation of (QP). Finally, in Section 4, we evaluate `MIQCR` from the computational point of view. Our experiments are carried out on instances of (QP) with one equality constraint. Section 5 is a conclusion.

2. A convex reformulation scheme for mixed-integer quadratic programs

In this section, we rewrite (QP) into an equivalent mixed-integer quadratic program $(QP_{\alpha,\beta})$ with a convex objective function. The idea is to add to the initial objective function $f(x)$ the following functions that vanish on the feasible domain of (QP) under the assumption that $y_{ij} = x_i x_j$, $\forall (i,j) \in P$, where $P = \{(i,j) \in (I \times I) \cup (I \times J) \cup (J \times I)\}$.

- $\alpha \sum_{r=1}^{m} (\sum_{i \in I \cup J} a_{ri} x_i - b_r)^2$ where $\alpha \in \mathbb{R}$.
- $\sum_{(i,j)\in P} \beta_{ij}(x_i x_j - y_{ij})$, where $\beta_{ij} \in \mathbb{R}$ and $\beta_{ij} = \beta_{ji}$ $\forall (i,j) \in P$, or equivalently, we consider $\beta \in \mathbf{S}_N$ with $\beta_{ij} = 0$ $\forall (i,j) \in J^2$.

We obtain the following program $(QP_{\alpha,\beta})$:

$$(QP_{\alpha,\beta}) \left\{ \begin{array}{ll} Min & f_{\alpha,\beta}(x,y) \\ s.t. & (33.1)(33.2) \\ & x,y,z,t \in P_{xyzt} \end{array} \right.$$

where

$$f_{\alpha,\beta}(x,y) = f(x) + \sum_{(i,j)\in P} \beta_{ij}(x_i x_j - y_{ij}) + \alpha \sum_{r=1}^{m} (\sum_{i \in I \cup J} a_{ri} x_i - b_r)^2$$

and P_{xyzt} is the following set:

$$P_{xyzt} \left\{ x,y,z,t : \begin{array}{lll} (33.3)(33.4) & & \\ x_i = \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k t_{ik} & i \in I & (33.7) \\ z_{ijk} \leqslant u_j t_{ik} & (i,k) \in E,\ j \in I \cup J & (33.8) \\ z_{ijk} \leqslant x_j & (i,k) \in E,\ j \in I \cup J & (33.9) \\ z_{ijk} \geqslant x_j - u_j(1 - t_{ik}) & (i,k) \in E,\ j \in I \cup J & (33.10) \\ z_{ijk} \geqslant 0 & (i,k) \in E,\ j \in I \cup J & (33.11) \\ y_{ij} = \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k z_{ijk} & (i,j) \in I \times I \cup J & (33.12) \\ t_{ik} \in \{0,1\} & (i,k) \in E & (33.13) \\ y_{ij} = y_{ji} & (i,j) \in P & (33.14) \\ y_{ij} \geqslant x_i u_j + x_j u_i - u_i u_j & (i,j) \in P & (33.15) \\ y_{ii} \geqslant x_i & i \in I & (33.16) \\ y_{ij} \leqslant u_i x_j & (i,j) \in I \times J & (33.17) \end{array} \right.$$

with $E = \{(i,k) : i = 1, \ldots, n,\ k = 0, \ldots \lfloor log(u_i) \rfloor\}$.

It is proven in [4] that $(QP_{\alpha,\beta})$ is equivalent to (QP).

3. Computing the best convex reformulation : the MIQCR method

In this section, we show how to compute, by semidefinite programming, values of α^* and β^* that make $f_{\alpha^*,\beta^*}(x,y)$ convex, and that maximize the continuous relaxation value of (QP_{α^*,β^*}), that is to say we have to solve the following problem (CP):

$$(CP) : \max_{\substack{\alpha \in \mathbb{R}, \beta \in \mathbf{S}_N \\ \beta_{ij} = 0,\ (i,j) \in J^2 \\ Q_{\alpha,\beta} \succeq 0}} \{v(\overline{QP}_{\alpha,\beta})\}$$

where $(\overline{QP}_{\alpha,\beta})$ is the continuous relaxation of $(QP_{\alpha,\beta})$, $v(\overline{QP}_{\alpha,\beta})$ is the optimal solution value of $(\overline{QP}_{\alpha,\beta})$ and $Q_{\alpha,\beta} = Q + \alpha A^T A + \beta$. Recall that $\beta_{ij} = 0$, $\forall (i,j) \in J^2$.

Theorem 33.1. [4] *Let (SDP) be the following semidefinite program:*

$$
(SDP)\left\{\begin{array}{lll}
Min & f(X,x) = \sum_{i=1}^{N}\sum_{j=1}^{N} q_{ij}X_{ij} + \sum_{i=1}^{N} c_i x_i & \\
s.t. & (33.1)(33.2)(33.4) & \\
 & \sum_{r=1}^{m}(\sum_{i=1}^{N}(\sum_{j=1}^{N} a_{ri}a_{rj}X_{ij} - 2a_{ri}b_r x_i)) = -\sum_{r=1}^{m} b_r^2 & (33.18)\\
 & X_{ij} \leqslant u_j x_i \qquad (i,j) \in P & (33.19)\\
 & X_{ij} \leqslant u_i x_j \qquad (i,j) \in P & (33.20)\\
 & -X_{ij} \leqslant -u_j x_i - u_i x_j + u_i u_j \qquad (i,j) \in P & (33.21)\\
 & -X_{ij} \leqslant 0 \qquad (i,j) \in P & (33.22)\\
 & -X_{ii} \leqslant x_i \qquad i \in I & (33.23)\\
 & \begin{pmatrix} 1 & x \\ x^T & X \end{pmatrix} \succeq 0 & (33.24)\\
 & x \in \mathbb{R}^N \quad X \in \mathbf{S}^N & (33.25)
\end{array}\right.
$$

An optimal solution (α^*, β^*) of (CP) can be deduced from the optimal values of the dual variables of (SDP). The optimal coefficient α^* is the optimal value of the dual variable associated with constraint (33.18). The optimal coefficients β^*_{ij} are computed as $\beta^*_{ij} = \beta^{1*}_{ij} + \beta^{2*}_{ij} - \beta^{3*}_{ij} - \beta^{4*}_{ij}$, for $(i,j) \in P$, $i \neq j$, and $\beta^*_{ii} = \beta^{1*}_{ii} + \beta^{2*}_{ii} - \beta^{3*}_{ii} - \beta^{4*}_{ii} - \beta^{5*}_{ii}$, $i \in I$ where β^{1*}_{ij}, β^{2*}_{ij}, β^{3*}_{ij}, β^{4*}_{ij}, and β^{5*}_{ii}, are the optimal values of the dual variables associated with constraints (33.19), (33.20), (33.21), (33.22), and (33.23), respectively.

From Theorem 33.1, we design an exact solution algorithm for non-convex mixed-integer quadratic programs (QP) based on the `MIQCR` approach:

	Solution algorithm to (QP) based on `MIQCR`
1	Solve the semidefinite program (SDP)
2	Deduce α^* and β^* as in Theorem 33.1.
3	Solve the program (QP_{α^*,β^*}), by a MIQP solver. *(Its continuous relaxation $(\overline{QP}_{\alpha^*,\beta^*})$ is a convex program with an optimal value equal to the optimal value of (SDP))*

To illustrate our approach, we consider the following example.

Example 33.2. Let (QP_e) be an instance of (QP) with 2 integer and 2 continuous variables:

$$
(QP_e)\left\{\begin{array}{ll}
Min & f(x) = x^T \left(\begin{array}{cc|cc} -7 & 3 & -15 & -4 \\ 3 & -14 & -7 & -13 \\ \hline -15 & -7 & 8 & 7 \\ -4 & -13 & 7 & 12 \end{array}\right) x + \begin{pmatrix} 15 \\ 10 \\ -7 \\ -4 \end{pmatrix}^T x \\
s.t & 5x_1 + x_2 + 8x_3 + 4x_4 = 95 \\
 & 0 \leqslant x_i \leqslant 10 \qquad i \in \{1,\dots,4\} \\
 & x_1, x_2 \in \mathbb{N} \\
 & x_3, x_4 \in \mathbb{R}
\end{array}\right.
$$

Observe that the sub-matrix $\begin{pmatrix} 8 & 7 \\ 7 & 12 \end{pmatrix}$ is positive semidefinite. The optimal solution of (QP_e) is $x = (8, 10, 2.03, 7.19)$ and its value is -3434.27. We perturb the Q matrix as follows:

$$\begin{pmatrix} -7+\beta_{11}+25\alpha & 3+\beta_{12}+5\alpha & -15+\beta_{13}+40\alpha & -4+\beta_{14}+20\alpha \\ 3+\beta_{12}+5\alpha & -14+\beta_{22}+\alpha & -7+\beta_{23}+8\alpha & -13+\beta_{24}+4\alpha \\ -15+\beta_{13}+40\alpha & -7+\beta_{23}+8\alpha & 8+64\alpha & 7+32\alpha \\ -4+\beta_{14}+20\alpha & -13+\beta_{24}+4\alpha & 7+32\alpha & 12+16\alpha \end{pmatrix}$$

where the optimal value of the α parameter is 291.49, and the optimal values of the β parameter are: $\beta_{11} = \beta_{13} = \beta_{14} = 0$, $\beta_{12} = -8.73$, $\beta_{22} = 18.22$, $\beta_{23} = -0.005$, and $\beta_{24} = 4.16$. For the reformulated problem, the optimal value of the continuous relaxation equals -3434.45. The integrality gap is hence of 0.005%.

4. Computational results

Our experiments concern instances of (QP) that consists of minimizing a quadratic function subject to a linear equality constraint.

$$(MQP)\begin{cases} Min & x^TQx + c^Tx \\ s.t. & \sum_{i=1}^{N} a_ix_i = b \\ & 0 \leqslant x_i \leqslant u_i \quad i \in I \cup J \\ & x_i \in \mathbb{N} \qquad\quad i \in I \\ & x_i \in \mathbb{R} \qquad\quad i \in J \end{cases}$$

For this problem, we generate two classes of problems (MQP_1) and (MQP_2). For these two classes we randomly generate the coefficient u, Q, c, a and b in the same way, and we vary the number of integer variables. More precisely the coefficients of $Q\ \forall\,(i,j) \in P$ are integers uniformly distributed in the interval $[-100, 100]$ (for any $i < j$, a number ν is generated in $[-100, 100]$, and then $q_{ij} = q_{ji} = \nu$). To generate the coefficients of $Q\ \forall\,(i,j) \in J^2$, we generate a matrix $M \in \mathbf{M}_{N-n}$ of integers uniformly distributed in the interval $[-10, 10]$, and we compute $M' = M^TM$ then $q_{ij} = m'_{ij}$, $\forall (i,j) \in J^2$. The c coefficients are integers uniformly distributed in the interval $[-100, 100]$. The a_i coefficients are integers uniformly distributed in the interval $[1, 50]$, $b = 20 * \sum_{i=1}^{n} a_i$ and $u_i = 50$, $i \in I$. Note that in these instances the solution $x_i = 20$, for all i is feasible. For the class (MQP_1), we take 1/3 of integer variables, and 2/3 of continuous ones. For the class (MQP_2), we take 2/3 of integer variables, and 1/3 of continuous ones. For each problem and for each $N = 40$, 50, or 60, we generate 5 instances obtaining a total of 30 instances.

Our experiments are carried out on a PC with an Intel core 2 duo processor 2.8 GHz and 2048 MB of RAM using a Linux operating system. We use the modeling language ampl and the solver Cplex version 11 [**2**] for solving mixed integer quadratic convex programs, and the solver CSDP [**1**] for solving semidefinite programs.

Legends of the tables:

- *Name*: `Problem_i_r_nb`, where `i` is the number of integer variables in (QP), `r` is the number of real variables in (QP) and `nb` is the instance number.
- *Optimum*: best solution found within 1 hour of CPU time.

		MIQCR			
Name	Optimum	initial gap	CSDP time(s)	Cplex time (s)	nodes
MQP_1_13_27_1	-1441722.34	0	1240	5	0
MQP_1_13_27_2	-4001173.84	0	1252	3	0
MQP_1_13_27_3	-4072930.12	0	1587	5	0
MQP_1_13_27_4	-4303086.42	0	1294	4	0
MQP_1_13_27_5	-5792796.84	0	1246	4	6
average		**0**	**1323.8**	**4.2**	**1.2**
MQP_1_16_34_1	-6770822.01	0	6088	10	0
MQP_1_16_34_2	-4281691.17	1.07	4746	57	58
MQP_1_16_34_3	-12003888.62	0	4747	17	7
MQP_1_16_34_4	-14917101.36	0	4726	11	0
MQP_1_16_34_5	-8450308.13	0.15	4744	42	51
average		**0.24**	**5010.2**	**27.4**	**23.2**
MQP_1_20_40_1	-11331739.74	0	14756	55	9
MQP_1_20_40_2	-9541493.46	0.20	17959	142	52
MQP_1_20_40_3	-11243727.11	0.37	14530	46	14
MQP_1_20_40_4	-17871860.96	0	14682	38	0
MQP_1_20_40_5	-11194683.38	0.15	15835	61	16
average		**0.14**	**15552.4**	**68.4**	**16.2**

Table 1. Solution of MQP_1

		MIQCR			
Name	Optimum	Initial gap	CSDP time(s)	Cplex time (s)	nodes
MQP_2_27_13_1	-12764814.30	1.53	4698	331	632
MQP_2_27_13_2	-13063090.09	0.73	4703	91	99
MQP_2_27_13_3	-12210409.85	2.72	4720	234	390
MQP_2_27_13_4	-15060832.30	0.80	4691	131	268
MQP_2_27_13_5	-11550064.15	0.83	4714	91	91
average		**1.32**	**4705.2**	**175.6**	**296**
MQP_2_34_16_1	-15746064.71	0.80	19164	1626	3088
MQP_2_34_16_2	-21504640.72	0.83	19100	104	0
MQP_2_34_16_3	-16337803.32	2.36	19245	1541	1349
MQP_2_34_16_4	-17942418.38	2.05	19174	1637	3973
MQP_2_34_16_5	-21394688.60	0.49	20327	646	210
average		**1.31**	**19402**	**1180.8**	**1724**
MQP_2_40_20_1	-34437235.29	0.08	54539	838	38
MQP_2_40_20_2	-26342341.28	2.29	54701	-	829
MQP_2_40_20_3	-25124557.73	6.61	54656	-	635
MQP_2_40_20_4	-27395752.85	11.02	54826	-	873
MQP_2_40_20_5	-22573097.16	8.69	54817	-	1038
average		**5.73**	**54707.8**	**838 (1)**	**682.6**

Table 2. Solution of MQP_2

- *Initial gap*: $\left|\frac{opt - l}{opt}\right| * 100$ where l is the optimal value of the continuous relaxation at the root node.
- *CSDP time*: CPU time (in seconds) required by the SDP solver to find the optimal solution of the semidefinite relaxation of the initial problem.
- *Cplex time*: CPU time (in seconds) required by the branch-and-bound algorithm to solve the reformulated program. The time limit is fixed to 1 hour.
- *nodes*: number of nodes visited by the branch-and-bound algorithm.

Tables 1 and 2 present the results for the classes (MQP_1) and (MQP_2), respectively. For the class (MQP_1), that has less integer variables than real ones, every instance of size 40, 50 or 60 can be solved by the MIQP solver in less than 142 seconds. For these instances the average initial gap over the 15 instances is of 0.13% and hence the number of nodes visited during the Branch & Bound algorithm after reformulation is rather small, with an average of 13.5 nodes. Experiments on this class of problems give good first results. To experiment the impact of the ratio of integer variables on the `MIQCR` approach, we generate a second class of instances, (MQP_2), in the same way that for (MQP_1), but we invert the ratio of integer variables versus the real one. The results reveal a similar trend. However, for this class of problems, `MIQCR` leads to a reformulated problem with bounds obtained by continuous relaxation a bit worst in comparison to the class (MQP_1). Indeed, the average initial gap increases from 0.13% for (MQP_1) to 2.79% for (MQP_2). Consequently, the number of nodes visited in the Branch & Bound algorithm is then increased, with an average of 900.8 nodes. is only able to solve 8 instances over the 15 proposed. Finally, let us mention the preprocessing time associated with the optimal solution of the semidefinite programs. For the class (MQP_1) it takes 1323.8 seconds, 5010.2 seconds and 15552.4 seconds on average, for instances of size 40, 50, and 60, respectively. For the class (MQP_2), this time is larger, with an average of 4705.2 seconds, ac seconds and ac seconds, for instances of size 40, 50, and 60, respectively. Observe that, as every feasible dual of (SDP) provides α, and β that make convex the objective function of the reformulated problem [**4**], and because the SDP solvers often provide dual feasible solutions as they progress, the solution of semidefinite programs can be stopped after a fixed time. This possibility is interesting for large instances since SDP solvers generally find a good solution very quickly.

5. Conclusion

In this paper, we present a computational study of `MIQCR`, a convex reformulation of general mixed-integer programs. The approach has two phases: the first phase consists of building a convex reformulation of (QP), and in the second phase the reformulated problem is submitted to a MIQP solver. Computational experiments on two types of problems show that after the reformulation by the `MIQCR` approach, most of the instances proposed can be solved within 1 hour of CPU time. However, the time to compute the parameters of the reformulation is still weighty, and an important future direction for research consists of trying to decrease the SDP solution time.

References

1. B. Borchers. A C library for semidefinite programming. *Optimization Methods and Software*, 11(1):613–623, 1999.
2. ILOG. Ilog CPLEX 11.0 reference manual. *ILOG CPLEX Division, Gentilly*, 2008.
3. D.S. Johnson and M.R. Garey. Computers and intractability: A guide to the theory of NP-completness. *W.H. Freeman, San Francisco*, 1979.
4. A. Lambert, S. Elloumi, and A. Billionnet. Solution of general mixed-integer quadratic programs through convex reformulation. *Mathematical Programming*, accepted for publication.

5. N. Sawaya, F. Margot, A. Lodi, Lee J., C. Laird, I. Grossmann, G. Cornuéjols, A. Conn, L. Biegler, and P. Bonami. An algorithmic framework for convex mixed integer nonlinear programming. *Discrete Optimization*, 5:186–204, 2005.

Reduced RLT constraints for polynomial programming

Sonia Cafieri[1] Pierre Hansen[2] Lucas Létocart[3]
Leo Liberti[4] Frédéric Messine[5]

[1] ENAC, 7 av. E. Belin, 31055 Toulouse, France
sonia.cafieri@enac.fr

[2] GERAD & HEC Montréal, Canada
pierre.hansen@gerad.ca

[3] LIPN, Univ. de Paris Nord, France
lucas.letocart@lipn.univ-paris13.fr

[4] LIX, École Polytechnique, France
liberti@lix.polytechnique.fr

[5] ENSEEIHT-IRIT, Toulouse, France
frederic.messine@n7.fr

Abstract

An extension of the reduced Reformulation-Linearization Technique constraints from quadratic to general polynomial programming problems with linear equality constraints is presented and a strategy to improve the associated convex relaxation is proposed.
Keywords: polynomial, MINLP, sBB, convex relaxation, RLT.

1. Introduction

Reduced RLT constraints (rRLT) are a special class of Reformulation-Linearization Technique (RLT) constraints, that apply to nonconvex (both continuous and mixed-integer) quadratic programming problems subject to linear equality constraints [**2, 4, 3**]. rRLT are obtained by replacing some of the quadratic terms with suitable linear constraints. These turn out to be a subset of the RLT constraints for quadratic programming [**7**].

We present an extension of the rRLT theory to the case of general polynomial programs. Then, we show a strategy to choose the basis of a matrix involved in the rRLT constraints generation so as to tighten the bound of the associated convex relaxation. This allows to improve the performance of a spatial Branch-and-Bound algorithm applied to nonconvex NLP and MINLP problems where such convex relaxation is computed at each node.

2. Extending rRLT to polynomial programs

Let n be the number of variables, q the degree of the polynomials in the targeted problem and $\mathcal{N} = \{1, \dots, n\}$, $Q = \{2, \dots, q\}$. For each monomial $x_{j_1} \cdots x_{j_p}$, $p \in Q$, appearing in the problem, we define a finite sequence $J = (j_1, \dots, j_p)$ and consider defining constraints of the following form:

$$w_J = \prod_{\ell \le |J|} x_{j_\ell} \tag{34.1}$$

(for $|J| = 1$, i.e. $J = (j)$, we also define $w_J = x_j$). For all $p \in Q$, $J \in \mathcal{N}^p$ and any permutation π in the symmetric group S_p we have that $w_J = w_{\pi J}$ by commutativity. We therefore define an equivalence relation $\sim$ on $\mathcal{N}^p$ stating that for $J, K \in \mathcal{N}^p$, $J \sim K$ only if $\exists \pi \in S_p$ such that $J = \pi K$. We then consider the index tuple set $\bar{\mathcal{N}}^p = \mathcal{N}^p/\sim$ to quantify over when indexing variables w_J.

We multiply the original linear constraints $Ax = b$ by all monomials $\prod\limits_{\ell \le p-1} x_{j_\ell}$ and replace them by the corresponding added variables $w_{(J',j)}$, where $J' \in \bar{\mathcal{N}}^{p-1}$. This yields the following rRLTS:

$$\forall p \in Q, J' \in \bar{\mathcal{N}}^{p-1} \quad A\,\mathbf{w}_{J'} = b w_{J'}, \tag{34.2}$$

where $\mathbf{w}_{J'} = (w_{(J',1)}, \dots, w_{(J',n)})$. We then consider the companion system:

$$\forall p \in Q, J' \in \bar{\mathcal{N}}^{p-1} \quad A\,\mathbf{z}_{J'} = 0. \tag{34.3}$$

Since (34.3) is a linear homogeneous system, there is a matrix M such that the companion system is equivalent to $Mz = 0$, the columns of which are indexed by sequences in $\bar{\mathcal{N}}^p$. We let $B \subseteq \bar{\mathcal{N}}^p$ and $N \subseteq \bar{\mathcal{N}}^p$ be index sets for basic and nonbasic columns of M. We define the following sets:

$$\begin{aligned} C &= \{(x,w) \mid Ax = b \wedge \forall p \in Q, J \in \bar{\mathcal{N}}^p (w_J = \prod_{\ell \le |J|} x_{j_\ell})\} \\ R_N &= \{(x,w) \mid Ax = b \wedge \forall p \in Q, J' \in \bar{\mathcal{N}}^{p-1} (A\,\mathbf{w}_{J'} = b w_{J'}) \wedge \\ &\qquad \forall J \in N (w_J = \prod_{\ell \le |J|} x_{j_\ell})\}. \end{aligned}$$

Theorem 2.1. For each partition B, N into basic and nonbasic column indices for the companion system $Mz = 0$, we have $C = R_N$.

3. Tightening the convex relaxation

Replacing C with R_N for some nonbasis N effectively replaces some monomial terms with linear constraints, and therefore contributes to simplify the problem. A convex relaxation for the reformulated problem is readily obtained by applying monomial convexification methods in the literature [**5, 6, 1**]. We observe that for any given linear system there is in general more than one way to partition the variables in basics and nonbasics. Hence the set B can be chosen in such a way as to decrease the discrepancy between the feasible region and its convex relaxation. Given $f : X \subseteq \mathbb{R}^n \to \mathbb{R}$ and the sets $S = \{(x,w) \mid w = f(x)\}$ and

$\bar{S} = \{(x, w) \mid \underline{f}(x) \le w \le \bar{f}(x)\}$, where $\underline{f}(x)$, $\bar{f}(x)$ are respectively a convex lower and a concave upper bounding function for f (and hence $\bar{S}$ is a convex relaxation of S), the *convexity gap* between S and $\bar{S}$ can be defined as the volume $V(S)$ of the set $\bar{S}$. Explicit expressions of $V(S)$ can be derived for a quadratic term x_i^2, for a bilinear term $x_i x_j$ using the Cayley-Menger formula in 3 dimensions, and for a general monomial, exploiting associativity recursively to rewrite it as product of lower degree monomials and using the preceding results.

Let B, N be the basic/nonbasic sets of column indices of the companion system, which we can write as $M_B z_B + M_N z_N = 0$. The elements of B, N are sequences $J \in \mathscr{M}$. For $S \subseteq \mathscr{M}$ and $p \in Q$ we define $V^{S,p} = \sum_{\substack{J \in S \\ |J|=p}} V_J$ and $V^S = \sum_{p \in Q} V^{S,p}$. If, for all $p \in Q$, $V^{N,p} < V^{\beta,p}$ then the total convexity gap of R_N is smaller than that of C. Thus, we aim to find N such that $V^{N,p}$ is minimized, or equivalently, to find B such that $V^{B,p}$ is maximized for all $p \in Q$. This yields the multi-objective problem:

$$\left.\begin{array}{r} \forall p \in Q \quad \max V^{B,p} \\ M_B \text{ is a basis of (34.3)} \end{array}\right\} \tag{34.4}$$

It can be shown that (34.4) is equivalent to a single-objective problem: any solution B of (34.4) maximizing V^B also maximizes $V^{B,p}$ for all $p \in Q$. In this way, we have derived a technique to choose a good basis for the companion system so as to improve the chances of tightening the lower bound of the convex relaxation associated to rRLT.

Preliminary computational experiments carried out on a set of randomly generated instances of the convex Quadratic Knapsack Problem (cQKP) show that the proposed strategy is promising in improving performances of a spatial Branch-and-Bound algorithm.

References

1. S. Cafieri, J. Lee, and L. Liberti. On convex relaxations of quadrilinear terms. *Journal of Global Optimization*, to appear.
2. L. Liberti. Linearity embedded in nonconvex programs. *Journal of Global Optimization*, 33(2):157–196, 2005.
3. L. Liberti. Compact linearization of binary quadratic problems. *4OR*, 5(3):231–245, 2007.
4. L. Liberti and C.C. Pantelides. An exact reformulation algorithm for large nonconvex NLPs involving bilinear terms. *Journal of Global Optimization*, 36:161–189, 2006.
5. G.P. McCormick. Computability of global solutions to factorable nonconvex programs: Part I — Convex underestimating problems. *Mathematical Programming*, 10:146–175, 1976.
6. C.A. Meyer and C.A. Floudas. Trilinear monomials with mixed sign domains: Facets of the convex and concave envelopes. *Journal of Global Optimization*, 29:125–155, 2004.
7. H.D. Sherali and A. Alameddine. A new reformulation-linearization technique for bilinear programming problems. *Journal of Global Optimization*, 2:379–410, 1992.

Mixed-Integer Second-Order Cone Programming for Lower Hedging of American Contingent Claims in Incomplete Markets

Ahmet Camci **Mustafa C. Pınar**

Department of Industrial Engineering
Bilkent University
Ankara, 06800, Turkey

`{camci,mustafap}@bilkent.edu.tr`

Abstract

We describe a very challenging class of large Mixed-Integer Second-Order Cone Programming models which arise in computing the maximum price that a buyer is willing to disburse to acquire an American contingent claim in an incomplete financial market with no arbitrage opportunities. Taking the viewpoint of an investor who is willing to allow a controlled amount of risk by replacing the classical no-arbitrage assumption with a "no good-deal assumption" defined using an arbitrage-adjusted Sharpe ratio criterion we formulate the problem of computing the pricing and hedging of an American option in a financial market described by a multi-period, discrete-time, finite-state scenario tree as a large-scale mixed-integer conic optimization problem. We report computational results with off-the-shelf Mixed-Integer Conic Optimization software.
Keywords: american options, mixed-integer second-order cone optimization.

1. Introduction

The emergence of integer programming in mathematical finance has been so far limited mainly to portfolio optimization problems where buying and selling securities were subject to fixed transaction costs or when minimum lot restrictions were present. Cardinality constraints in the context of portfolio optimization may also lead to the use of discrete variables; see e.g., [**10, 18**].

In the present contribution, we report on a different area of mathematical finance where finite dimensional optimization models with discrete-valued variables is of interest: pricing of stochastic cash flows. The pricing problem of stochastic cash flows is complicated by the fact that most financial markets are incomplete, i.e., not all future uncertain cash flows can be replicated exactly using the existing instruments. This observation leads to a wealth of literature on pricing and hedging in incomplete markets; see e.g., [**3, 5, 8, 17**]. When markets are incomplete state prices and claim prices are not unique. Since markets are almost never complete due to market imperfections as discussed in Carr *et al.* [**5**], and characterizing all possible future states of economy is impossible, the common practice is to find the cheapest portfolio dominating a stochastic future cash flow and the most expensive portfolio dominated by it, and use these respective values as bounds on the price of the stochastic cash flow. These bounds are referred to as super-replication and sub-replication bounds or no-arbitrage (or equilibrium) bounds. They are also known as upper hedging and the lower hedging prices. In the absence of arbitrage, the lower hedging price is the value of the most precious self-financing portfolio strategy composed of market instruments whose pay-off is dominated by the contingent claim pay-off at expiration. The lower hedging price can also be interpreted as the largest amount the contingent claim buyer can borrow (in the form of cash or by short-selling stocks) to acquire the claim while paying off his/her debt in a self-financed manner using the contingent claim pay-off at expiration [**9**]. Hence, we refer to this price as the buyer's price as well as the lower hedging price. For European contingent claims with no early exercise and termination possibility, the upper and lower hedging prices are expressed as supremum and infimum, respectively, of the expectation of the discounted contingent claim pay-off (at expiration) over all probability measures that make the underlying stock price a martingale. We direct the reader to the book by Föllmer and Schied [**12**] for an in-depth treatment of pricing contingent claims in discrete time, and to Chalasani and Jha [**9**] for American contingent claims (ACC), which can be exercised at any time until expiration. The upper hedging price for ACC is the supremum of the expectation of the discounted contingent claim pay-off (at some time between now and expiration) over all stopping times and all probability measures that make the underlying stock-price process a martingale. While the upper hedging price for ACC can be cast as a linear programming problem in discrete time [**9, 16**], the lower hedging price is harder to compute since it requires the solution of a mixed-integer programming problem [**16**]. Recently, Camcı and Pınar [**4**] showed that the lower hedging price can also be computed by solving the LP relaxation of the mized-integer model, i.e., the LP relaxation is exact. On the other hand, as indicated in [**12**], the computed upper and lower hedging prices may be far apart, and useless in practice.

In the present work, we are interested in pricing ACCs using a risk criterion introduced in [**11**] and further developed in [**6, 8**]. Our work is based on good-deals defined as investments with high arbitrage adjusted Sharpe ratio [**8**]. Our motivation is to derive a tighter lower hedging price for an ACC in an incomplete market. In a multi-period, discrete time, discrete state space framework we define the stock price process as a non-recombinant tree and formulate a mixed-integer second-order cone programming problem for computing the lower hedging price under the Sharpe ratio risk criterion. After a brief introduction to the financial market setting, we describe our pricing measure and give our mixed-integer second-order cone optimization formulation. We develop an application using S & P 500 index options data, and report our preliminary computational experience on these very difficult and large problem instances whose even convex relaxation may prove tough to handle. These instances clearly pose a challenge to the numerical optimization community. We obtain the optimal solution of the instances using duality theory and guesswork.

2. The Stochastic Scenario Tree

We approximate the behavior of the stock market by assuming that security prices and other payments are discrete random variables supported on a finite probability space $(\Omega, \mathcal{F}, P)$ whose atoms are sequences of real-valued vectors (asset values) over the discrete time periods $t = 0, 1, \ldots, T$ as in [**13**]. We further assume the market evolves as a discrete, non-recombinant scenario tree (hence, suitable for incomplete markets) in which the partition of probability atoms $\omega \in \Omega$ generated by matching path histories up to time t corresponds one-to-one with nodes $n \in \mathcal{N}_t$ at level t in the tree. The set $\mathcal{N}_0$ consists of the root node $n = 0$, and the leaf nodes $n \in \mathcal{N}_T$ correspond one-to-one with the probability atoms $\omega \in \Omega$. The σ-algebras are such that, $\mathcal{F}_0 = \{\emptyset, \Omega\}$, $\mathcal{F}_t \subset \mathcal{F}_{t+1}$ for all $0 \leqslant t \leqslant T-1$ and $\mathcal{F}_T = \mathcal{F}$. A stochastic process is said to be $(\mathcal{F}_t)_{t=0}^T$-adapted if for each $t = 0, \ldots, T$, the outcome of the process only depends on which element of $\mathcal{F}_t$ has been realized at stage t. Similarly, a decision process is said to be $(\mathcal{F}_t)_{t=0}^T$-adapted if for each $t = 0, \ldots, T$, the decision depends on which element of $\mathcal{F}_t$ has been realized at stage t. In the scenario tree, every node $n \in \mathcal{N}_t$ for $t = 1, \ldots, T$ has a unique parent denoted $\pi(n) \in \mathcal{N}_{t-1}$, and every node $n \in \mathcal{N}_t$, $t = 0, 1, \ldots, T-1$ has a non-empty set of child nodes $\mathcal{C}(n) \subset \mathcal{N}_{t+1}$. We denote the set of all nodes in the tree by $\mathcal{N}$. $\mathcal{A}(n)$ denotes the ascendant nodes or path history of node n including itself. The probability distribution P is obtained by attaching positive weights p_n to each leaf node $n \in \mathcal{N}_T$ so that $\sum_{n \in \mathcal{N}_T} p_n = 1$. For each non-terminal (intermediate level) node in the tree we have, recursively,

$$p_n = \sum_{m \in \mathcal{C}(n)} p_m, \ \ \forall\, n \in \mathcal{N}_t, \ t = T-1, \ldots, 0.$$

Hence, each intermediate node has a probability mass equal to the combined mass of the paths passing through it.

A random variable X is a real valued function defined on Ω. It can be *lifted* to the nodes of a partition $\mathcal{N}_t$ of Ω if each level set $\{X^{-1}(a) : a \in \mathbb{R}\}$ is either the empty set or is a finite union of elements of the partition. In other words, X can be lifted to $\mathcal{N}_t$ if it can be assigned a value on each node of $\mathcal{N}_t$ that is consistent with its definition on Ω, [**13**]. This kind of random variable is said to be measurable with respect to the information contained in the nodes of $\mathcal{N}_t$. A stochastic process $\{X_t\}$ is a time-indexed collection of random variables such that each X_t is measurable with respect $\mathcal{N}_t$. The expected value of X_t is uniquely defined by the sum $\mathbb{E}^P[X_t] := \sum_{n \in \mathcal{N}_t} p_n X_n$.

The market consists of $J+1$ traded securities indexed by $j = 0, 1, \ldots, J$ with prices at node n given by the vector $S_n = (S_n^0, S_n^1, \ldots, S_n^J)$. We assume as in [**16, 7**] that the security indexed by 0 has strictly positive prices at each node of the scenario tree.

The amount of security j held by the investor in state (node) $n \in \mathcal{N}_t$ is denoted θ_n^j. Therefore, to each state $n \in \mathcal{N}_t$ is associated a vector $\theta_n \in \mathbb{R}^{J+1}$. The value of the portfolio at state n is $S_n \cdot \theta_n = \sum_{j=0}^J S_n^j \theta_n^j$.

In our finite probability space setting an American contingent claim H is a stochastic process measurable with respect to N_t, and hence, generates payoff opportunities H_n, $(n \geqslant 0)$ to its holder depending on the states n of the market.

3. Arbitrage-Adjusted Sharpe Ratio Good-Deals

In our context *a good-deal opportunity* is defined as follows. Let the random variable X represent the uncertain cash flow of an investment. The random variable X is split into two components:

$$X = X_{sh} + X_{arb},$$

where X_{arb} is a non-negative random variable measuring the arbitrage (zero risk) component of the cash flow (we will illustrate this concept in the sequel), and X_{sh} measures the Sharpe ratio component of X. Then the arbitrage-adjusted Sharpe ratio of X is defined as

$$\frac{\mathbb{E}[X]}{\sqrt{\sigma(X)}}$$

where $\sigma(X)$ denotes the variance of the random variable X. For $n \in \mathcal{N}_T$ let $S_n \cdot \theta_n = x_n + v_n$ where v_n is non-negative. Assume that there exist a set of vectors $\theta_0, \theta_1, \ldots, \theta_{|\mathcal{N}|}$ such that

$$S_0 \cdot \theta_0 = 0$$

$$S_n \cdot (\theta_n - \theta_{\pi(n)}) = 0, \ \forall\, n \in \mathcal{N}_t, t \geqslant 1$$

and

$$\sum_{n \in \mathcal{N}_T} p_n x_n - \lambda \sqrt{\sum_{n \in \mathcal{N}_T} p_n (x_n - \sum_{n \in \mathcal{N}_T} p_n x_n)^2} > 0$$

for $\lambda > 0$. This sequence of portfolio holdings is said to yield a "Sharpe ratio good-deal opportunity" at level λ. This formulation is similar to the Sharpe ratio criterion treated in [**8, 11**]. Here, the parameter λ can be interpreted as a loss aversion parameter of the individual investor because as λ gets larger the investor is closer to seeking an arbitrage.

4. The Formulation

Now, let us assume that an American contingent claim H is available in our financial market setting. The potential buyer is interested in borrowing cash to acquire the claim, and with the remaining cash to form a portfolio of traded instruments. She/he will modify this portfolio later using proceeds from the claim (if exercised) or through self-financing transactions. At the expiry date of the option the final positions that the buyer carries should satisfy the Sharpe ratio risk constraint, i.e., the ratio of the average position to the standard deviation of the position should remain bounded by λ. The exercise of the American contingent claim is controlled by means of binary variables e_n, $n \in N$. Hence, under the assumption of no good-deal opportunities for the stock price process, the price of the ACC that provides no good-deals to the buyer must be greater than or equal to the optimal value of the following optimization problem.

$$
\begin{aligned}
\max \quad & V \\
\text{s.t.} \quad & S_0 \cdot \theta_0 = H_0 e_0 - V \\
& S_n \cdot (\theta_n - \theta_{\pi(n)}) = H_n e_n,\ \forall\, n \in \mathcal{N}_t, 1 \leqslant t \leqslant T \\
& S_n \cdot \theta_n - x_n - v_n = 0,\ \forall\, n \in \mathcal{N}_T \\
& \sum_{n \in \mathcal{N}_T} p_n x_n - \lambda \sqrt{\sum_{n \in \mathcal{N}_T} p_n (x_n - \sum_{n \in \mathcal{N}_T} p_n x_n)^2} \geqslant 0 \\
& \sum_{m \in \mathcal{A}(n)} e_m \leqslant 1,\ \forall\, n \in \mathcal{N}_T \\
& v_n \geqslant 0,\ \forall\, n \in \mathcal{N}_T \\
& e_n \in \{0, 1\},\ \forall\, n \in \mathcal{N}.
\end{aligned}
$$

This problem is a mixed integer second-order cone programming (MISOCP) problem. The first constraint ties the initial wealth (borrowed) to the value of an initial portfolio. The second set of constraints represent the portfolio transactions at the nodes of subsequent periods. The third constraint is the Sharpe ratio good-deal constraint while the fourth set of constraints limits exercise to a single node over each sample path. For computational purposes we shall work on a slightly different model described next.

5. Calibrated Option Bounds

In the setting of [**14**] adopted for our numerical tests, liquid options traded in the market are used for hedging purposes in addition to securities. These liquid options give the investor the possibility of forming buy-and-hold strategies in the hedging portfolio sequence. In other words, every liquid option can be bought or shorted by the investor at time zero with the purpose of hedging a contingent claim, and no intermediary trading is available for these options. Assuming there are K such liquid options, we denote them by G^k, $k = 1, \ldots, K$. Bid and ask prices observed in the market at time 0 for option k are denoted by C_b^k and C_a^k, respectively, with the latter greater than or equal to the former. G_n^k is the payoff of option k at node n of the scenario three and G_n is the vector of option payoffs at node n. At this point we will assume that $S_n^0 = 1$, $\forall n \in \mathcal{N}$, to use normal stock prices instead of discounted prices. This assumption is consistent with our numerical experiments since we use a zero interest rate. The non-negative K-vectors ξ_+ and ξ_- denote, respectively, the long and short initial buy-and-hold positions in the liquid options. Under these assumptions the buyer's problem we shall be solving is modified as (referred to as (P))

$$\begin{array}{rr}
\max & -S_0 \cdot \theta_0 - C_a \cdot \xi_+ + C_b \cdot \xi_- + H_0 e_0 \\
\text{s.t.} & S_n \cdot (\theta_n - \theta_{\pi(n)}) = G_n \cdot (\xi_+ - \xi_-) + e_n H_n, \ \forall\, n \in \mathcal{N}_t, t \geqslant 1 \\
& \sum_{n \in \mathcal{N}_T} p_n x_n - \lambda \sqrt{\sum_{n \in \mathcal{N}_T} p_n (x_n - \sum_{n \in \mathcal{N}_T} p_n x_n)^2} \geqslant 0 \\
& v_n + x_n = S_n \cdot \theta_n, \ \forall\, n \in \mathcal{N}_T, \\
& \sum_{m \in \mathcal{A}(n)} e_m \leqslant 1, \ \forall\, n \in \mathcal{N}_T \\
& e_n \in \{0,1\}, \ \forall\, n \in \mathcal{N} \\
& \xi_+, \xi_- \geqslant 0, \\
& v_n \geqslant 0, \ \forall\, n \in \mathcal{N}_T.
\end{array}$$

6. Numerical Results and Solution via Duality

We use 48 European options written on the $S\&P500$ index. The option data were available in the market on September 10, 2002. The first 21 are call options and the remaining ones are put options. Strikes and maturities as well as actual bid and ask prices (columns C_b and C_a) of these options are given in Table 1. We compute "calibrated" pricing bounds for each option treating that option as an American option. This means that the buyer or writer of the option can include buy-and-hold positions in the 47 remaining European options into his/her hedge portfolio sequence.

We use $S = (1, S^1)$ as the traded securities. Having $S^0 = 1$ for all dates means that interest rate is zero. We assume that the price of the $S\&P$ 500 index (i.e. S^1) follows a geometric Brownian motion. Under this assumption, we generate a scenario tree by the Gauss-Hermite process which was discussed in [**14, 15**] in detail. We use a branching structure of (50,10,10). It means that the tree divides into 50 nodes at the second period. Then, each node branches into ten nodes in the second period hence there are additional 500 nodes in the third period. Then, again each node of the third period is divided into 10 and there are 5000 leaf nodes of the tree. We assume that investors can trade at days 0, 17, 37 and 100, and form instances of problem P with four periods, which already yields huge MISOCP instances with up to 25,553 constraints and 31,749 variables of which 5,551 are binary. Let X denote the strike price. We have now $H_n = (S_n^1 - X)_+$ for call options and $H_n = (X - S_n^1)_+$ for put options for all $n \in N$.

We used the conic interior point optimizer MOSEK version 5.0.0.127 [**2**] under default settings through GAMS Version 23.2 [**1**] to determine buyer's prices for each option for $\lambda = 5.7$. We report computational results in Table 1. It proved impossible to get most MINLP solvers to work on the present instances using the primal formulation P. MOSEK was the only code to solve the convex relaxations with some success, but it stopped immediately with an error message when the models were input as mixed-integer conic models. In Table 1 below, the continuous (relaxed) models marked with a B for "binary" solution, and solver and model status both equal to 1 were solved successfully with MOSEK, and yielded an integer e component hence the optimal solution to the original model for claims with the earliest maturity date (day 17). However, for the remaining models either the

Option No.	Type	Strike	Maturity	C_b	C_a	Buyer's Price	Solver & Model Status	Solution
1	Call	890	17	31.5	33.5	31.22	1, 1	B
2	Call	900	17	24.4	26.4	24.67	1, 1	B
3	Call	905	17	21.2	23.2	21.80	1, 1	B
4	Call	910	17	18.5	20.1	18.92	1, 1	B
5	Call	915	17	15.8	17.4	16.09	1, 1	B
6	Call	925	17	11.2	12.6	-	4, 6	-
7	Call	935	17	7.6	8.6	7.68	1, 1	B
8	Call	950	17	3.8	4.6	3.39	1, 1	B
9	Call	955	17	3	3.7	2.99	1, 1	B
10	Call	975	17	0.95	1.45	0.65	1, 1	B
11	Call	980	17	0.65	1.15	0.66	1, 1	B
12	Call	900	37	42.3	44.3	40.58	1, 1	F
13	Call	925	37	28.2	29.6	-	4, 6	-
14	Call	950	37	17.5	19	-	4, 6	-
15	Call	875	100	77.1	79.1	75.49	1, 1	F
16	Call	900	100	61.6	63.6	59.87	1, 1	F
17	Call	950	100	35.8	37.8	-	4, 6	-
18	Call	975	100	26	28	-	4, 6	-
19	Call	995	100	19.9	21.5	-	4, 6	-
20	Call	1025	100	12.6	14.2	-	4, 6	-
21	Call	1100	100	3.4	3.8	-	4, 6	-
22	Put	750	17	0.4	0.6	-	4, 6	-
23	Put	790	17	1	1.3	1.01	1, 1	B
24	Put	800	17	1.3	1.65	1.21	1, 1	B
25	Put	825	17	2.5	2.85	2.05	1, 1	B
26	Put	830	17	2.6	3.1	2.74	1, 1	B
27	Put	840	17	3.4	3.8	3.41	4, 6	-
28	Put	850	17	3.9	4.7	4.41	1, 1	B
29	Put	860	17	5.5	5.8	5.39	4, 6	-
30	Put	875	17	7.2	7.8	7.60	4, 6	-
31	Put	885	17	9.4	10.4	10.40	1, 1	B
32	Put	750	37	5.5	5.9	-	2, 6	-
33	Put	775	37	6.9	7.7	7.60	1, 1	F
34	Put	800	37	9.3	10	9.89	1, 1	F
35	Put	850	37	16.7	18.3	-	4, 6	-
36	Put	875	37	23	24.3	22.39	1, 1	F
37	Put	900	37	31	33	32.70	1, 1	F
38	Put	925	37	41.8	43.8	43.61	1, 1	F
39	Put	975	37	73	75	72.57	1, 1	F
40	Put	995	37	88.9	90.9	-	4, 6	-
41	Put	650	100	5.7	6.7	2.73	1, 1	F
42	Put	700	100	9.2	10.2	10.10	1, 1	F
43	Put	750	100	14.7	15.8	-	4, 6	-
44	Put	775	100	17.6	19.2	-	4, 6	-
45	Put	800	100	21.7	23.7	-	4, 6	-
46	Put	850	100	33.3	35.3	32.93	1, 1	F
47	Put	875	100	40.9	42.9	-	4, 6	-
48	Put	900	100	50.3	52.3	51.99	1, 1	F

Table 1. Numerical Results for the Continuous Relaxation of Sharpe Ratio Lower Hedging Primal Problem (P) with $\lambda = 5.7$

Solver Status Codes: 1= Normal Completion, 2=Iteration Interrupt, 4=Terminated by Solver

Model Status Codes: 1= Optimal, 6=Intermediate Infeasible, F=Fractional, B=Binary

solver successfully returned an optimal fractional solution (hence, an upper bound on the buyer's price), or stopped with an error message (reported in the table). It is conjectured that the solvers run into numerical problems due to the very small entries for the node probabilities generated from the Gauss-Hermite process. For models with five trading dates that are even larger, it is expected that the problems will grow even more challenging.

Having failed at obtaining solutions for a large number of instances using the primal formulation P of the problem at hand we turned at a bounding approach using duality.

Let E be the set $\{e_n | \sum_{m\in\mathcal{A}(n)} e_m \leqslant 1, \; \forall\, n \in \mathcal{N}_T \text{ and } e_n \in \{0,1\}\,, \; \forall\, n \in \mathcal{N}\}$. Based on our earlier work [**6**], for fixed $e \in E$ the inner maximization over the remaining variables can be seen to be equivalent via duality to the following problem DL over the variables q_n (under a strict feasibility assumption to guarantee zero-duality gap):

$$\min \sum_{n\in N} e_n q_n H_n \tag{35.1}$$

subject to (35.2)-(35.3)-(35.4)-(35.5)-(35.6) as defined below:

$$q_m S_m = \sum_{n\in\mathcal{C}(m)} q_n S_n, \; \forall\, m \in \mathcal{N}_t, 0 \leqslant t \leqslant T-1 \tag{35.2}$$

$$q_0 = 1, \tag{35.3}$$

$$q_n \geqslant 0, \; \forall n \in \mathcal{N}, \tag{35.4}$$

$$\sqrt{\sum_{n\in\mathcal{N}_T} p_n (\frac{q_n}{p_n} - 1)^2} \leqslant \lambda, \tag{35.5}$$

$$C_b \leqslant \sum_{t=1}^{T} \sum_{n\in N_t} q_n G_n \leqslant C_a. \tag{35.6}$$

Solving the above problem with e fixed gives a lower bound to the optimal value $OPT(P)$. Inspired by the well-known folklore result that it is sub-optimal to exercise American options before maturity (see e.g. [**12**]) we fix $e_n = 1$ for all $n \in N_T$ (and thereby force all other e's to zero) and solve the above problem DL successfully using GAMS/MOSEK (solver returns an optimal solution with default parameter settings). The results are reported in Table 2 under column "LB" for lower bound.

To assess the quality of the lower bound, we look for a relaxation of the problem that would be easy to solve, and aim for the dual of the continuous relaxation of P to give an upper bound. This dual problem, referred to as DU, is the following problem (after some algebra and elimination of variables, the details of which are left to the reader):

$$\min \sum_{n\in N_T} q_n H_n \tag{35.7}$$

subject to (35.2)-(35.3)-(35.4)-(35.5)-(35.6)-(35.8) where (35.8) is as defined below:

$$q_n H_n \leqslant \sum_{m\in C(n)} q_m H_m \; \forall\, n \in N_t, 0 \leqslant t \leqslant T-1. \tag{35.8}$$

Solving problem DU successfully (solver returned optimal solution with default settings) in GAMS/MOSEK we find that the lower and upper bounds collapse for almost all the difficult instances! The results are reported in Table 2. The small

Option No.	Type	Strike	Maturity	C_b	C_a	UB	LB
6	Call	925	17	11.2	12.6	10.57	10.57
12	Call	900	37	42.3	44.3	40.58	40.58
13	Call	925	37	28.2	29.6	26.86	26.86
14	Call	950	37	17.5	19	14.32	14.32
15	Call	875	100	77.1	79.1	75.48	75.48
16	Call	900	100	61.6	63.6	59.88	59.88
17	Call	950	100	35.8	37.8	33.81	33.80
18	Call	975	100	26	28	24.99	24.99
19	Call	995	100	19.9	21.5	18.90	18.90
20	Call	1025	100	12.6	14.2	10.06	10.03
21	Call	1100	100	3.4	3.8	0.389	0.384
22	Put	750	17	0.4	0.6	0.545	0.545
27	Put	840	17	3.4	3.8	3.44	3.44
29	Put	860	17	5.5	5.8	5.41	5.41
30	Put	875	17	7.2	7.8	7.65	7.65
32	Put	750	37	5.5	5.9	3.80	3.80
33	Put	775	37	6.9	7.7	7.60	7.60
34	Put	800	37	9.3	10	9.89	9.89
35	Put	850	37	16.7	18.3	15.84	15.84
36	Put	875	37	23	24.3	22.43	22.43
37	Put	900	37	31	33	32.72	32.72
38	Put	925	37	41.8	43.8	43.62	43.62
39	Put	975	37	73	75	72.70	72.70
40	Put	995	37	88.9	90.9	87.72	87.72
41	Put	650	100	5.7	6.7	2.72	2.72
42	Put	700	100	9.2	10.2	10.02	10.02
43	Put	750	100	14.7	15.8	15.02	15.02
44	Put	775	100	17.6	19.2	17.84	17.84
45	Put	800	100	21.7	23.7	21.04	21.03
46	Put	850	100	33.3	35.3	32.97	32.97
47	Put	875	100	40.9	42.9	42.52	42.52
48	Put	900	100	50.3	52.3	52.02	52.02

Table 2. Numerical Results on the Most Difficult Instances of Table 1 for Sharpe Ratio Lower Hedging Bounding Problems DL and DU with $\lambda = 5.7$

disprencies (always after the decimal point) between upper bounds of Table 1 and those of Table 2 are attributed to the numerical difficulties experienced by the conic solver when faced with the primal problem that is deemed less stable compared to the dual models.

The above result indicates that it may be always optimal to exercise the index options treated in this paper under Sharpe-ratio restrictions at the final period. A result in this direction can be rigorously proven, but will be published elsewhere.

Acknowledgments

Research supported by TUBITAK Grant 107K250.

References

1. *GAMS: A User's Guide.* The Scientific Press, San Fransisco, California, 1992.

2. Mosek ApS. Mosek solver manual. Technical report, Mosek ApS c/o Symbion Science Park, Fruebjergvej 3, Box 15, 2100 Copenhagen, Denmark, 2009.
3. A. E. Bernardo and O. Ledoit. Gain, loss and asset pricing. *J. Political Economy*, 108:144–172, 2000.
4. A. Camcı and M. Ç. Pınar. Pricing american contingent claims by stochastic linear programming. *Optimization*, 58:627–640, 2009.
5. P. Carr, H. Geman, and D.B. Madan. The pricing and hedging in incomplete markets. *J. Financial Economics*, 62:131–167, 2001.
6. M. Ç. Pınar. Sharpe-ratio pricing and hedging of contingent claims in incomplete markets by convex programming. *Automatica*, 44:2063–2073, 2008.
7. M. Ç. Pınar, A. Altay-Salih, and A. Camcı. Expected gain-loss pricing and hedging of contingent claims in incomplete markets by linear programming. *European J. Oper. Res.*, 201:770–785, 2010.
8. A. Cerny. Generalized sharpe ratios and pricing in incomplete markets. *European Finance Review*, 7:191–233, 2003.
9. P. Chalasani and S. Jha. Randomized stopping times and american option pricing with transaction costs. *Mathematical Finance*, 11:33–77, 2001.
10. T.J. Chang, N. Meade, J.E. Beasley, and Y.M. Sharaiha. Heuristics for cardinality constrained portfolio optimization. *Computers and Oper. Res.*, 27:1271–1302, 2000.
11. J. H. Cochrane and J. Saa-Requejo. Beyond arbitrage: Good-deal asset price bounds in incomplete markets. *J. Political Economy*, 108:79–119, 2000.
12. H. Föllmer and A. Schied. *Stochastic Finance: An Introduction in Discrete Time*. W. De Gruyter, 2004.
13. A. J. King. Duality and martingales: A stochastic programming perspective on contingent claims. *Mathematical Programming Series B*, 91:543–562, 2002.
14. A.J. King, M. Koivu, and T. Pennanen. Calibrated option bounds. *Int. J. of Theor. and Appl. Finance*, 8:141–159, 2005.
15. E. Omberg. Efficient discrete time jump process models in option pricing. *J. of Financial and Quantitative Analysis*, 23:161–164, 1988.
16. T. Pennanen and A. King. Arbitrage pricing of american contingent claims in incomplete markets - a convex optimization approach. Technical report, Helsinki School of Economics, 2006.
17. J. Staum. Fundamental theorems of asset pricing for good deal bounds. *Mathematical Finance*, 14:141–161, 2004.
18. S.A. Zenios, editor. *Financial Optimization*. Cambridge University Press, 1993.

Formulation symmetries in circle packing

Alberto Costa[1] **Pierre Hansen**[1,2] **Leo Liberti**[1]

[1] LIX, École Polytechnique
91128 Palaiseau, France
{costa,liberti}@lix.polytechnique.fr

[2] GERAD & HEC,
3000 ch. Côte-S.te-Catherine, H3T 2A7 Montreal, Canada
pierre.hansen@gerad.ca

Abstract

The performance of Branch-and-Bound algorithms is severely impaired by the presence of symmetric optima in a given problem. We propose here a method to automatically find MINLP formulation symmetries. We show an application of our method to the "circle packing in a square" problem, in order to get a reformulation that should cut away symmetric optima.

Keywords: MINLP, spatial Branch-and-Bound, Global Optimization, group, reformulation.

1. Introduction

It is well known that problems involving a high degree of symmetry are particularly difficult to solve with Branch-and-Bound (BB) techniques. Intuitively, since optimal solutions are to be found at leaf nodes of the BB tree, the presence of many optima causes fewer prunings, longer branches, and hence a higher number of nodes to explore. One possibility for breaking symmetries, proposed in [**2**], is to reformulate the problem by adjoining *symmetry-breaking constraints* (SBC) to the original formulation, yielding a reformulation of the *narrowing* type [**1**]. The main theoretical contribution of this paper is the determination of the group structure of circle packing problem.

2. Automatic symmetry detection

In this section we discuss a method for computing Mathematical Program (MP) symmetries automatically; conceptually, it is the same as in [**2**] but the formal presentation is different. We consider a Mixed-Integer Nonlinear Program (MINLP)

P:

$$\min\{f(x) \mid g(x) \leq 0 \wedge x \in \mathcal{X}\}, \tag{36.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$, $x \in \mathbb{R}^n$, and $\mathcal{X} \subseteq \mathbb{R}^n$ is a set which might include variable ranges $x^L \leq x \leq x^U$ as well as integrality constraints on a subset of variables $\{x_i \mid i \in Z\}$ for some $Z \subseteq \{1, \ldots, n\}$. Let $\mathcal{G}(P)$ be the set of global optima of P and $\mathcal{F}(P)$ be its feasible region. The group $G^*_P = \text{stab}(\mathcal{G}(P), S_n)$ is called the *solution group* of P (where S_n is the symmetric group of order n). The solution group is the largest subgroup of S_n which maps every global optimum into another global optimum. Since G^*_P depends on $\mathcal{G}(P)$ it cannot, in general, be found before the solution process. We therefore try to find subgroups of G^*_P. In particular, we consider the subgroup of G^*_P consisting of all variable permutations which "fix the formulation" of P. For $\pi \in S_n$ and $\sigma \in S_m$ we define $\sigma P \pi$ to be the following MINLP:

$$\min\{f(\pi x) \mid \sigma g(\pi x) \leq 0 \wedge \pi x \in \mathcal{X}\}, \tag{36.2}$$

where σ acts on $g = (g_1, \ldots, g_m)$ by $\sigma g = (g_{\sigma^{-1}(1)}, \ldots, g_{\sigma^{-1}(m)})$. Consider the group $\bar{G}_P = \{\pi \in S_n \mid \exists \sigma \in S_m \ (\sigma P \pi) = P\}$. Whenever P is a Mixed-Integer Linear Program (MILP), $\bar{G}_P$ is called the *LP relaxation group* [**3**]. For general MINLPs, determining whether $\forall x \in \text{dom}(f)\ f(\pi x) = f(x)$ and $\forall x \in \text{dom}(g)\ \sigma g(\pi x) = g(x)$ is an undecidable problem.

We therefore introduce the following restriction: f, g_i ($i \leq m$) must be strings of the formal language $\mathscr{L}$ on the alphabet $\mathscr{A}$ given by the operators in $\{+, -, \times, \div, \uparrow, \log, \exp, (,)\}$ (where $a \uparrow b = a^b$), the variable symbols in $\{x_1, \ldots, x_n\}$ and the constant symbols in $\mathbb{R}$. This restriction allow us to define the *formulation group* $G_P = \{\pi \in S_n \mid \exists \sigma \in S_m \ (\sigma P \pi \cong P)\}$ of P (where the symbol $\cong$ indicates that the formulations are equal respetct to this restriction). It is easy to show that $G_P \leqslant \bar{G}_P \leqslant G^*_P$. For MILPs, $G_P = \bar{G}_P$ [**2**].

Once G_P is known, we aim to find a reformulation Q of P which ensures that at least one symmetric optimum of P is in $\mathcal{G}(Q)$. Such reformulations are known as *narrowings* [**1**]. A set of constraints $h(x) \leq 0$ are SBCs with respect to $\pi \in G_P$ if there is $y \in \mathcal{G}(P)$ such that $h(\pi y) \leq 0$. Adjoining SBCs to P yields a narrowing Q of P [**2**].

3. Circle Packing in a square

We consider the following problem.

> Circle Packing in a Square (CPS). Given $N \in \mathbb{N}$ and $L \in \mathbb{Q}_+$, can N non-overlapping circles of unit radius be arranged in a square of side $2L$?

We formulate the CPS as the following nonconvex NLP:

$$\max\{\alpha \mid \forall i < j \leq N \ \|x_i - x_j\|^2 \geq 4\alpha \wedge x \in [1 - L, L - 1]^{2N}\} \tag{36.3}$$

For any given $N, L > 1$, if a global optimum (x^*, α^*) of (36.3) has $\alpha^* \geq 1$ then the CPS instance is a YES one. Using the theory above, we are able to prove this

Theorem
The formulation group of the CPS is isomorphic to $S_2 \times S_N$.

This result allow us to add some SBCs to the original formulation. Some preliminary tests show that solving the reformulated circle packing problem is more easy than solve the original one, in term of time and nodes generated by the Spatial Branch and Bound.

References

1. Liberti, L., *Reformulations in mathematical programming: Definitions and systematics*, RAIRO-RO **43** (2009), pp. 55–86.
2. Liberti, L., *Reformulations in mathematical programming: Symmetry*, Mathematical Programming (in revision).
3. Margot, F., *Symmetry in integer linear programming*, in: M. Jünger, T. Liebling, D. Naddef, G. Nemhauser, W. Pulleyblank, G. Reinelt, G. Rinaldi and L. Wolsey, editors, *50 Years of Integer Programming*, Springer, Berlin, 2010 pp. 647–681.

Weighted Biclique Completion via CP-SDP Randomized Rounding

Stefano Gualandi **Federico Malucelli**

DEI, Politecnico di Milano
viale Ponzio 34/5
Milano, 20133, Italy

{gualandi,malucell}@elet.polimi.it

Abstract

In this work, we present a randomized rounding algorithm for solving the weighted k-Clustering Minimum Biclique Completion problem. This problem has a significant application in telecommunications for bundling channels in multicast transmissions. In the literature, it has been solved to optimality with an Integer Bilinear Programming formulation and a hybrid Constraint Programming and Semidefinite Programming approach. In this work, we consider for the first time the weighted version of the problem, and we propose a randomized rounding algorithm that exploits the Semidefinite Relaxation within a Constraint Programming solver.

Keywords: semidefinite programming, constraint programming, randomized rounding.

1. Introduction

The k-Clustering Minimum Biclique Completion (k-MinBCP) problem consists in partitioning an undirected bipartite graph into k clusters such that the sum of the edges that complete each cluster into a biclique, *i.e.*, a complete bipartite subgraph, is minimum. While the problem of covering undirected graphs by bicliques has been widely studied in the literature for its connections to factorization problems of 0/1 matrices (for a survey see [**9**]), the k-MinBCP problem has received little attention. This combinatorial optimization problem is NP-hard even for $k = 2$, as proven in [**3**]. In the literature, it has been tackled with Integer Bilinear Programming in [**3**] and with an hybrid Constraint Programming and Semidefinite Programming approach in [**7**].

The k-MinBCP problem has a significant application in telecommunications, as shown in [**3**], for bundling channels in multicast transmissions. In this paper, we consider a weighted version of k-MinBCP, where each edge of the complementary bipartite graph has a positive weight. Given a set of demands of services from clients, the application consists of finding k multicast sessions that partition the set of demands. Each service has to belong to a single multicast session, while each client can appear in more sessions. This problem is represented on a bipartite graph $G = (S, T, E)$ as follows: every service i is represented by a vertex in S, and every client j by a vertex in T. The demand of a service i from a client j is represented by the edge $(i, j) \in E$. In the case each service i requires a different bandwidth, we can weight the edges incident to an edge with a positive integer proportional to the required bandwidth. Hence, the cost of each cluster gives a measure of the waste of bandwidth of the corresponding multicast session. Solving the k-MinBCP problem on this bipartite subgraph, is equivalent to finding k multicast sessions that minimize the overall waste of bandwidth.

The relaxed (weighted) version of k-MinBCP is solved in this work with an extension of the SDP relaxation proposed in [**7**], but a simple cutting plane procedure is used to tighten the lower bound. In addition, the SDP relaxation is used to develop a randomized rounding heuristic enhanced by Constraint Programming. Preliminary computational results show that this approach yields very good heuristic solutions.

Notation. Let $G = (S, T, E)$ be an undirected bipartite graph, where $E \subseteq S \times T$. A **biclique** is a complete bipartite graph, that is, $E = S \times T$. The **complementary graph** of a bipartite graph is $\bar{G} = (S, T, \bar{E})$, with $\bar{E} = \{(i, j) \mid i \in S, j \in T, (i, j) \notin E\}$. The bipartite subgraph induced by two subsets $S' \subseteq S$ and $T' \subseteq T$ is denoted by $G[S', T'] = (S', T', E')$, where $E' = \{(i, j) \in E \mid i \in S', j \in T'\}$. The neighborhood of a vertex i is denoted by $N(i)$. The **one-shore-induced** bipartite subgraph induced by a subset $S'' \subseteq S$ is denoted by $G[S''] = (S'', T'', E'')$, where $T'' = \bigcup_{i \in S''} N(i)$, and $E'' = \{(i, j) \in E \mid i \in S'', j \in T''\}$.

2. Problem Description

Let $G = (S, T, E)$ be a bipartite undirected graph, let k be the number of desired clusters, and let every edge $e = (i, j)$ in $\bar{E}$ have a positive integer weight c_e. A subset S_p of S induces a bipartite subgraph $G[S_p]$ of cost $c_p = \sum_{e \in G[S_p]} c_e$. The weighted k-MinBCP problem consists of partitioning the set of vertices S into k subsets S_p, with $p = 1, \ldots, k$, such that the sum of the cost c_p of each one-shore-induced bipartite graph $G[S_p]$ is minimum.

Example 37.1. Figure 1.a shows a bipartite graph G with $S = \{1, \ldots, 4\}$ and $T = \{5, \ldots, 9\}$. Every edge incident to vertex 1 has weight 3, to vertex 2 has weight 4, to vertex 3 has weight 1, and to vertex 4 has weight 2. Figure 1.b represents a possible 2-clustering induced by $S_1 = \{1, 2\}$ and $S_2 = \{3, 4\}$. The dashed edges belong to the one-shore-induced subgraph $G[S_1]$ and those in bold to $G[S_2]$. Note that vertex 9 belongs to both clusters. The complementary graph $\bar{G}$ is shown in Figure 1.c. The cost of this 2-clustering is 12, given by three edges in $\bar{G}[S_1]$,i.e., $c_1 = 3 + (4 + 4) = 11$, and the edge in $\bar{G}[S_2]$, i.e., $c_2 = 1$.

Integer Bilinear Programming formulation. Let x_{ip} and y_{jp} be 0–1 variables indicating whether the vertex $i \in S$ and the vertex $j \in T$, respectively, are in cluster p. Let $K = \{1, \ldots, k\}$ be the set of the clusters. The Integer Bilinear

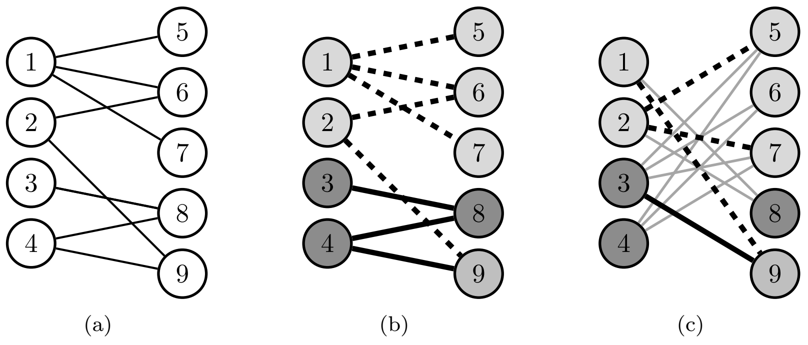


Figure 1. An example of the k-MinBCPproblem for $k = 2$.

Programming formulation of the weighted k-MinBCP problem is as follows:

$$w_k^* = \min \sum_{p\in K} \sum_{(i,j)\in \bar{E}} c_{ij}\, x_{ip}\, y_{jp} \tag{37.1}$$

$$\text{s.t.} \quad \sum_{p\in K} x_{ip} = 1, \qquad \forall i \in S, \tag{37.2}$$

$$y_{jp} \geqslant x_{ip}, \qquad \forall (i,j) \in E, \forall p \in K, \tag{37.3}$$

$$x_{ip} \in \{0,1\}, \qquad \forall i \in S, \forall p \in K, \tag{37.4}$$

$$y_{jp} \in \mathbb{R}^+, \qquad \forall j \in T, \forall p \in K. \tag{37.5}$$

The objective function (37.1) minimizes the sum of the edge weights that complete each cluster into a biclique. Constraints (37.2) assign each vertex of the shore S to a single cluster. Constraints (37.3) force each neighbor j of a vertex i to be (also) in the same cluster of i.

3. Semidefinite Programming relaxation

The SDP relaxation of k-MinBCP proposed in our previous work [**7**] is strictly related to the SDP relaxations of the Max-Cut problem proposed in [**6**], of the Max-k-Cut problem proposed in [**4**], and of the Min-k-Partition problem proposed in [**2**]. First, we review the weighted SDP relaxation for the case $k = 2$, and then we extend the formulation to the case $k > 2$. Finally, we investigate a stronger formulation that uses the triangle and clique inequalities.

Case $k = 2$. Let x_i be a $\{-1, 1\}$ variable that indicates whether the vertex $i \in S$ is in the cluster S_1 ($x_i = 1$), or in the cluster S_2 ($x_i = -1$). Note that if two vertices i and j of S are in the same cluster, then the product $x_i x_j$ is equal to 1, and, otherwise, is equal to -1.

The difficult part of the SDP formulation is to take into account which edges of the complementary bipartite graph do contribute to objective function. Let z_e be a $\{-1, 1\}$ variable equal to -1 if we pay for the complementary edge $e = (i,j) \in \bar{E}$, and equal to 1 otherwise. The objective function must account for a given edge $e = (i,j) \in \bar{E}$, if there is *at least* another vertex $l \in S$ such that $(l,j) \in E$ and l is in the same cluster of i. This relation is formulated with:

$$z_e = \max_{(l,j)\in E} \{x_i x_l\}, \quad \forall e = (i,j) \in \bar{E} \tag{37.6}$$

which can be restated with the following inequalities: $z_e \geqslant x_i x_l, \forall e = (i,j) \in \bar{E}, \forall (l,j) \in E$.

Using the vertex variables x_i and the edge variables z_e, the weighted 2-MinBCP problem is formulated as follows:

$$\begin{aligned}
&(37.7) && w_2 = \min \quad \sum_{e\in\bar{E}} c_e \frac{1+z_e}{2} && \\
&(37.8) && \text{s.t.} \quad z_e \geqslant x_i x_l, && \forall e = (i,j) \in \bar{E}, (l,j) \in E,\\
&(37.9) && x_i \in \{-1,1\}, && \forall i \in S,\\
&(37.10) && z_e \in \mathbb{R}, && \forall e \in \bar{E}.
\end{aligned}$$

We derive an SDP relaxation of problem (37.7)–(37.10) using the labeling technique proposed in [**6**] for the Max-Cut problem. Every vertex i of S is labeled with a unit vector $\mathbf{v}_i \in \mathbb{R}^{|S|}$. Then, two vertices i and j are considered to be in the same cluster if the angle between them is small enough, that is, if $\mathbf{v}_i^t \mathbf{v}_j = 1$. Let V be a matrix such that column i is given by vector $\mathbf{v}_i$, and let $X = V^t V$. Let $\mathbf{e} \in \mathbb{R}^{|S|}$ be a vector of all ones. The SDP relaxation of the weighted 2-MinBCP problem is as follows:

$$\begin{aligned}
&(37.11) && \min \quad \sum_{e\in\bar{E}} c_e \frac{1+z_e}{2} && \\
&(37.12) && z_e \geqslant X_{il}, && \forall e = (i,j) \in \bar{E}, (l,j) \in E,\\
&(37.13) && \operatorname{diag}(X) = \mathbf{e}, && \\
&(37.14) && X \succeq 0, && \\
&(37.15) && z_e \in \mathbb{R}, && \forall e \in \bar{E}.
\end{aligned}$$

Constraints (37.12) are equivalent to constraints (37.8), where each entry (i,l) of matrix X is used for the product $x_i x_l$. Constraints (37.13)–(37.14) together correspond to relax constraints (37.9) as $-1 \leqslant X_{ij} \leqslant 1$.

Case $k > 2$. We extend the relaxation (37.11)–(37.15) to the case $k > 2$ by using the same technique used in [**4**] to derive the SDP relaxation of the Max-k-Cut problem. The resulting model is similar to the min-k-partition formulation presented in [**8**].

Let us consider k unit vectors $\mathbf{a}_1, \ldots, \mathbf{a}_k \in \mathbb{R}^{k-1}$ satisfying $\mathbf{a}_i^t \mathbf{a}_j = -\frac{1}{k-1}$, for $1 \leqslant i \neq j \leqslant k$. We label each vertex $i \in S$ with a vector of real variable $\mathbf{x}_i$. The variable vector has the domain ranging in $\{\mathbf{a}_1, \ldots, \mathbf{a}_k\}$, i.e., $\mathbf{x}_i \in \{\mathbf{a}_1, \ldots, \mathbf{a}_k\}$. Two vertices i and j are in the same cluster if $\mathbf{x}_i^t \mathbf{x}_j = 1$, while are in different clusters whenever $\mathbf{x}_i^t \mathbf{x}_j = -\frac{1}{k-1}$. Note that for $k = 2$, we get exactly $a_1 = -1$, $a_2 = 1$, and $x_i \in \{-1, 1\}$.

Let z_e be again a variable that indicates whether we pay for the edge $e = (i,j) \in \bar{E}$. The value of z_e is assigned using the relation (37.6) defined on the vectors of variables $\mathbf{x}_i$ and $\mathbf{x}_j$. Note that in this case the possible values are either $z_e = -\frac{k}{k-1}$ if there exists an edge $(l,j) \in E$ such that the vertex l is in the same cluster of i, or $z_e = 1$ if such edge does not exist.

The formulation of the k-MinBCP problem is as follows:

$$w_k = \min \sum_{e \in \bar{E}} c_e \frac{1 + (k-1) z_e}{k} \tag{37.16}$$

$$\text{s.t.} \quad z_e \geqslant \mathbf{x}_i^t \mathbf{x}_l, \qquad \forall e = (i,j) \in \bar{E}, (l,j) \in E, \tag{37.17}$$

$$\mathbf{x}_i \in \{\mathbf{a}_1, \dots, \mathbf{a}_k\}, \qquad \forall i \in S, \tag{37.18}$$

$$z_e \in \mathbb{R}, \qquad \forall e \in \bar{E}. \tag{37.19}$$

Constraints (37.17) restate the relation (37.6). Note the the objective function is equal to either 1, if $z_e = 1$, or is equal to 0, if $z_e = \frac{-1}{k-1}$. The SDP relaxation of the k-MinBCP problem for $k > 2$ is obtained using the same labeling technique as for $k = 2$, and is as follows:

$$w_{sdp} = \min \sum_{e \in \bar{E}} c_e \frac{1 + (k-1) z_e}{k} \tag{37.20}$$

$$z_e \geqslant X_{il}, \qquad \forall e = (i,j) \in \bar{E}, (l,j) \in E, \tag{37.21}$$

$$\operatorname{diag}(X) = \mathbf{e}, \tag{37.22}$$

$$X_{ij} \geqslant -\frac{1}{k-1}, \qquad \forall i,j \in S, i \neq j, \tag{37.23}$$

$$X \succeq 0, \tag{37.24}$$

$$z_e \in \mathbb{R}, \qquad \forall e \in \bar{E}. \tag{37.25}$$

Together constraints (37.22)–(37.24) relax constraints (37.18) and (37.19) into $-\frac{1}{k-1} \leqslant X_{ij} \leqslant 1$.

Valid inequalities. In order to further improve the relaxation (37.20)–(37.24), we tried to strength the value of the SDP relaxation by generating violated valid inequalities, in the same vein of [**8, 5**].

Let i, j, l be three vertices in S. Since we are solving a clustering problem, if vertex i is in the same cluster of vertex j, and vertex j is in the same cluster of vertex l, then necessarily, vertex i and l are in the same cluster too. With respect with the formulation (37.16)–(37.19), this is equivalent to add the well–known triangle inequalities:

$$X_{ij} + X_{jl} \leqslant 1 + X_{il}, \tag{37.26}$$

$$X_{jl} + X_{il} \leqslant 1 + X_{ij}, \tag{37.27}$$

$$X_{il} + X_{ij} \leqslant 1 + X_{jl}. \tag{37.28}$$

Note that we have $3\binom{|S|}{3}$ triangles that can be easily enumerated.

In addition, we consider also the clique inequalities for the set of vertices S. The idea is that for any subset $C \subseteq S$, with $|C| = k+1$, at least two vertices must be in the same cluster. For the case $k = 2$, the clique inequalities reduce to:

$$X_{il} + X_{ij} + X_{jl} \geqslant -1 \tag{37.29}$$

For $k > 2$, the clique inequalities for $X_{ij} \in \{\frac{-1}{k-1}, 1\}$ are as follows [**5**]:

$$\sum_{i,j \in C : i < j} X_{ij} \geqslant -\frac{k}{2}, \quad \forall C \subseteq S, \text{ with } |C| = k+1. \tag{37.30}$$

Unfortunately, there are $\binom{|S|}{k+1}$ of such inequalities, and therefore must be generated via a cutting plane procedure.

4. CP-SDP Randomized Rounding

Constraint Programming (CP) is a programming paradigm for solving combinatorial problems that combines *expressive* modeling languages with *efficient* solver implementations. The two basic concepts of CP are *constraint propagation* and *constructive search.* Constraint propagation is an efficient inference mechanism aiming at reducing the domains of the problem variables by exploiting the semantics of the problem constraints. The inference mechanism is implemented into filtering algorithms that filter out values from the domain of each variable. When constraint propagation is unable to further reduce the domains of the variables, the constructive search comes into play. Constructive search explores the search tree by tightening the domain of a single variable at a time. In practice, it performs a search in the space of partial solutions. The simplest form of constructive search consists of selecting an undetermined variable, *i.e.*, a variable having more than a value in its domain, and labeling a value to that variable. This form of search is called *labeling.* By iterating constraint propagation and labeling, the CP solver computes the solution(s) of a given problem, or it reports that none exists. A reader not familiar with the basic notion of Constraint Programming can refer to [**1**] for an introductory textbook, or to [**10**] for the state-of-the-art on the research trends in CP.

The main idea of this work is to exploit the Semidefinite Programming relaxation obtained after the cutting plane procedure presented in the previous section, by devising a randomized rounding algorithm enhanced by the *constraint propagation* inherent in Constraint Programming. Although the idea is related to the exact hybrid method presented in [**7**], we are interested here in a heuristic solution.

In addition, remark that we are not proposing a new specific randomized algorithm with performance guarantees, but rather we propose a general framework for implementing randomized rounding algorithms on top of general purpose CP and SDP solvers.

4.1. The CP model

Let (Z_p,Y_p) be a pair of finite domain integer set variables that represents the vertices of the p-th cluster. Let c_p be an integer variable representing the cost of the p-th cluster, *i.e.*, the number of added edges, and let d be an integer variable for the sum of the cost of each cluster. The CP formulation of the k-MinBCP problem is as follows:

$$\begin{array}{lll}
(37.31) & \textit{variables/domains:} & Z_p \subseteq S, \qquad \forall p \in K, \\
(37.32) & & Y_p \subseteq T, \qquad \forall p \in K, \\
(37.33) & & 0 \leqslant c_p \leqslant |\bar{E}|, \quad \forall p \in K, \\
(37.34) & & 0 \leqslant d \leqslant |\bar{E}|, \\
(37.35) & \textit{constraints:} & \texttt{partition}([Z_1,\dots,Z_p],S), \\
(37.36) & & \texttt{osi-qbiclique}(Z_p,Y_p,c_p,G), \quad \forall p \in K, \\
(37.37) & & d = \sum_{p\in K} c_p.
\end{array}$$

Constraints (37.35) force each vertex of the shore S to appear in a single cluster. Constraints (37.36) constrain each subgraph induced by a pair (Z_p, Y_p) to be a cluster of cost equal to c_p. Constraint (37.37) sums up the cost of every cluster. The CP model (37.31)–(37.37) relies on the filtering algorithm used for the `osi-qbiclique` constraint, which was introduced in [**7**].

4.2. The CP-SDP Randomized Rounding algorithm

The basic scheme of the randomized rounding algorithm we propose is as follows:

(1) Start the execution of the CP solver running model (37.31)–(37.37), and let the inference mechanism of Constraint Programming to reduce the domains of the variables Z_i and Y_i as much as possible (i.e., until the CP solver has reached a fix point).
(2) Map the domain reduction achieved by the CP solver into a new problem instance, and solve the SDP relaxation (37.20)–(37.25) obtaining also the semidefinite matrix X^*.
(3) Generate the violated triangle and clique inequalities (37.26)–(37.28) and (37.30), respectively. Since the instances solved in literature are rather small, we manage to generate all the violated inequalities.
(4) Start the Randomized Rounding:
   (a) choose an undetermined variable Z_p, see (37.31);
   (b) select among the pairs of vertices v and w that appear the domain of variable Z_p and maximize the value X^*_{vw}, see (37.24);
   (c) assign both v and w to variable Z_p;
   (d) let the CP solver propagates the new assignment (i.e., to reach a fix point);
   (e) if all the variables Z_p are assigned stops, otherwise go back to step (4a).
(5) Go back to step (4) until a timeout is expired or a given number of rounds are executed.

The two crucial steps of the algorithm are (4a) and (4b). We are currently investigating alternative criteria for selecting both the variable Z_i and the pair of vertices v and w.

5. Further research

Currently, we are finalizing the implementation of the CP-SDP randomized rounding algorithm. Experimental results show that the SDP relaxation is tight and only few among the triangle and clique inequalities are violated in the relaxation. The CP-SDP randomized rounding produce good solutions on random bipartite graphs, but further work is required to compare our results with a standard randomized rounding (i.e., without constraint propagation). However, note that the proposed CP-SDP Randomized Rounding algorithm is quite general and can be adapted to other combinatorial optimization problems.

References

1. K.R. Apt. *Principles of Constraint Programming.* Cambridge University Press, 2003.
2. A. Eisenblatter. The semidefinite relaxation of the k-partition polytope is strong. In *Proc Integer Programming and Combinatorial Optimization*, volume LNCS 2337, pages 273–290. Springer, 2002.
3. N. Faure, P. Chrétienne, E. Gourdin, and F. Sourd. Biclique completion problems for multicast network design. *Discrete Optimization*, 4(3):360–377, 2007.
4. A. Frieze and M. Jerrum. Improved approximation algorithms for max k-cut and max-bisection. *Algorithmica*, 18:67–81, 1997.
5. B. Ghaddar, M.F. Anjos, and F. Liers. A branch-and-cut algorithm based on semidefinite programming for the minimum k-partition problem. *Annals of Operations Research*, pages 1–20, 2009. availabel on-line.
6. M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J ACM*, 42:1115–1145, 1995.
7. Stefano Gualandi. k-clustering minimum biclique completion via a hybrid CP and SDP approach. In *Proc Integration of AI and OR Techniques in CP for Combinatorial Optimization*, volume LNCS 5547, pages 87–101. Springer, 2009.
8. C. Helmberg and F. Rendl. Solving quadratic (0, 1)-problems by semidefinite programs and cutting planes. *Mathematical Programming*, 82(3):291–315, 1998.
9. S.D. Monson, N.J. Pullman, and R. Rees. A survey of clique and biclique coverings and factorizations of (0,1)-matrices. *Bulletin of the Combinatorics and its Applications*, 14:17–86, 1992.
10. F. Rossi, P. Van Beek, and T. Walsh. *Handbook of Constraint Programming.* Elsevier Science, 2006.

A Semidefinite Relaxation for the Queueing Covering Location Problem with an M/G/1 System

Hossein T. Kakhki **Foroogh M. Moghadas**

Department of Mathematics,
Ferdowsi University of Mashhad
Vakil Abad Blvd.
Mashhad, 91775-1159, Iran

taghizad@math.um.ac.ir

Abstract

We consider the queueing maximal covering location problem with an M/G/1 system. We formulate the problem as a binary quadratic program and present a semi-definite programming relaxation to obtain an upper bound for the problem. A lower bound for the problem is obtained by a Lagrangian relaxation of some coupling constraints which yields a mixed integer nonlinear program.
Keywords: queueing covering location, M/G/1, semidefinite relaxation.

1. Introduction

Queueing maximal covering location-allocation problem (QM-CLAP) is an extension of the maximal covering location problem (MCLP) of Church and ReVelle (1974) introduced by Marianov and Serra (1998). It can be stated as: given m demand points and their populations, find the location of p new *facilities* and the assignment of *demand points* to the new facilities so that the population covered is maximized. A demand point i is considered *covered* by a facility j, if its Euclidean distance to j is not greater than a given distance. Each demand point can be covered by at most one facility. In addition, the model requires that the probability of at most say b people being in line should be larger than or equal to a given α; or the probability of waiting time at a center being less than or equal to a maximum allowable time, τ , should be at least α.

A *less restricted* version suggested by them is that the expected number of customers in center j should be less than or equal to b; or the expected time spent at center j should be less than or equal to τ.

This last form is what we will be dealing with in this paper. Other variants of the MCLP problem, under M/M/1, M/M/m, M/G/1, M/G/k, M/G/s-loss queueing systems, sometimes called location in congested systems, have been considered by many researchers including Berman,Larson and Chiu (1985), Batta and Berman (1989), Marianov and ReVelle (1996), Marianov (2003); Baron, Berman and Krass (2008), Silva and Serra (2008), and Erkut, et.al (2009). A review of many of these models can be found in Goldberg (2004) and, Berman and Krass (2002).

In what follows, we first present a mathematical model for our problem. Then we show that for the M/G/1 system, the problem reduces to a quadratically constrained mathematical programming problem. By lifting the problem into a higher dimension, we obtain a semi-definite relaxation of our model.

2. Mathematical model

The model presented by Marianov and Serra (1998) is as follows:

$$\max \sum_{i\in I: j\in N_i} a_i x_{ij}$$

st:

$$x_{ij} \leq y_j \quad \forall i \in I, j \in N_i \tag{38.1}$$

$$\sum_{j\in N_i} x_{ij} \leq 1 \quad \forall i \in I \tag{38.2}$$

$$\sum_{j\in J} y_j \leq p \tag{38.3}$$

$$W_j \leq \tau_j \quad \forall j \in J \tag{38.4}$$

$$x_{ij}, y_j = 0,1 \quad \forall i \in I \ \ and \ \ j \in J \tag{38.5}$$

where,
$I :=$ the set of existing demand points
$J :=$ the set of candidate locations for new facilities
$N_i :=$ the set of points in the neighborhood of i ;i.e., $N_i = \{j|\ d_{ij} \leq R\}$
$y_j :=$ 1, if a new facility is located at site $j \in J$, and 0, otherwise
$x_{ij} :=$ 1 if a call from point i is answered by a center j; and 0, otherwise
$a_i :=$ population at point i
$p :=$ the maximum number of new centers
$W_j :=$ average waiting time at station j
$\tau_j :=$ maximum waiting time at a center j

The objective in this model maximizes the population coverage. Constraints (2.1) ensure that a demand point is being served only by an *open* facility. Constraints (2.2) guarantee that a demand point i is served at most by one new facility. Constraint (2.3) establishes p new facilities, and constraints (2.4) ensures that the average waiting time does not exceed a given threshold. Here we have assumed that this waiting time is the same for all centers.

Marianov and Serra (1998) propose a heuristic solution procedure for the M/M/1 case which is based on first satisfying the waiting time constraints. Silva and Serra (2008) propose a GRASP type heuristic also for the M/M/1 case. Recently Correa, Lorena and Ribeiro (2009) proposed a decomposition approach by first constructing a covering graph, then partitioning it using the routine MEITS of Karypis and Kumar (1998). The resulting block diagonal structure is then solved via Danzig-Wolfe decomposition approach.

Here we consider the case where the underlying system is an M/G/1 queueing system and try to formulate the problem as a binary quadratic program. A semidefinite programming relaxation is then obtained which provides an upper bound.

To develop our model note that since we assume that the arriving calls for demand is a Poisson process with intensity f_i , then the arriving calls at each center j would also be a Poisson process but with the rate $\lambda_j = \sum_{i\in I:j\in N_i} f_i x_{ij}$ (Marianov and Serra 1998). Also for the M/G/1 the waiting time is given by :

$$W_j = \frac{\lambda_j \overline{S_j^2}}{2(1-\lambda_j \overline{S_j}\ )}, \quad 1-\lambda_j\overline{S_j} > 0 \tag{38.6}$$

where, $\overline{S_j}$ and $\overline{S_j^2}$ are, respectively, the first and the second moments of the service time at center j. Hence, constraints (2.4) can be written as:

$$\lambda_j(\frac{1}{2}\overline{S_j^2} + \tau_j \overline{S_j}\ ) \le \tau_j \tag{38.7}$$

Then, we have:

$$\sum_{t\in I:j\in N_t}\sum_{i\in I:j\in N_i} h_i f_t(\frac{1}{2}\overline{T_{ij}^2} + \tau_j\overline{T_{ij}})x_{tj}x_{ij} \quad \le \quad \tau_j, \tag{38.8}$$

where h_i is the probability of a call originating from demand point i.

Therefore the M/G/1 model can be formulated as:

$$(P) \qquad \max \sum_{i\in I, j\in J} a_i x_{ij}$$

st:

(2.1)-(2.3), (2.5) and (2.8)

Note that in the above formulation we need to determine $\overline{T_{ij}}$ and $\overline{T_{ij}^2}$. To do so, we use the same definition as used by Berman, et.al. (1985), that is

$$\overline{T_{ij}} = \beta d_{ij}/v + \overline{Z_i} \tag{38.9}$$

$$\overline{T_{ij}^2} = (\beta d_{ij}/v + \overline{Z_i})^2 + 2(\beta d_{ij} v)\overline{Z_i} + \overline{Z_i}^2 \tag{38.10}$$

where d_{ij} is the Euclidean distance between demand point i and center at j; v is the speed, β is a parameter greater than 1; and $\overline{Z_i}$ is the average on-the-scene plus off-the-scene service time at demand point i.

3. An SDP relaxation

Since its early development in the 1990's, semi-definite programming (SDP) has found many applications in areas such as combinatorial optimization, control theory, convex optimization, and experimental design; See for example Wolkowicz, et al (2000) for a more comprehensive discussion. One particular application of SDP in combinatorial optimization and non-convex quadratic programming, has been its use as a relaxation to obtain bounds for these problems (see eg. Goemans and Rendl (2000), and Nesterov, et.al. (2000). A well known example is the Goemans and Williamson (1995) use of a SDP to obtain tight bounds for the max-cut problem. For more examples the interested reader can refer to Alizadeh (1991), Ding and Wolkowicz (2008) and Rendl (2010). For two matrices A and B let us define $A \bullet B$ as the *trace* of the product of A and B, $tr(AB)$. Also, let S^n denote the set of symmetric $n \times n$ matrices, and S^n_+ the set of $n \times n$ symmetric positive semi-definite matrices. Then by homogenizing the constraints, and defining C, B_i, D_j, and Z; respectively, as

$$C = \begin{bmatrix} 0 & \frac{1}{2}c^t \\ \frac{1}{2}c & \mathbf{0} \end{bmatrix}$$

$$B_i = \begin{bmatrix} -b_i & \frac{1}{2}A_i \\ \frac{1}{2}{A_i}^t & \mathbf{0} \end{bmatrix}$$

A_i is the i^{th} row of A. $\mathbf{O}$ here, is an $(mn+m) \times (mn+m)$ matrix of all zeroes, while 0 and 1 are scalars.

$$D_j = \begin{bmatrix} -\tau_j & 0 \\ 0 & Q_j \end{bmatrix}$$

and,

$$Z = \begin{bmatrix} x_0 \\ x \end{bmatrix} \begin{bmatrix} x_0 \\ x \end{bmatrix}^t = \begin{bmatrix} 1 & x^t \\ x & xx^t \end{bmatrix} = \begin{bmatrix} x_0 & x^t \\ x & Y \end{bmatrix}$$

Note that $x_0 = 1$ and that the constraint $Y = xx^t$ is not convex, so it is replaced by $Y \succeq xx$ or $Y - xx \succeq 0$ to have a convex constraint; We also relax the rank one restriction on Y. We would then have:

$$\begin{aligned} C \bullet Z = \begin{bmatrix} 0 & \frac{1}{2}c^t \\ \frac{1}{2}c & \mathbf{0} \end{bmatrix} \bullet \begin{bmatrix} x_0 \\ x \end{bmatrix} \begin{bmatrix} x_0 \\ x \end{bmatrix}^t &= tr \begin{bmatrix} 0 & \frac{1}{2}c^t \\ \frac{1}{2}c & \mathbf{0} \end{bmatrix} \begin{bmatrix} 1 & x^t \\ x & xx^t \end{bmatrix} \\ &= tr \begin{bmatrix} \frac{1}{2}c^t x & \frac{1}{2}c^t xx^t \\ \frac{1}{2}c & \frac{1}{2}cx^t \end{bmatrix} \\ &= c^t x \end{aligned}$$

Also

$$A_i x - b_i x_0 = \begin{bmatrix} -b_i & \frac{1}{2}A_i \\ \frac{1}{2}{A_i}^t & \mathbf{0} \end{bmatrix} \bullet \begin{bmatrix} x_0 \\ x \end{bmatrix} \begin{bmatrix} x_0 \\ x \end{bmatrix}^t = B_i \bullet Z$$

Similarly, we can write (2.8) as:

$$x^t Q_j x - \tau_j x_0^2 = \begin{bmatrix} -\tau_j & 0 \\ 0 & Q_j \end{bmatrix} \bullet \begin{bmatrix} x_0 \\ x \end{bmatrix} \begin{bmatrix} x_0 \\ x \end{bmatrix}^t = D_j \bullet Z$$

Note that C, B_j, D_j in $S^{k\times k}$ are sparse, and $Z \in S_+^{k\times k}$.

Finally, we replace the binary variables x_j by the nonlinear inequalities :

$$x_j(x_j - 1) \leqslant 0$$
$$-x_j(x_j - 1) \leqslant 0$$

Now, lumping the coefficient matrix of all the constraints, and denoting it by G_i, for simplicity, we would have the following so called *vector lifting semi-definite relaxation* of problem (P):

(38.11) $$(P'') \;\; max \;\; C \bullet Z$$

st:

(38.12) $$G_i \bullet Z \leq 0 \quad ,\forall i$$

(38.13) $$Z \succeq \mathbf{0}$$

(38.14)

Solving (P'') , will provide an upper bound for our problem (P).

References

1. Alizadeh, F. (1991) Combinatorial Optimization with Semi-Definite Matrices, in E. Balas, G. Cornuejols, and R. Kanan (eds.), *Integer Programming and Combinatorial Optimization*, pp. 385-405.
2. Baron, O., Berman, O., and Krass, D., (2008) Facility Location with Stochastic Demand and Constraints on Waiting Time, *Manufacturing & Operations Management* 10(3), 484-505.
3. Batta, R., and Berman, O., (1989) A Location Model for a Facility Operating as an M/G/k Queue, *Networks* 19, 717-728.
4. Berman, O., and Krass D. (2002) Facility Location problems with Stochastic Demands and Congestion, in Z. Drezner and H. W. Hamacher (eds.), *Facility location: Applications and Theory*, Springer, pp. 329-371.
5. Berman, O., Larson, R. C., and Chiu, S.S., (1985) Optimal Server Location on a Network Operating as an M/G/1 Queue, *Operations Research* 33 (4) 746-771.

6. Correa, F. D. A., Lorena, L. A. N., and Ribiero, G. M., (2009). A decomposition approach for the probabilistic maximal covering location-allocation problem, *Computers & Operations Research*, 36 (10), 2729-2739.
7. Ding, Y. and H. Wolkowicz, (2008) A Low Dimensional Semidefinte Relaxation for the Quadratic Assignment problem, *Optimization online* (April 19).
8. Erkut, E., Ingolfson, A., Sim, T., and Erdogan, G. (2009), Computational Comparison of Five Maximal Covering Models for Locating Ambulances, *Geographical Analysis* 41, 43-65.
9. Goemans, M.X., and Rendl, F. (2000) Combinatorial Optimization, in *Handbook of Semidefinite Programming: Theory, Algorithms, and Applications*, Kluwer Academic Publishers, pp. 343-360.
10. Goemans, M.X., and Willamson, D.P.(1994) 0.878-approximation algorithms for MAX CUT and MAX 2SAT, *ACM Symposium on Theory of Computing (STOC).*
11. Goldberg, J. B., (2004) Operations Research Model for the Deployment of Emergency Services Vehicles, *EMS Management Journal* 1 (1), 20-39.
12. Marianov, V. (2003), Location of Multiple Server Congestible Facilities for Maximizing Expected Demand when Services are Non-essential, *Annals of Operations Research* 123, 125-141.
13. Marianov, V., and ReVelle, C. (1996), The Queueing Maximal Availability Location problems: A Model for the Siting of Emergency Vehichles, *European Journal of Operational Research* 93, 110-120.
14. Marianov, V., and Serra, D. (1998) Probabilistic , Maximal Covering Location-Allocation Models for Congested Systems. *Journal of Regional Science* 38(3) , 401-424.
15. Nesterov, Y., Wolkowicz, H., and Ye, Yinyu (2000) Nonconvex quadratic optimization, in *Handbook of Semidefinite Programming: Theory, Algorithms, and Applications*, Kluwer Academic Publishers, pp.361-420.
16. Rendl, F. (2010) Semidefinite Relaxations for Integer Programming, in Jünger, M.; Liebling, Th.M.; Naddef, D.; Nemhauser, G.L.; Pulleyblank, W.R.; Reinelt, G.; Rinaldi, G.; Wolsey, L.A. (eds.), *50 Years of Integer Programming 1958-2008 From the Early Years to the State-of-the-Art*, Springer, pp. 687-726.
17. Silva, F., and Serra, D., (2008) Locating emergency services with different priorities : the priority queuing covering location problem, *Journal of the Operational Research Society* 59, 1229-1238.
18. Wolkowicz, H., Saigal, R. and Vandenberghe, L. (eds.) (2000), *Handbook of Semidefinite Programming: Theory, Algorithms, and Applications*, Kluwer Academic Publishers.

On the existence and efficient construction of general cutting planes for mixed integer quadratic programs

Thomas Lehmann

Department of Computer Science
University of Bayreuth
Bayreuth, 95440, Germany

thomas.lehmann@uni-bayreuth.de

Abstract

We propose an algorithmic procedure, that either constructs general cutting planes for mixed integer quadratic programs efficiently, or shows their non-existence. The method generalizes an algorithm presented by Balas and Perregard for efficient construction of disjunctive cutting planes, such that it is applicable for non-basic solutions.
Keywords: mixed integer quadratic programming, disjunctive cutting planes.

1. Disjunctive Programming

We want to solve the strictly convex mixed integer quadratic programming problem (MIQP) subject to linear equality and inequality constraints

$$\begin{array}{lll} \min\limits_{x\in X,\ y\in Y} & \frac{1}{2}\left(x^T, y^T\right) C \begin{pmatrix} x \\ y \end{pmatrix} + d^T \begin{pmatrix} x \\ y \end{pmatrix} & \\ s.t. & a_j^T \begin{pmatrix} x \\ y \end{pmatrix} + b_j = 0, & j = 1, \ldots, m_e, \\ & a_j^T \begin{pmatrix} x \\ y \end{pmatrix} + b_j \geqslant 0, & j = m_e + 1, \ldots, m, \end{array} \tag{39.1}$$

where x and y denote the vectors of the continuous and integer variables, respectively. The two sets X and Y are defined by

$$X := \{x \in \mathbb{R}^{n_c} : x_{lb} \leqslant x \leqslant x_{ub}\}, \tag{39.2}$$

$$Y := \{y \in \mathbb{N}^{n_i} : y_{lb} \leqslant y \leqslant y_{ub}\}, \tag{39.3}$$

(39.4)

where n_c is the number of continuous variables and n_i is the number of integer variables. The total number of variables is denoted by n, i.e., $n := n_i + n_c$. C is an n by n positive definite matrix, while d is an n-dimensional vector. The constraints are given by an m by n matrix $A = (a_1, \ldots, a_m)^T$, and an m-vector b. Lower and upper bounds for the continuous and integer variables are denoted by x_{lb}, x_{ub}, y_{lb} and y_{ub}.

Since the combination of branch and bound and cutting plane techniques led to a tremendous speed-up for solving mixed-integer linear programming problems (MILP), we want to apply this state-of-the-art technique to solve MIQP (39.1). Most cutting plane techniques relay either on the fact that the current relaxed solution to be cut off is a feasible basis solution, i.e., a corner of the feasible polyhedral set, or on special structures of some constraints or the whole feasible region. In general, the solution of a relaxed quadratic program is not basic, nor is special information about the constraints available in most cases. This is probably the main reason why only very few results on applying cutting planes for solving mixed-integer quadratic programming problems of the form (39.1) exist.

The mixed-integer nonlinear programming (MINLP) toolbox developed by Exler, Lehmann and Schittkowski [**4**] is our main motivation for considering convex MIQP problems. It is based on the mixed-integer sequential quadratic programming method MISQP [**3**] and contains very efficient solvers for realistic MINLP applications, associated with expensive function calls. All solvers rely on the successive solution of convex MIQP problems (39.1) and exhibit a excellent performance on an academic MINLP test case collection [**5**].

In Section 1 we introduce the theory of disjunctive programming going back to Balas [**2**] and we review a method for constructing disjunctive cutting planes in the original problem dimension, developed by Perregaard [**6**]. Section 2 proposes a generalized algorithm for constructing disjunctive cuts for non-basic solutions efficiently.

Consider the solution $(\bar{x}, \bar{y})$ of the continuous relaxation of a mixed-integer linear program. Let k be the index of an integer variable y_k possessing a fractional value at $(\bar{x}, \bar{y})$. Then we know that either $y_k \geqslant \lceil \bar{y}_k \rceil$ or $y_k \leqslant \lfloor \bar{y}_k \rfloor$ holds at the optimal solution (x^*, y^*) of the original mixed integer program. This condition can be expressed via a logical condition, which is called a disjunction $\vee$ and was introduced by Balas [**2**], i.e.,

$$y_k^* \geqslant \lceil \bar{y}_k \rceil \quad \vee \quad y_k^* \leqslant \lfloor \bar{y}_k \rfloor. \tag{39.5}$$

Using this disjunctive condition we can build the so-called disjunctive relaxation, which extends the continuous relaxation by requiring condition (39.5) to hold. The disjunctive relaxation can be used to generate cutting planes for the original problem. The reason is that there exists a compact representation of the convex hull of the union of $h \in Q$ polyhedra in a higher dimension, where Q is a finite index set. This convex hull can be projected back to the original space yielding so-called disjunctive cutting planes.

Originally these cutting planes are constructed by solving the so-called cut generating linear program ($CGLP_k$) corresponding to disjunction (39.5):

$$\begin{aligned}
&\min_{(a,\beta)\in\mathbb{R}^n\times\mathbb{R},\ (u,u_0)\in\mathbb{R}^m\times\mathbb{R},\ (v,v_0)\in\mathbb{R}^m\times\mathbb{R}} && a^T\begin{pmatrix}\bar{x}\\ \bar{y}\end{pmatrix}-\beta\\
&s.t. && a-A^Tu+u_0e_k\geqslant 0,\\
&(39.6) && a-A^Tv-v_0e_k\geqslant 0,\\
& && -\beta+b^Tu-u_0\lfloor\bar{y}_k\rfloor=0,\\
& && -\beta+b^Tv+v_0\lceil\bar{y}_k\rceil=0,\\
& && u,\ u_0,\ v,\ v_0\geqslant 0,
\end{aligned}$$

together with a normalization constraint of the form

$$\sum_{i=1}^m u_i+u_0+\sum_{i=1}^m v_i+v_0=1. \tag{39.7}$$

The optimization variables (a,β) correspond to the coefficients of the resulting cutting plane, whereas the variables (u,u_0,v,v_0) ensure the validity of this constructed cut.

The simple disjunctive cut $\pi^Ts_J\geqslant\pi_0$ is also derived from disjunction (39.5) for some fractional y_k and is given by

$$\begin{aligned}
\pi_0 &:= (\bar{y}_k-\lfloor\bar{y}_k\rfloor)(\lceil\bar{y}_k\rceil-\bar{y}_k),\\
\pi_i &:= \max\{\pi_i^1,\pi_i^2\},\quad\forall i\in J,\\
\pi_i^1 &:= (\bar{y}_k-\lceil\bar{y}_k\rceil)(A_J^{-1})_{ki},\\
\pi_i^2\ " &:= (\bar{y}_k-\lfloor\bar{y}_k\rfloor)(A_J^{-1})_{ki},
\end{aligned}\tag{39.8}$$

where J is the index set of the basic variables.

The equivalence of the simple disjunctive cut (39.8) and a basic solution of the $CGLP_k$ (39.6), which was established by Balas and Perregard [**1**], can be used to solve the cut generating linear program implicitly in the original problem dimension. This procedure suggested by Perregard [**6**] yields substantial advantages since it decreases the computational effort significantly. Apart from working within the original problem dimensions, the computation time is significantly decreased by the reduction of degeneracy. The reason is that the partition of J into M_1 and M_2, introduced in the subsequent theorem, can be automatically determined by the sign of $(A_J^{-1})_{kj}$.

Theorem 1.1. Let (a,β,u,u_0,v,v_0) be a non-trivial basic, feasible solution to the $CGLP_k$ (39.6), i.e., $u_0,v_0>0$ and (a,β) basic. Assume that the basic components of u and v are indexed by M_1 and M_2 respectively. Let $\bar{s}$ correspond to the surplus variables of $A\begin{pmatrix}x\\ y\end{pmatrix}\geqslant b$ corresponding to the solution $(\bar{x},\bar{y})$. After introducing standard notation

$\bar{a}_{kj}:=-(e_kA_J^{-1})_j,\quad \bar{a}_{ij}:=-(a_iA_J^{-1})_j,\quad \bar{a}_{k0}:=e_kA_J^{-1}b_J,\quad \bar{a}_{i0}:=(a_iA_J^{-1}b_J-b_i),$

the reduced costs for u_i and v_i, $i \notin J \cup \{k\}$ at this basic solution are respectively

$$\begin{aligned} r_{u_i} &:= -\sigma - \tau - \bar{a}_{i0}(\bar{y}_k - \lceil \bar{y}_k \rceil), \\ r_{v_i} &:= -\sigma + \tau + \bar{a}_{i0}(\bar{y}_k - \lceil \bar{y}_k \rceil) + \bar{s}_i, \end{aligned} \tag{39.9}$$

with

$$\sigma := \frac{\sum_{j \in M_2} \bar{a}_{kj}\bar{s}_j + (\bar{a}_{k0} - \lfloor \bar{y}_k \rfloor)(\bar{y}_k - \lceil \bar{y}_k \rceil)}{1 + \sum_{j \in J} |\bar{a}_{kj}|}, \tag{39.10}$$

$$\tau := \sigma \sum_{j \in M_1} \bar{a}_{ij} - \sigma \sum_{j \in M_2} \bar{a}_{ij} + \sum_{j \in M_2} \bar{a}_{ij}\bar{s}_j, \tag{39.11}$$

where $\bar{s}_j$ denotes the value of the slack variable s_j of constraint j at $(\bar{x}, \bar{y})$

Theorem 1.1 can be used to identify unaccounted constraints, that strengthen the current simple disjunctive cut. If all reduced costs (39.9) are positive the $CGLP_k$ (39.6) is implicitly solved to optimality. The next theorem shows how one can identify the constraint that should be removed from the current simple disjunctive cut to be replaced by another improving constraint.

Theorem 1.2. The pivot column in row i of the LP simplex tableau that is most improving with respect to the cut from row k is indexed by $l^* \in J$, that minimizes $f^+(\gamma_l)$ if $\bar{a}_{kl}\bar{a}_{il} < 0$ or $f^-(\gamma_l)$ if $\bar{a}_{kl}\bar{a}_{il} > 0$ over all $l \in J$, where $\gamma_l := -\frac{\bar{a}_{kl}}{\bar{a}_il}$ and for any γ,

$$\begin{aligned} f^+(\gamma) &= \frac{\sum_{j \in J}(\max\{0, -(\bar{a}_{kj} + \gamma\bar{a}_{ij})\}\bar{s}_j)}{1 + |\gamma| + \sum_{j \in J} |\bar{a}_{kj} + \gamma\bar{a}_{ij}|} \\ &+ \frac{(\lceil \bar{a}_{k0} + \gamma\bar{a}_{i0} \rceil - \bar{a}_{k0} - \gamma\bar{a}_{i0})(\lfloor \bar{a}_{k0} + \gamma\bar{a}_{i0} \rfloor - \bar{y}_k)}{1 + |\gamma| + \sum_{j \in J} |\bar{a}_{kj} + \gamma\bar{a}_{ij}|}, \end{aligned}$$

$$\begin{aligned} f^-(\gamma) &= \frac{\sum_{j \in J}(\max\{\gamma\bar{a}_{ij}, -\bar{a}_{kj}\}\bar{s}_j)}{1 + |\gamma| + \sum_{j \in J} |\bar{a}_{kj} + \gamma\bar{a}_{ij}|} \\ &+ \frac{(\lceil \bar{a}_{k0} + \gamma\bar{a}_{i0} \rceil - \bar{a}_{k0} - \gamma\bar{a}_{i0})(\lceil \bar{a}_{k0} + \gamma\bar{a}_{i0} \rceil - \bar{y}_k) + \bar{a}_{k0} - \lceil \bar{a}_{k0} + \gamma\bar{a}_{i0} \rceil}{1 + |\gamma| + \sum_{j \in J} |\bar{a}_{kj} + \gamma\bar{a}_{ij}|}. \end{aligned}$$

2. Disjunctive Cutting Planes for Non-basic Solutions

The aim of this section is to construct disjunctive cutting planes efficiently for non-basic solutions $(\bar{x}, \bar{y})$ of the continuous MIQP relaxation, i.e., less than n constraints are active, or to proof that none exists for the current disjunction.

The straight forward way to construct disjunctive cutting planes by solving the full $CGLP_k$ for every fractional integer variable is inefficient. Therefore we want to generalize the method presented in the last section, such that it is applicable for non-basic solutions. One key property is the correspondence of the simple disjunctive cut (39.8) and a feasible, basic solution of the $CGLP_k$ (39.6). Since we cannot construct a simple disjunctive cut for a non-basic solution a priori no starting point is known.

Once we are able to provide a feasible, basic solution of the $CGLP_k$, we can apply the method of Balas and Perregard without further adjustments. Consider a basic solution $(\tilde{x}, \tilde{y})$, determined by all $n_a < n$ constraints, that are active in $(\bar{x}, \bar{y})$.

Additionally $n - n_a$ constraints are chosen, such that all constraints, indexed by J, are linearly independent.

If we now consider a fractional integer variable $\bar{y}_k$, two possibilities arise. Either $\lfloor \bar{y}_k \rfloor = \lfloor \tilde{y}_k \rfloor$ or $\lfloor \bar{y}_k \rfloor = \lfloor \tilde{y}_k \rfloor + l$ with $l \in \mathbb{Z},\ l \neq 0$ holds. In case one we already found a feasible basic solution to start, since the simple disjunctive cut determined by $\tilde{y}_k$ is a basic, feasible solution for the $CGLP_k$ corresponding to $(\bar{x}, \bar{y})$.

In general if $\bar{y}_k$ and $\tilde{y}_k$ lie within different disjunctions, i.e., $\lfloor \bar{y}_k \rfloor = \lfloor \tilde{y}_k \rfloor + l$ with $l \in \mathbb{Z},\ l \neq 0$, $(\tilde{x}, \tilde{y})$ is no basic feasible solution of the $CGLP_k$, since the constraints determining the right hand side β of the resulting cutting plane are violated.

By replacing one of the constraints in J by an artificial constraint, denoted by

$$a_a \begin{pmatrix} x \\ y \end{pmatrix} \geqslant b_a, \tag{39.12}$$

we can construct a basic solution $(\hat{x}, \hat{y})$ lying in the correct disjunction, i.e., $\lfloor \bar{y}_k \rfloor = \lfloor \hat{y}_k \rfloor$ holds. This basic solution $(\hat{x}, \hat{y})$ is in principle an appropriate starting point for the algorithm presented in the last section.

The drawback is that the simple disjunctive cut determined by $(\hat{x}, \hat{y})$ is not valid for the original quadratic program, since the additional constraint (39.12) is not part of the constraints of MIQP (39.1). The subsequent lemma shows a situation where the artificial constraint does not influence the original method.

Lemma 2.1. Let the basic solution $(\hat{x}, \hat{y})$ satisfy $\lfloor \bar{y}_k \rfloor = \lfloor \hat{y}_k \rfloor$. Furthermore let one of the constraints, say $a_a \begin{pmatrix} x \\ y \end{pmatrix} \geqslant b_a$, determining $(\hat{x}, \hat{y})$ be artificial, i.e., it is not part of the original set of constraints. Suppose that the simple disjunctive cut corresponding to $(\hat{x}, \hat{y})$ does not cut off the relaxed solution $(\bar{x}, \bar{y})$. Furthermore assume that the reduced costs for every constraint, inactive at $(\hat{x}, \hat{y})$, are positive. Then no disjunctive cut for $(\bar{x}, \bar{y})$ exists in the current disjunction.

Therefore we have to find a way to construct a basic solution that yields no simple disjunctive cutting plane for the relaxed solution $(\bar{x}, \bar{y})$.

The trivial basic solution of the $CGLP_k$, given u_0 and v_0 satisfying $u_0 v_0 = 0$, corresponds to a simple disjunctive cut, that is a non-negative linear combination of the original constraints. Furthermore this simple disjunctive cut is determined by a basic solution $(\hat{x}, \hat{y})$ with $\hat{y}_k = \lfloor \bar{y} \rfloor$ or $\hat{y}_k = \lceil \bar{y} \rceil$. Therefore it satisfies the requirements of Lemma 2.1. Furthermore the simple disjunctive cut corresponding to this trivial basic solution $(\hat{x}, \hat{y})$ can be considered as the limit with $\epsilon \to 0$ for $\hat{y}_k + \epsilon = \lceil \bar{y}_k \rceil$ of the simple disjunctive cut (39.8) or analogue as the limit $\epsilon \to 0$ for $\hat{y}_k - \epsilon = \lfloor \bar{y}_k \rfloor$.

In principle it is possible to use $(\hat{x}, \hat{y})$ as starting point, since all terms of the reduced algorithm can be determined by the corresponding basis matrix A_J and its inverse.

As we will see in Lemma 2.2, the artificial constraint (39.12) has to possess an additional property. Consider a constraint $a'_a \begin{pmatrix} x \\ y \end{pmatrix} \geqslant b'_a$ that is determined such that $(\hat{x}, \hat{y})$ is the unique optimal solution of the linear program

$$\begin{aligned} \min_{x \in X, y \in Y_{\mathbb{R}}} \quad & a'_a \begin{pmatrix} x \\ y \end{pmatrix} \\ s.t. \quad & A_J \begin{pmatrix} x \\ y \end{pmatrix} = b_J, \end{aligned} \tag{39.13}$$

where $Y_{\mathbb{R}} := \{y \in \mathbb{R}^{n_i} : y_{lb} \leqslant y \leqslant y_{ub}\}$ holds. As a consequence $(\hat{x}, \hat{y})$ is the only point satisfying all constraints $A_J \begin{pmatrix} x \\ y \end{pmatrix} \geqslant b_J$ and $a'_a \begin{pmatrix} x \\ y \end{pmatrix} \geqslant b'_a$. We now define the artificial constraint by $a_a := -a'_a$ and $b_a := -b'_a$.

Lemma 2.2. Let $(\hat{x}, \hat{y})$ be a basic solution with $\hat{y}_k = \lfloor \bar{y}_k \rfloor$. Let the corresponding basic variables be indexed by J and the corresponding basic matrix be denoted by A_J. Suppose that A_J contains one artificial constraint $a_a \begin{pmatrix} x \\ y \end{pmatrix} \geqslant b_a$, which is determined as stated above and indexed by a, while all other constraints are indexed by $1, \ldots, n-1$, i.e., $J = \{1, \ldots, n-1, a\}$. Furthermore let there be an improving simple disjunctive cut, which is constructed by replacing constraint s : $a_s \begin{pmatrix} x \\ y \end{pmatrix} \geqslant b_s$, $s \in J$ with constraint j, $a_j \begin{pmatrix} x \\ y \end{pmatrix} \geqslant b_j$, $j \notin J$.

Then an improving cut is also determined by the constraints $a_i \begin{pmatrix} x \\ y \end{pmatrix} \geqslant b_i$, $\forall i \in J \backslash \{a\}$ and $a_j \begin{pmatrix} x \\ y \end{pmatrix} \geqslant b_j$.

Lemma 2.1 and Lemma 2.2 allow the generalization of the algorithm introduced in the previous section for non-basic solutions. Applying Lemma 2.1, we can proof that no disjunctive cuts exists for the current disjunction (39.5). Otherwise Lemma 2.2 guarantees, that we can construct a disjunctive cut analogue to the method of Perregaard in the original problem dimensions.

References

1. Perregaard M. Balas E. A precise correspondence between lift-and-project cuts, simple disjunctive cuts, and mixed integer gomory cuts. *Mathematical Programming*, 94:221–245, 2003.
2. Balas E. Disjunctive programming. *Annals of Discrete Mathematics*, 5:3–51, 1979.
3. Schittkowski K. Exler O. A trust region sqp algorithm for mixed-integer nonlinear programming. *Optimization Letters*, 1:269–280, 2007.
4. Schittkowski K. Exler O., Lehmann T. An outer approximation algorithm for nonlinear mixed-integer programming based on sequential quadratic programming with trust region stabilization. *to appear.*
5. Schittkowski K. A collection of 100 test problems for nonlinear mixed-integer programming in fortran - user's guide. *Report, Department of Computer Science, University of Bayreuth*, 2009.
6. Perregaard M. Generating disjunctive cuts for mixed integer programs. *Doctoral Dissertation, Carnegie Mellon University Pittsburgh, USA*, 2003.

40

SOC avatars for solving MINLPs (in 3-D)

Ashutosh Mahajan **Todd S. Munson**

Argonne National Laboratory,
9700 S Cass Avenue,
Argonne, IL 60439, USA
`{mahajan,tmunson}@mcs.anl.gov`

A Mixed Integer Nonlinear Program (MINLP) is a mathematical problem of finding the best solution for a system of nonlinear inequalities when some or all variables are constrained to certain discrete values. A large variety of problems arising in design, planning and operations of complex systems like energy-distribution, basic infrastructure, engineering design, networks, etc., are expressed as MINLPs. Most of these problems have constraints that are nonconvex and it makes these problems hard to solve. In this work, we devise a reformulation strategy by which certain nonconvex constraints can be reformulated after branching as Second Order Cones (SOCs). Efficient nonlinear solvers can then be used to quickly search over these convex regions.

When a constraint only has quadratic and linear functions, we use a reformulation technique based on eigenvalue decomposition. Besides giving us a branching variable that converts the constraint into an SOC, this reformulation can also be used to identify if the problem is infeasible or if a constraint is redundant. We also extend this technique to other types of functions: higher order polynomials, exponential functions and multilinear constraints. In each of these cases, we reformulate the problem in such a way that the new formulation becomes convex after branching. We also describe automatic procedures for reformulating such constraints.

We show by some computational experiments that for problems with such constraints, our technique can be faster than existing state-of-the-art MINLP solvers by several orders of magnitude. We also provide a geometric view of how this technique works with the help of 3-D stereographic images.

Tightening the Linear Relaxation of a Mixed Integer Nonlinear Program Using Constraint Programming

Sylvain Mouret[1] **Ignacio Grossmann**[1] **Pierre Pestiaux**[2]

Carnegie Mellon University
5000, Forbes Avenue
Pittsburgh, 15213, USA
smouret,grossmann@cmu.edu

Total Refining & Marketing
Z.I. du port autonome du Havre
76700, Harfleur, France
pierre.pestiaux@total.com

Keywords: mixed-integer nonlinear programming, constraint programming, McCormick convex envelope, crude-oil operations scheduling.

Many optimization problems arising in the chemical industry involve nonconvex nonlinear functions which makes them difficult to solve to global optimality. In this work, we study the crude-oil scheduling problems introduced in [**1**]. It consists in preparing various types of crude-oil blends throughout the horizon in order to continuously feed each crude distillation unit (CDU), while satisfying the demand for each crude blend. The scheduling system is composed of crude tankers, storage and charging tanks, and CDUs. Three types of crude transfer operations are available: unloading from crude-oil marine vessels to storage tanks, transfer between tanks, and transfers from charging tanks to CDUs. The goal is to determine which operations will be executed, how many times and when they will be performed, and the amount of crude to be transferred. The objective is to minimize the logistics costs which include sea waiting costs and unloading costs for marine vessels, storage costs in tanks, and CDU switching costs.

The problem is formulated as a nonconvex mixed integer nonlinear programming (MINLP) model developed in [**3**] which is based on a continuous-time scheduling representation. The model involves scheduling, sequencing, inventory, demand, and CDU specification constraints. The objective function is nonlinear as it contains bilinear terms expressing the storage costs in inventories.

The solution procedure proposed is a simple two-step MILP-NLP procedure. In the first step, an MILP relaxation is solved. The solution returned during this step may not satisfy all nonlinear constraints. In such case, the binary variables of the models are fixed, which means that the sequence of operations is fixed, and the resulting nonlinear programming (NLP) model is solved. This NLP model contains all constraints from the MINLP, including nonlinear constraints, with all binary variables fixed. The solution obtained during this step might not be the global optimum of the full model, but the optimality gap can be estimated from the lower bound given by the MILP solution and the upper bound given by the NLP solution. The quality of the final solution obtained with this procedure, estimated by the optimality gap, strongly depends on the tightness of the MILP relaxation.

The MINLP is classically reformulated by replacing the bilinear terms in the objective function with new variables, which are defined by new bilinear constraints. Therefore, the objective function becomes linear and the bilinearities are embedded in the constraints. Using variable lower and upper bounds, these bilinear terms can be replaced by their corresponding McCormick convex envelopes [**2**], leading to a tighter MILP relaxation.

However, rather than than applying these McCormick constraints at the root node, they are generated *on the fly* during the MILP Branch & Bound search. At each node, whenever an integer feasible is found, new variables bounds can be generated for the corresponding node subproblem using Constraint Programming (CP). These new bounds are tighter than the bounds obtained at the root node as many decision variables have already been fixed, and can be used to generate new tighter McCormick cuts. Those that are violated are then added to the current node subproblem, which is iteratively re-solved in a cutting-plane fashion until no new constraints can be added. The solution obtained for the node subproblem is generally more accurate as the linear relaxation of the bilinear terms is tighter, and therefore it is more suitable for the NLP step.

The four problems introduced in [**1**] have been solved using two algorithms. The *BasicRelaxation* algorithm consists of initially adding McCormick constraints to the MILP model without generating new cuts during the search. The *ExtendedRelaxation* algorithm consists of adding McCormick constraints to the MILP and new cuts during the search. Both approaches have been developed in C++ using Ilog Cplex 11.0 (MILP) and Ilog Solver 6.5 (CP). The NLPs have been solved with CONOPT 3.

The results show that using the *ExtendedRelaxation* approach leads to important reductions of the optimality gap compared to the *BasicRelaxation* algorithm (3.48% vs 14.83% average optimality gap). Besides, a better feasible solution (3.3% cost reduction) has been found for problem 2 when using *ExtendedRelaxation* algorithm.

In terms of computational efficienty, the average CPU time increases by 9.5% for the *ExtendedRelaxation* procedure. This is due to the increase of the number of nodes explored in some cases (problems 2 and 3), the increase of the model size for subproblems for which McCormick cuts have been generated, and the time used for performing constraint propagation on the CP model (4.7% of total CPU time).

Finally, both approaches are compared with state-of-the-art NLP local and global solvers. Although the smallest problem 1 remains solvable by local optimizers, these solvers are not suitable for directly solving any of the 3 larger problems.

References

1. H. Lee, J. M. Pinto, I. E. Grossmann, and S. Park. Mixed-integer linear programming model for refinery short-term scheduling of crude oil unloading with inventory management. *Industrial and Engineering Chemistry Research*, 35(5):1630–1641, 1996.
2. G. P. McCormick. Computability of global solutions to factorable nonconvex programs: Part 1 - convex underestimating problems. *Mathematical Programming*, 10:147–175, 1976.
3. S. Mouret, I. E. Grossmann, and P. Pestiaux. A novel priority-slot based continuous-time formulation for crude-oil scheduling problems. *Industrial and Engineering Chemistry Research*, 48(18):8515–8528, 2009.

42

Using DRL* relaxations for quadratically constrained pseudoboolean optimization: application to robust Min-Cut

Michel Minoux   **Hacene Ouzia**

LIP6, Université de Paris 6
ave. P. Kennedy 75016, Paris, France

hacene.ouzia@lip6.fr

Keywords: mixed integer programming, pseudoboolean optimization, Min-Cut, robust optimization, Reformulation-Linearization-Technique.

In this work we focus on solving quadratically constrained pseudoboolean optimization problems with quadratic objective as mixed integer linear programs. The standard mixed integer linear formulation of such problems is strengthened using valid inequalities derived from solving Reformulation-Linearization relaxation called partial DRL* relaxation. The proposed PDRL* relaxation features block-decomposable structure which are exploited to improve computational efficiency. We present computational results obtained with the rank 2 PDRL*, showing that the proposed mixed integer linear formulation gives rise to significant reduction factors (typically more than 1000) in the size of the branch and bound trees on instances of robust minimum cut problem with weight constraints.

A t-linearization to exactly solve 0-1 quadratic knapsack problems

Carlos-Diego Rodrigues[1,2] **Dominique Quadri**[1]
Philippe Michelon[1] **Serigne Gueye**[3]

[1] LIA, Université d'Avignon
Avignon, F-84911 Cedex 9, France
{carlos-diego.rodrigues,dominique.quadri,philippe.michelon}@univ-avignon.fr

[2] Universidade Federal do Ceará
Campus do Pici, Fortaleza, Brasil

[3] LMAH, Université du Havre
25 rue Philippe Lebon, BP 540, 76058 Le Havre
serigne.gueye@univ-lehavre.fr

Abstract

This paper presents an exact solution method based on a new linearization scheme for the 0-1 Quadratic Knapsack Problem (QKP) which consists of maximizing a quadratic pseudo-Boolean function with non negative coefficients subject to a linear capacity constraint. Contrasting to traditional linearization schemes, our approach only adds one extra variable. We first convert (QKP) into an equivalent problem using only one additional real decision variable t and a quadratic constraint. We then replace the additional quadratic constraint by a set of linear constraints derived from the characterization of the induced integer quadric hull. The linear relaxation of the resulting problem (called the t-relaxation) provides an upper bound used in branch-and-bound scheme. This upper bound is numerically compared with the Billionnet et al. [**2**] bound, and the Branch-and-Bound scheme with the exact algorithm of Pisinger et al. [**12**]. The experiments show that our upper bound is competitive with the best upper bound method known [**2**] for (QKP) (less than 1% from the optimum). In addition, the proposed branch-and-bound clearly outperforms the exact algorithm [**12**] for low density instances (25% and 50%) for all problem sizes up to 300 variables.

Keywords: 0-1 Quadratic Knapsack Problem, linearization techniques, upper bounds, exact solution method.

1. Introduction

Knapsack problems are among the most famous and studied problems in Operations Research [**10**], due to their simplicity in definition, hardness in solving [**6**] and broadly applicability in both practical and theoretical problems [**8**]. The $0-1$ Knapsack Problem can be defined as $\{\max c^t x : ax \leqslant b,\ x \in \{0,1\}^n\}$, where the inclusion of the object in the knapsack is defined by the decision vector x and $a, b, c \in \mathbb{R}^{n+}$.

When modeling the cases where the combinations of objects are more important than their individual inclusion in the knapsack, we usually consider the quadratic knapsack problem, which has been shown to be NP-hard [**9**] and can be expressed as follows

$$(QKP)\ \left\{\begin{array}{ll} \max & \sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} q_{ij} x_i x_j \\ s.t. & \left| \begin{array}{l} \sum_{i=1}^{n} a_i x_i \leqslant b \\ x_i \in \{0,1\}\ \ i = 1, ..., n \end{array}\right. \end{array}\right.$$

where q_{ij} $(i = 1, ..., n-1,\ j = i+1, ..., n)$, c_i, a_i $(i = 1, ..., n)$ and b are positive reals.

The quadratic summation is nonlinear and can not be treated easily. In order to solve this problem, a common approach is to linearize the objective function (see [**5**], [**7**], [**4**]). The linearization procedures proposed in the literature imply a large number of additional variables in comparison with the initial program. In order to get around this diffuclty, we propose here a orginal linearization scheme (called t-linearization) where only one extra variable is added.

The next section is dedicated to briefly describe the t-linearization. Section 3 describes the bound procedure. In the Section 4 the branch-and-bound algorithm is presented. Section 5 presents numerical experiments. We finaly conclude in Section 6.

2. t-linearization scheme

The t-linearization technique we propose consists of two steps. We first replace the quadratic terms of the objective function of (QKP) by a real variable t and add a quadratic constraint to the initial capacity constraint of (QKP). The second step is dedicated to rewrite the additional quadratic constraint as a set of linear ones. These are derived from the characterization of the quadric convex hull resulting from the linearization of the unconstrained 0-1 quadratic program.

Since all coefficients q_{ij} are positive, (QKP) can be rewritten as

$$(LP(Q))\ \left\{\begin{array}{ll} \max & t + \sum_{i=1}^{n} c_i x_i \\ s.t. & \left| \begin{array}{l} \sum_{i=1}^{n} a_i x_i \leqslant b \\ t \leqslant \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} q_{ij} x_i x_j \\ t \in \mathbb{R},\ x_i \in \{0,1\}\ \ i = 1, ..., n \end{array}\right. \end{array}\right.$$

where c_i, q_{ij}, a_i and b are positive reals.

Problems (LP(Q)) and (QKP) are equivalent. The objective function is now linear nevertheless the additional constraint is quadratic. Therefore, the second step of the proposed linearization framework consists of replacing this constraint by a set of linear constraints. This step is a straight consequence of the following Theorem.

Theorem 3. Consider the two following sets:

$$X = \left\{ (x,t) \in \mathbb{R}^{n+1} \mid \ t \leqslant \sum_{1 \leqslant i < j \leqslant n} q_{ij} x_i x_j, x \in \{0,1\}^n \right\}$$

and

$$QP_n = \left\{ \ (x,t) \in \mathbb{R}^{n+1} \mid \ t \leqslant \sum_{i=2}^{n} \sum_{j=1}^{i-1} q_{\pi(j)\pi(i)} x_{\pi(i)}, \ x \in [0,1]^n \ \forall \ \pi \ \in \ S_n \right\}$$

where S_n is the set of all the permutations of $\{1, ..., n\}$.

We have $QP_n = Co(X)$ ($Co(X)$ represents the convex hull of the set X). [1]

Since the convex hull is origninaly defined for the unconstrained 0-1 quadratic problem (cf. Theroem 1), we have to extend the result to the constraint case and incorporate the capacity constraint relative to (QKP). We can now establish the 0-1 linear program denoted by (LP) which is equivalent to (QKP) :

$$(LP) \left\{ \begin{array}{ll} \max \ t + \sum_{i=1}^{n} c_i x_i & \\ s.t. \left| \begin{array}{l} \sum_{i=1}^{n} a_i x_i \leqslant b \\ t \leqslant \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} q_{\pi(j)\pi(i)} x_{\pi(i)}, \ \forall \pi \in S_n \\ t \in \mathbb{R}, \ x_i \in \{0,1\} \ , \ i = 1, ..., n \end{array} \right. & \end{array} \right.$$

Constrasting to the unconstrained case the separation problem is more difficult to solve. Consequently, we propose to work with the relaxation of the integer variables of (LP) and compute heuristically only a subset of suitable constraints that is to say constraints which design as well as possible the quadric polytope. Hence, a relaxation method is proposed, called t-relaxation and an upper bound of (LP) is obtained that implies an upper bound of (QKP). We detail this t-relaxation in the following section.

3. Deriving an upper bound: the t-relaxation

The upper bound method we suggest for (QKP) is provided by the optimal value of $\overline{LP}$ which is the continuous variables relaxation of (LP) (i.e. $x_i \in [0,1]$, $i = 1, ..., n$) for which a subset of suitable constraints is computed. A key step of the resolution of the relaxation of (LP) is the generation of these constraints based on the different permutations. Consequently we propose a constraint generation method. Given a permutation π of the variables, we generate the suitable constraint $t \leqslant \alpha_\pi x$ and test whether it is violated or not. If so, we add it to the continuous linearized model and solve it. This process should be repeated until we cannot find a permutation that gives a violated constraint. Figure 2 provides an outline of the main steps of this procedure.

Step 3 is the retrieving of a permutation π from $\overline{x}$ the optimal solution of $\overline{LP}$. This is done by ordering the indices of the variables in a non-decreasing order of x. As a secondary criterium we use the reduced costs c provided by the optimal solution of $\overline{LP}$. Since the separation problem is solved by an heuristic, there is

[1]The proof of the present theorem can be found in the Appendix section.

1. Find a permutation of the variables.
To initiate the procedure we choose the permutation associated to a feasible solution we compute using the heuristic of Billionnet and Calmels [**1**].
The corresponding constraint is added to $\overline{LP}$.

2. Solve the resultant $\overline{LP}$.

3. Solve heuristically the separation problem. Let π the permutation found.

4. If the constraint corresponding to π is violated
then add the constraint to $\overline{LP}$ and go to step 2 else stop.

Figure 1. Main steps to compute the proposed upper bound for (QKP)

no garantee that a violated constraint may be found. In step 4, we check if either or not the permutation of step 3 discards the current optimal solution. In case of success, we add :

$$t \leqslant \alpha_\pi x = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} q_{\pi(i)\pi(j)} x_{\pi(i)}$$

This step corresponds to heuristically solving the separation problem of our constraint generation phase in order to obtain the t-linearization upper bound.

Since the generation of the constraint is heuristic we suggest to improve the quality of the constraints used. That can be done by adding to the separation problem the knapsack constraint. In addition one could also consider to include the integrality constraint to the separation problem model. This would result in a knapsack problem. The resolution of a knapsack problem can be very time consuming when compared to its LP-relaxed counterpart. Indeed it is possible to solve the LP-relaxation of a knapsack problem in linear time on the number of variables (see [**10**]), while the original problem is NP-hard. Therefore, it seems knowledgeable for us to work with the LP-relaxation of the knapsack problem. This ensures that we take into account the domain of the original problem while not having to solve $O(n)$ NP-hard problems at each constraint generation iteration. The separation problem, for each coefficient in the generated constraint, can be expressed as follow:

$$\alpha_\pi(j) = \max \left\{ \sum_{i=1}^{j-1} q_{\pi(i)\pi(j)} x_i \quad : \quad \sum_{i=1}^{j-1} a_i x_i \leqslant b - a_j, \quad x \in [0,1]^n \right\}. \tag{43.1}$$

4. The developped branch-and-bound algorithm

The branch-and-bound algorithm we propose for solving the linearized problem (LP(Q)), and hence (QKP), starts by finding a feasible solution x_H to the problem. The chosen heuristic is the one presented by Carprara et al. [**3**]. This solution x_H allows us to build a permutation π_H given by a non-increasing order of x_H. As tie-breakers we used the ratio: $\frac{c_i + \sum_{j=1}^{n} q_{ij}}{a_i}$ defined for each variable x_i $\forall i = 1, ..., n$. Then the constraint $t \leqslant \alpha_{\pi_H} x$ is added to our root node model.

Before the branching phase, as the constraint generation procedure could generate too many constraints we often consider to select a subset of constraints to carry over for the descendent nodes. Here we will do an aggregation of the constraints in the node in order to pass only one constraint over. This will severally reduce the size of our model, allowing us to have more nodes at the same time in the branch-and-bound tree and reducing the time to solve the relaxed model itself. As a disadvantage, some constraints may have to be generated again in the children nodes, this time stronger, depending on which values the variables were assigned.

The aggregation procedure for a set of constraint indexes K consists of adding a new constraint $t \leqslant \tilde{\alpha}_K x$ on which:

$$\tilde{\alpha}_K(j) = \sum_{i \in K} \frac{\overline{\mu_i} \cdot \alpha_{\pi_i}(j)}{\sum\limits_{k \in K} \overline{\mu_k}}$$

where $\overline{\mu_i}$ is the dual value resulting from the linear relaxation solution related to the constraint $t \leqslant \alpha_{\pi_i} x$. Let $K^\star$ be the index set that includes all indexes of generated constraints. We can show that $\sum_{k \in K^\star} \overline{\mu_k} = 1$. That simplifies the expression for each $\tilde{\alpha}_{K^\star}(j)$:

$$\tilde{\alpha}_{K^\star}(j) = \sum_{i \in K^\star} \overline{\mu_i} \cdot \alpha_{\pi_i}(j).$$

Moreover, we can show that the linear relaxed model in which all constraints $t \leqslant \alpha_{\pi_i} x$ for $i \in K^\star$ are substituted for $t \leqslant \tilde{\alpha}_{K^\star} x$ has the same objective value than the model with all the constraints in $K^\star$. To prove this, consider the problem in which variable x is fixed to $\overline{x}$, the linear problem solution. Then we have $\overline{t} = \{\max t : t \leqslant \alpha_{\pi_i} \cdot \overline{x}, \forall i \in K^\star\}$. From the Strong Duality Theorem, we have that

$$\sum_{i \in K^\star} (\alpha_{\pi_i} \cdot \overline{x}) \cdot \overline{\mu_i} = \overline{t}.$$

At this point it should be clear that $c^T \overline{x} + \overline{t}$ is the optimal value of the t-relaxation model.

Hence, the constraint $t \leqslant \tilde{\alpha}_{K^\star} x$ is in our branch-and-bound the only constraint passed down to the children nodes. The branching rule follows the classical criterium of least integral variable, the one which linear solution value is the closest to 0.5.

5. Numerical experiments

In this section we conduct several computational experiments to test the performance of both our upper bound method and our exact solution algorithm. For these experiments, we use the same instances as in Billionnet et al. [**2**] and the same generator as in Pisinger et al. [**12**].

More specifically, we report computational results concerning: (i) the quality of our upper bound and the computational in comparison with the lagrangean upper bound proposed by Billionnet et al. [**2**]; (ii) the computational performance of the proposed branch-and-bound compared to that of the nowadays best branch-and-bound algorithm known Pisinger et al. [**12**].

Table 1. Comparison of the quality of the upper bounds

Method	Opt	[**2**]		*t*-relaxation		Integral *t*-relaxation		
No.		UB	Gap(%)	UB	Gap(%)	UB	Gap(%)	NGC
1	18558	18910.56	1.90	19124.1	3.05	18865	1.65	109
2	56525	56574.63	0.09	56576	0.09	56525	0.00	32
3	3752	3807.68	1.48	3900.53	3.96	3785	0.88	312
4	50382	50448.08	0.13	51064.5	1.35	50589	0.41	580
5	61494	61623.22	0.21	61621	0.21	61494	0.00	6
6	36360	36464.87	0.29	36654.9	0.81	36399	0.11	283
7	14657	14749.58	0.63	14853.5	1.34	14657	0.00	445
8	20452	20525.15	0.36	20528.5	0.37	20452	0.00	46
9	35438	35485.16	0.13	35487	0.14	35438	0.00	361
10	24930	25191.5	1.05	25496.9	2.27	20190	1.04	401
Average	**32255**	**32378.04**	**0.63**	**32530.7**	**0.85**	**32339.6**	**0.26**	**257.5**

We coded our upper bound method as well as the proposed branch-and-bound algorithm in C++ and all experiments were conducted on a Pentium 4 2.66 GHz Intel processor and 1024 MB of RAM, exepted the results relative to [**2**] and [**12**].

5.1. Upper bounds comparison

The quality of our upper bound provided by the proposed *t*-relaxation is compared to that of : (i) the upper method suggested by Billionnet et al. [**2**] (which is based on a Langrangean decomposition); (ii) the Integral *t*-relaxation we developped (which consists of solving the final *t*-linearized model with $0-1$ variables instead of continuous variables).

The comparison of the quality of those three bounds procedures are reported in Table 1. The experiments has been performed on a HP9000 for [**2**] and on a Pentium 4 2.66 GHz Intel processor and 1024 MB of RAM. Because of the difference between the CPU used by Billionnet et al. [**2**] and for our upper bound methods, comparing these two CPU times is not fair or representative. Thus, in our tables, processing times have been omitted.

The three upper bound methods (cf. Table 1) were tested on instances with 100 variables and 25% density on the quadratic member, as presented in [**2**].

Table 1 displays the resulting upper bounds values for each method and the deviation (denoted by gap in %) of each upper bound to the optimal value (which is refered in column Opt). In addition, the number of generated constraints (column NGC) is given for the *t*-linearization procedure.

The bound provided by [**2**] is of better quality than the bound given by the *t*-linearization. Nevertheless, the last remains, in average, within less than 1% from the optimum. When the integer variables are considered, the Integral *t*-linearization bound is clearly better (the gap is lesser than 0.5%). We can also state that the number of generated constraints is smaller than the total number of permutations ($O(n!)$). In the light of the average deviation (0.26%) of the integral *t*-linearization, as well as the little average number of additional constraints (257.5), it may be observed that our method is able to describe properly the optimal solution neighborhood with few constraints compared to the $O(n^2)$ constraints of the classic linearization.

Table 2. Average time for 100 and 200-variables instances in seconds

	t-linearization		[**12**]			t-linearization		[**12**]	
Instance Set	Time	Opt	Time	Opt	Instance Set	Time	Opt	Time	Opt
GHS100.25	0.78	10	210.7	10	GHS200.25	42.32	10	860	9
GHS100.50	1.54	10	54.2	10	GHS200.50	13.09	10	168.9	10
GHS100.75	0.41	10	6.7	10	GHS200.75	27.84	10	23	10
GHS100.100	0.2	10	2.7	10	GHS200.100	397.11	10	76.5	10
Average	**0.73**	100%	**68.56**	100%	**Average**	**120.09**	100.0%	**267.28**	97.5%

Table 3. Average time for each 300 and 400-variables instances in seconds

	t-linearization		[**12**]			t-linearization		[**12**]	
Instance Set	Time	Opt	Time	Opt	Instance Set	Time	Opt	Time	Opt
GHS300.25	74.92	10	4031	10	GHS400.25	87.97	10	1190.1	9
GHS300.50	1480.18	9	556.8	10	GHS400.50	1468.31	10	978.3	10
GHS300.75	2134.21	10	94.7	10	GHS400.75	2698.21	9	275.0	10
GHS300.100	2163.94	9	90.3	10	GHS400.100	516.45	9	173.2	10
Average	**1444.43**	95%	**1193.2**	100%	**Average**	**1170.92**	95%	**640.41**	97.5%

5.2. Exact solution methods

We compare in this section the computational performance of the proposed branch-and-bound based on a t-linearization with the branch-and-bound algorithm presented by Pisinger et al.[**12**].
A time limit of twelve hours was imposed to each instance.

All instances were generated as in [**11**], [**3**] and [**12**]. Coefficients c_i and q_{ij} of the objective function are integers uniformly distributed between 0 and 100, the weights a_i of the constraint are uniformly distributed between 0 and 50. The right-hand side of the constraint, b, is a random number between 50 and $max\{50, \sum_{i=1}^{n} a_i\}$. CPU times are presented in seconds (s) and concern the average time of exact solution of 10 instances in a given set.

We compare the CPU time given by our algorithm with the best results known for this problem. These results are due to Pisinger et al. [**12**]. It is important to notice that the instance sets used in these comparisons are not the same, but generated similarly. The computational environement is different, as well. Pisinger et al. [**12**] use a Pentium IV 2.4Ghz with 1GB RAM. Pisinger et al. [**12**] computing quality is sightly lower than ours but remains very close to our environment. Thus, a fair comparison between the two methods is possible.

In Tables 2 and 3 we can see that our branch-and-bound clearly outperforms the algorithm proposed by Pisinger et al. [**12**] for the smallest problem sizes ($n = 100$) whatever the density. For 75% and 100% density and 200 variables the branch-and-bound suggested by Pisinger et al. [**12**] behaves better than the solution method we present. Nevertheless, on average our branch-and-bound is approximately 2 times faster than the second approach [**12**] for $n = 200$. For 300 and 400 variables, the reduction proposed by [**12**] keeps the problem size small enough to guarantee a good performance in average. This means that only the low density (25%) instances are outperformed by the t-linearization approach.

6. Conclusion

We developed in this paper a branch-and-bound algorithm to solve the 0-1 quadratic knapsack problem (QKP). The upper bound method suggested is based on a original linearization scheme requiring only one extra variable. The resulting upper bound is competitive with the best upper bound method known [**2**] for (QKP). Our branch-and-bound clearly outperforms other existing methods for low density instances (25% and 50%) for small problems (100 and 200 variables). A possible way to improve the CPU time of our method would consist of incorporating the reduction procedure proposed by [**12**]. This a non trivial task, requiring mathematical and algorithmical analysis that will be the subjects of future developments.

References

1. A. Billionnet and F. Calmels. Linear programming for the 0-1 quadratic knapsack problem. *European Journal of Operational Research*, 92:310–325, 1996.
2. A. Billionnet, A. Faye, and E. Soutif. An upper bound for the 0-1 quadratic knapsack problem. *European Journal of Operational Research*, 112:664–672, 1999.
3. A. Caprara, D. Pisinger, and P. Toth. Exact solution of the quadratic knapsack problem. *INFORMS Journal on Computing*, 11:125–137, 1999.
4. W. Chaovalitwongse, P.M. Pardalos, and O.A. Prokopyev. A new linearization technique for multi-quadratic 0-1 programming problems. *Operations Research Letters*, 32:517–522, 2004.
5. R. Fortet. Application de l'algèbre de boole en recherche opérationelle. *Revue Française de la Recherche Opérationelle*, 4:17–25, 1960.
6. M.R. Garey and D.S. Johnson. *Computers and intractability. A guide to the theory of NP-completness.* San Francisco, 1979.
7. F. Glover. Improved linear integer programming formulations of nonlinear integer problems. *Management Science*, 22(4):455–460, 1975.
8. H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems.* Springer Verlag, 2005.
9. G.S. Lueker. Two np-complete problems in nonnegative integer programming. *Computer Science Labatory, Princenton, NJ*, report 178 (A6), 1975.
10. S. Martello and P. Toth. *Knapsack Problems: Algorithms and computer implementations.* John Wiley and Sons Inc, 1990.
11. P. Michelon and L. Veilleux. Lagrangean methods for the 0-1 quadratic knapsack problem. *European Journal of Operational Research*, 92:326–341, 1996.
12. W.D. Pisinger, A.B. Rasmussen, and R. Sandvik. Solution of large quadratic knapsack problems through aggressive reduction. *INFORMS Journal on Computing*, 19(2):280–290, 2007.

Appendix

Theorem 4. Consider the two following sets:

$$X = \left\{ (x,t) \in \mathbb{R}^{n+1} \mid t \leqslant \sum_{1 \leqslant i < j \leqslant n} q_{ij} x_i x_j, x \in \{0,1\}^n \right\}$$

and

$$QP_n = \left\{ (x,t) \in \mathbb{R}^{n+1} \mid t \leqslant \sum_{i=2}^{n}\sum_{j=1}^{i-1} q_{\pi(j)\pi(i)} x_{\pi(i)},\ x \in [0,1]^n\ \forall\, \pi \in S_n \right\}$$

where S_n is the set of all the permutations of $\{1, ..., n\}$.

We have $QP_n = Co(X)$ ($Co(X)$ represents the convex hull of the set X).

Proof. Let us first show that $QP_n \subset Co(X)$. Let $(x,t) \in QP_n$. We have to show that :

$$\exists\, p \in N,\ (x^i, t^i) \in X\ ,\ \beta_i \geqslant 0\ (0 \leqslant i \leqslant p) \mid \sum_{i=0}^{p} \beta_i (x^i, t^i) = (x,t)\ ,\ \sum_{i=0}^{p} \beta_i = 1.$$

Without loss of generality, we can suppose that the components of x are sorted in decreasing order. Let $\alpha_1, \alpha_2, ..., \alpha_m$ be the values (also sorted in decreasing order) taken by the component of x. Then:

$$x = (\alpha_1, \alpha_1, ..., \alpha_1, \alpha_2, \alpha_2, ..., \alpha_2, ..., \alpha_m, \alpha_m, ..., \alpha_m).$$

More formally, by denoting k_j the greatest index l such that $x_l = \alpha_j$ ($l \in \{1, ..., n\}$ and $j \in \{1, ..., m\}$), we can write:

$$x_l = \alpha_j \quad , \quad k_{j-1} < l \leqslant k_j$$

with $k_0 = 0$ and $\alpha_0 = 1 - \alpha_1$.

Now, let us denote by x^0 the null vector of R^n and by x^i $(1 \leqslant i \leqslant m)$ the point of R^n defined by:

$$x_l^i = \begin{cases} 1, & \text{if } l \leqslant k_i \\ 0, & \text{otherwise} \end{cases}$$

We then have $x = \sum\limits_{i=1}^{m-1} (\alpha_i - \alpha_{i+1}) x^i + \alpha_m x^m + \alpha_0 x^0$. Hence, by considering that $p = m$ and

$$\beta_i = \begin{cases} \alpha_0 = 1 - \alpha_1, & \text{if } i = 0 \\ \alpha_i - \alpha_{i+1}, & \text{if } 1 \leqslant i \leqslant p-1 \\ \alpha_m, & \text{if } i = p \end{cases}$$

it follows that $x = \sum\limits_{i=0}^{p} \beta_i x^i$ with $\beta_i \geqslant 0$ (since $1 \geqslant \alpha_1 > ... > \alpha_m \geqslant 0$) and $\sum\limits_{i=0}^{p} \beta_i = \beta_0 + (\alpha_1 - \alpha_2) + (\alpha_2 - \alpha_3) + ... + \alpha_m = \beta_0 + \alpha_1 = 1$.

Now, for an index $i = 1, ..., p$ and a scalar $\delta \leqslant 0$, let us consider the following value:

$$t_\delta^i = \sum_{l=1}^{n-1} \sum_{u=l+1}^{n} q_{lu} x_l^i x_u^i + \delta = \sum_{l=2}^{n} \sum_{u=1}^{l-1} q_{ul} x_l^i x_u^i + \delta = \sum_{l=2}^{k_i} \sum_{u=1}^{l-1} q_{ul} + \delta$$

by the construction of t_δ^i and by the definition of X, $(x^i, t_\delta^i) \in X$.

In addition, by defining $t^0_\delta = \delta$, we have $(x^0, t^0_\delta) \in X$. Thus, it remains to prove that $\exists\ \delta \geqslant 0$ such that $t = \sum\limits_{i=0}^{p} \beta_i t^i_\delta$.

Given that $(x,t) \in QP_n$, we have

$$t \leqslant \sum_{l=2}^{n} \sum_{u=1}^{l-1} q_{\pi(u)\pi(l)} x_{\pi(l)}, \quad \forall\ \pi\ \in\ S_n.$$

In particular, for the identity permutation, we obtain the following inequality

$$t \leqslant \sum_{l=2}^{n} \sum_{u=1}^{l-1} q_{ul} x_l = \sum_{i=1}^{p} \alpha_i \Big(\sum_{l=k_{i-1}+1}^{k_i} \sum_{u=1}^{l-1} q_{ul} \Big).$$

Let $\delta = t - \sum\limits_{i=1}^{p} \alpha_i (\sum\limits_{l=k_{i-1}+1}^{k_i} \sum\limits_{u=1}^{l-1} q_{ul})$. We then have $\delta \geqslant 0$ and it is a fastidious but straightforward calculation to see that $\sum\limits_{i=0}^{p} \beta_i t^i_\delta = t$. Hence, (x,t) is written as a convex combination of points of X.

Conversely we need to show that $Co(X) \subset QP_n$. For this purpose, it is sufficient to establish that any of the inequalities defining QP_n is valid for X, or, in other words, we have to establish that

$$\forall (x,t) \in X,\ \forall \pi \in S_n\ t \leqslant \sum_{i=2}^{n} \sum_{j=1}^{i-1} q_{\pi(j)\pi(i)} x_{\pi(i)}.$$

Since $(x,t) \in X$, we have $t \leqslant \sum\limits_{i=1}^{n-1} \sum\limits_{j=i+1}^{n} q_{ij} x_i x_j$ and therefore, it is suficient to show that

$$\forall (x,t) \in X,\ \forall \pi \in S_n \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} q_{ij} x_i x_j \leqslant \sum_{i=2}^{n} \sum_{j=1}^{i-1} q_{\pi(j)\pi(i)} x_{\pi(i)}.$$

This is easily achieved by noting that

$$\begin{aligned} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} q_{ij} x_i x_j &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} q_{\pi(i)\pi(j)} x_{\pi(i)} x_{\pi(j)} \\ &= \sum_{i=2}^{n} x_{\pi(i)} \Big(\sum_{j=1}^{i-1} q_{\pi(i)\pi(j)} x_{\pi(j)} \Big) \end{aligned}$$

for any permutation π and by taking into account that $x_{\pi(j)} \in \{0,1\}$ and $q_{\pi(j)\pi(i)} \geqslant 0$ (so that $q_{\pi(j)\pi(i)} x_{\pi(j)} \leqslant q_{\pi(j)\pi(i)}$).

$\square$

Optimization to measure performance in the Tailorshop test scenario — structured MINLPs and beyond

Sebastian Sager Carola M. Barth Holger Diedam
Michael Engelhart Joachim Funke

Interdisciplinary Center for Scientific Computing
Department of Psychology
University of Heidelberg
INF 368, 69120 Heidelberg, Germany

sebastian.sager@iwr.uni-heidelberg.de

Abstract

Obtaining objective means to measure performance is of crucial importance in the research field Complex Problem Solving. While for traditional tests like the Tower of Hanoi the correct solutions were known, this is more difficult for modern, complex, simulation-based test scenarios, as the *Tailorshop*. We derive a problem class of nonconvex mixed-integer nonlinear programs (MINLPs) which stem from such economic test scenarios. In a round based scenario participants need to make decisions. A posteriori a performance indicator is calculated and correlated to their ability of emotion regulation. We solve altogether 2088 optimization problems with different size and initial conditions. They are based on real world experimental data from 12 rounds of 174 participants. The goals are twofold: first, from the solutions we gain additional insight into a complex system, which facilitates the analysis of a participant's performance in the test. Second, we propose a methodology to automatize this process by providing a new criterion based on the solution of a series of optimization problems. We disprove the assumption that the "fruit fly of complex problem solving", the *Tailorshop* scenario that has been used for dozens of published studies, is not mathematically accessible. By providing a detailed mathematical description and the computational tool *Tobago* [**12**] for an optimization-based analysis we hope to foster further interdisciplinary research between psychologists and applied mathematicians and provide a source for benchmarking of MINLP solvers.

Keywords: mixed integer programming, nonlinear programming, cognitive psychology.

1. Introduction

Psychologists define complex problem solving as a *high-order cognitive process.* The complexity may result from one or several different characteristics, such as a coupling of subsystems, nonlinearities, dynamic changes, intransparency, or others [**6**]. The main intention of the research field *complex problem solving* of human beings is the desire to understand how certain *variables* influence a solution process. In general, *personal and situational variables* are differentiated. In our study we analyze the personal variable *emotion regulation.* Other interesting personal variables are *working memory*, *amount of knowledge*, and *intelligence.*

Psychologists have been working in the research fields of problem solving for approximately 80 years. Since the 1970s and 1980s also computer-based test scenarios are in use. The overall idea, compared to early works in problem solving, is still the same: one evaluates the performance of a participant by calculating an *indicator function* and either correlates it to personal attributes or analyzes the influence of different experimental conditions for groups of participants. The main difference is that for the early test scenarios the correct solution is known at every stage. For more complex scenarios the performance evaluation is not so straightforward. The availability of an objective performance indicator is an obstacle for analysis and it has often been argued that inconsistent findings are due to the fact that

> *"...it is impossible to derive valid indicators of problem solving performance for tasks that are not formally tractable and thus do not possess a mathematically optimal solution. Indeed, when different dependent measures are used in studies using the same scenario (i.e., Tailorshop* [**7, 13, 11**]*), then the conclusions frequently differ."*

as stated by Wenke and Frensch [**15**, p.95]. The *Tailorshop* is sometimes referred to as the "Drosophila" for problem solving researchers [**9**] and thus a prominent example for a computer-based test scenario. In Section 2 we will derive a mathematical model for the *Tailorshop.* In Section 3 we will discuss mathematically optimal solutions, and finally formulate a valid indicator function in Section 4.

To our knowledge, numerical optimization methods have only scarcely been used for the analysis of participants' decisions. The general approach to compare performance to optimal solutions has been discussed by [**10**]. However, the authors do not provide a mathematical model for their test scenario *EPEX.* Hence, they need to use the software as a black box for brute-force simulation or derivative free strategies, such as Nelder-Mead. Such strategies result in significantly higher computational runtimes, give less insight, and have poor theoretical convergence properties.

2. Tailorshop MINLP Model

The *Tailorshop* has been developed and implemented as a test scenario in the 1980s by Dörner [**6**]. It has been used in a large number of studies. Also comprehensive reviews on studies and results in connection with the *Tailorshop* have been published, e.g., [**8**].

A participant has to take economic decisions to maximize the profit of a small company, specialized in the production and sales of shirts. The scenario comprises twelve rounds (months), in which the participant can modify infrastructure (employees, machines, distribution vans), financial settings (wages, maintenance, prices), and logistical decisions (shop location, buying raw material). As feedback he gets some key indicators in the next round, such as the current number of sold shirts, machines, employees, and the like. Arrows next to the indicators show if the value increased or decreased with respect to the previous round.

We derive a mathematical formulation as an optimization problem. The basic idea is that for different initial values (the current state in round n_{s} of a participant's test run) the optimal solution for the remaining $N - n_{\mathrm{s}}$ rounds can be calculated. The optimal solution can then either be used for a manual comparison and analysis of the participant's decisions, Section 3, or for an automated indicator function, as discussed in Section 4.

The *Tailorshop* has been developed as a test scenario in `GW-Basic` code. On the basis of this code we derived a mathematical optimization problem for a participant and month $0 \leq n_{\mathrm{s}} < N$ as

$$\tag{44.1}\begin{array}{llcll} \max\limits_{x,u,s} & F(x_N) & & & \\ \text{s.t.} & x_{k+1} & = & G(x_k, u_k, s_k, p), & k = n_{\mathrm{s}} \dots N-1, \\ & 0 & \leq & H(x_k, x_{k+1}, u_k, s_k, p), & k = n_{\mathrm{s}} \dots N-1, \\ & u_k & \in & \Omega, & k = n_{\mathrm{s}} \dots N-1, \\ & x_{n_{\mathrm{s}}} & = & x^p_{n_{\mathrm{s}}}. & \end{array}$$

The model is dynamic with a discrete time $k = 0 \dots N$, where $N = 12$ is the number of rounds. The control vector $u_k = u(k)$ has 15 (or 13 when van purchase is fixed) entries for each $k = 0 \dots N-1$ corresponding to the decisions the participant can make in the test. The vector of dependent state variables $x_k = x(k)$ comprises 16 entries. We define

$$(x^{\mathrm{p}}, u^{\mathrm{p}}) = (x^{\mathrm{p}}_0, \dots, x^{\mathrm{p}}_N, u^{\mathrm{p}}_0, \dots, u^{\mathrm{p}}_{N-1})$$

to be the vector of decisions and state variables for all months of a participant. Certain entries $x^p_{n_{\mathrm{s}}}$ enter (44.1) as fixed initial values. The goal is to find decisions u_k that maximize the overall balance at the end of the time horizon. The objective function is given by $F(x_N) = x^{OB}_N$. The resulting problem is a nonconvex mixed-integer nonlinear program with n_{s}–dependent dimension.

3. Optimization and numerical results

We want to solve a series of optimization problems of the form (44.1) for different participant data that has been obtained experimentally.

3.1. Implementation

To be able to analyze and visualize the data in a convenient way, to have a simulation environment for own studies, and to be able to automatize the optimization of all $2088 = 174 \cdot 12$ problems, we implemented the software framework *Tobago* [**12**]. It is publically available under an open source license, includes a GUI, and may as well be used for experimental setups. This data generation and analysis tool can be hooked to a variety of optimization solvers. Currently the software supports *AMPL* interfaces. This allows for the usage of solvers from the *COIN-OR*

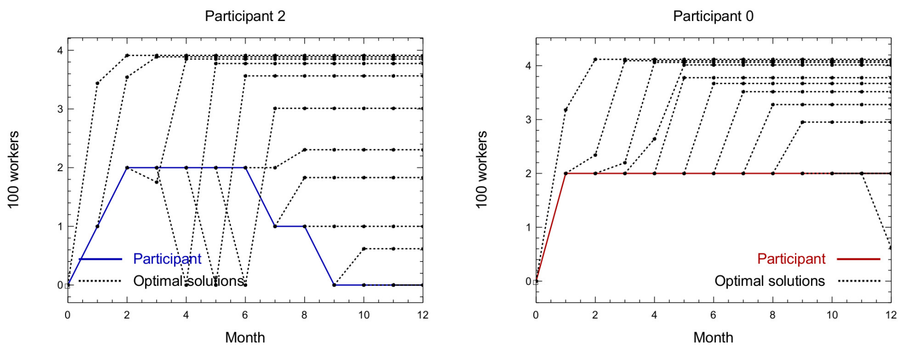


Figure 1. Top row: state variable $x_k^{W_{100}}$ that indicates how many workers for the 100 machines are employed. The left and right column show the results for two different participants. For both the optimal strategy is to have a fixed number of 0 to 4 workers which is decreasing as n_s increases. Note that the values are solutions of the relaxed problem where also non-integer values are possible.

initiative, which are also available under a public license. In this study we use the global solver *Couenne* [**3**] and the local solvers *Bonmin* [**4**] and *Ipopt* [**14**]. We used the currently latest stable version 0.2.2 of *Couenne*, and for better comparability the versions 1.1.1 of *Bonmin* and 3.6.1 of *Ipopt* it is interfaced with. For all solvers we used the default settings exclusively and the MA27 sparse solver for numerical linear algebra.

It turns out, however, that the size and complexity of the problems presented in this paper leads to extremely long runtimes of the global solver and can only be used on a small subset of the problems. We present a problem-specific cut to avoid bad local minima and guarantee monotonicity of the analysis function that builds on the locally optimal objective function values.

3.2. Optimal Solutions

In total, 2088 optimization problems have been solved. Depending on the value of n_s in (44.1), each consists of $13(N - n_s)$ control, $16(N - n_s)$ state, and $5(N - n_s)$ slack values. The total number of optimized variables for all 174 participants sums up to

$$n_{\text{var}} = 174 \sum_{n_s=0}^{N-1} 34(N - n_s) = 174 \cdot 2652 = 461448.$$

This many variables are obviously difficult to discuss and visualize comprehensively. As an illustration, in Figure 1 the state variable $x_k^{W_{100}}$ is depicted. It indicates how many workers for the 100 machines are employed at time k.

3.3. Local minima and integer solutions

The optimization problems (44.1) are nonconvex. Depending on initial values for the optimization variables different local minima can be found. Hence one has to use a global optimization solver, such as *Couenne* or one of the solvers listed on [**5**]. As mentioned above, we used three different solvers to obtain solutions. Table 1

S	0	1	2	3	4	5	6	7	8	9	10	11
1	0.15	0.13	0.17	0.11	0.1	0.06	0.05	0.03	0.02	0.02	0.0	0.0
2	1183	264	1552	1464	356	36	5	4	16	3	0.2	0.2

Table 1. CPU times in seconds for the solution of (44.1) for one participant. The columns show the start month n_s. Solver S 1: *Ipopt* for the relaxation of (44.1). Solver S 2: *Bonmin.* The global solver *Couenne* could only solve the problem for $n_\mathrm{s} = 11$ in 3 seconds, for $n_\mathrm{s} = 10$ the B&B tree grew too fast.

shows an overview of computational times that have been obtained with *Ipopt* and *Bonmin.* Note that the runtime is not monotonically increasing as n_s is reduced. The reason is that the solution process strongly depends upon the local minima of the relaxations that need to be solved.

The global solver *Couenne* was able to solve (44.1) for $n_\mathrm{s} = 11$ in 3 seconds. For the next larger problem, $n_\mathrm{s} = 10$, however, no results could be obtained. The solver terminated after processing 600.000 nodes in 7 hours, because the computer ran out of memory. The stack comprised about 2.000.000 open nodes at that time. To reduce the search space, we introduced and tightened the bounds on all variables to extremal values found with the local approaches. However, even with this restriction and a relaxation of all integer variables the same happened, now after 8.800.000 processed nodes with 2.9 million NLPs still on the tree. The best solution at that time was 500497 with the upper bound of 506610 still leaving a certain gap. For comparison: the objective function values found by *Bonmin* and *Ipopt* are 490385 and 500779, respectively. When heuristic non-convexity options `num_resolve_at_root` and `num_resolve_at_node` are used with a value of 1 (or 2) for *Bonmin*, an integer solution with value 500188 (500438) is found after 142 (317) seconds, which is considerably higher than the 0.2 seconds with the standard settings.

Obviously already for one participant data set the computational times are prohibitive for global approaches. For the analysis of all 174 participants we therefore solved 2088 NLP relaxations with the local optimizer *Ipopt.*

A crucial feature of our method is that the *How much is still possible*–function, see Section 4.1, decreases monotonically with n_s increasing. To take this into account, we exploit this knowledge in our a posteriori analysis. We define

$$(x^*, u^*, s^*) = (x^*_{n_\mathrm{s}}, \ldots, x^*_N, u^*_{n_\mathrm{s}}, \ldots, u^*_{N-1}, s^*_{n_\mathrm{s}}, \ldots, s^*_{N-1})$$

as a locally optimal solution obtained by solving problem (44.1) for month n_s.

We initialize the variables for problem (44.1) with a feasible solution. To avoid local minima with a worse performance, we add the additional cut

$$x_N^{OB} \quad \geq \quad x_N^{*,OB} \tag{44.2}$$

to (44.1).

Computational experience shows that the primal-dual interior point solver we are using cannot exploit the initialization to its full extent and in many cases *Ipopt* converged to locally infeasible points although it started from a primally feasible one. Future studies should therefore include active set based solvers. For this study we iterated in an inner loop with random initializations until the objective function cut (44.2) was fulfilled for all problems.

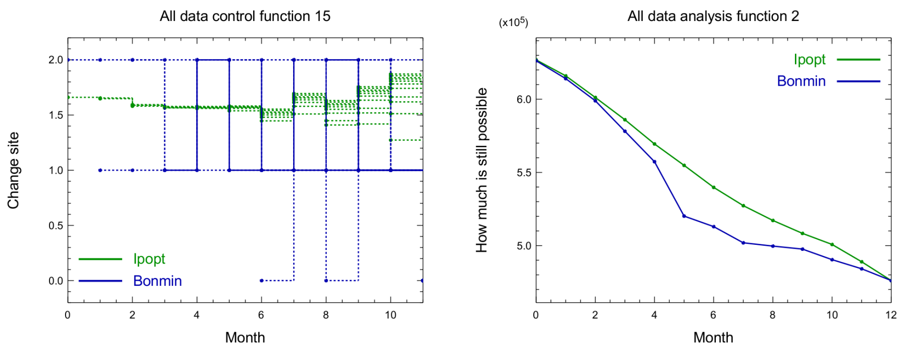


Figure 2. Left: optimal choices of site for one participant and all start months n_s, calculated with *Ipopt* (green, relaxed values between 1.1 and 1.9) and *Bonmin* (blue, integer values of 0, 1, and 2). Right: *How much is still possible*–function for one participant, calculated with *Ipopt* (green, upper curve) and *Bonmin* (blue, lower curve). The integer gap seems to be largest for intermediate values of n_s.

Within our analysis approach, local minima can lead to a violation of the goal to have an objective measurement for participant performance. Whenever possible, global solvers with a guaranteed, deterministic global minimum should be used. If the size of the problem is still too large for current algorithms and computational platforms, we propose to use relaxations and include the cut (44.2) as a compromise. The difference between participant's performance and global optimum seems to be so far apart compared to the distance between global and local minimum, especially when the cuts (44.2) are used, that the analysis based on a local *How much is still possible*–function should still be valid.

Several of the control variables are restricted to integer values. A comparison of (locally) optimal relaxed and integer solutions shows that some of the variables show typical behavior for most $x^p_{n_s}$, such as the maintenance u_k^{MA} or the purchase of raw material $u_k^{\Delta MS}$. Others, in particular the numbers of machines and workers, the shirt price u_k^{SP}, and the choice of the site u_k^{CS} are more sensitive to local optima and/or the fixation of some of the variables to integer values. Figure 2 shows an example.

3.4. Analyzing Lagrange Multipliers

Using optimization as an analysis tool yields insight on several levels. Structural properties of the problem, e.g., the unboundedness, can be understood. Also the performance of a participant can be compared to the optimal solution, and the *How much is still possible*–function to be discussed in Section 4 delivers a temporal resolution of this performance. But even a more detailed analysis is possible. While an analysis of the *How much is still possible*–function indicates at what rounds the participant made particularly good or bad decisions, the question of what of the decisions contributed significantly to the success or failure remains and might be of importance in a given test scenario.

We propose to combine two concepts. First, the comparison of the participant's decisions at month n_s with the optimal solution, $u^p_{n_s} - u^*_{n_s}$, gives a global indication of differences in the controls. However, it is unclear from this comparison how significant a single deviation is. Therefore we use, second, Lagrange Multipliers for the participant's decisions to measure the effect on the objective function. We augment problem (44.1) with the additional constraint

$$u_{n_\mathrm{s}} = u^\mathrm{p}_{n_\mathrm{s}} \tag{44.3}$$

Note that necessarily it holds $x^*_{n_\mathrm{s}+1} = x^\mathrm{p}_{n_\mathrm{s}+1}$, hence the augmented problem for month n_s has the same solution as problem (44.1) for $n_\mathrm{s}+1$. Hence there is no need for additional optimization problems to be solved. The advantage is that an optimization code will also calculate the dual variables or *Lagrange multipliers* λ_{n_s} for the constraints (44.3). It is well known that the Lagrange multipliers indicate the shadow prices, i.e., how much the objective function will vary if the corresponding constraints were relaxed, assumed that the active set stays constant.

4. A correct indicator function for Tailorshop

We propose to use the solutions of (44.1) for all n_s as an indicator function for the performance of a participant. The approach described in Section 4.1 is generic and should also be used for other test scenarios in complex problem solving in the future. In Section 4.2 we describe the results we obtained by using this indicator function for a psychological study.

4.1. How much is still possible

On an individual basis, the performance of every participant can be better understood by a comparison with optimal solutions as illustrated in Section 3. For an evaluation of large data sets that shall be related to characteristics of participants or experimental setup, an automatization and a reduction to an indicator function is necessary. To measure performance within the *Tailorshop* scenario different indicator functions have been proposed in the literature, e.g., the evolution of profit or overall worth of the tailorshop. An obvious drawback of comparing the results of several rounds with one another is that the main goal of the participant is to maximize the value at the end of the test, not necessarily in between.

Hence it might happen that decisions are analyzed to be bad, while they are actually good ones and vice versa. To overcome this problem, we propose to compare the decisions to mathematically optimal solutions. In a certain analogy to the cost-to-go-function in dynamic programming, the optimal objective function values for *all* rounds yield the monotonically decreasing *How much is still possible*–function. We look at the series of optimal objective function values $F^*(x_N; n_\mathrm{s})$ for $n_\mathrm{s} = 0, \ldots, N-1$. By comparing $F^*(x_N; n_\mathrm{s} = k)$ with $F^*(x_N; n_\mathrm{s} = k+1)$ we obtain the exact value of how much less the participant is still able to obtain, assumed he would take the best solutions from now on.

We conclude that the newly proposed methodology based on the *How much is still possible*–function is more reliable and generally applicable to test scenarios in complex problem solving.

4.2. Impact of Emotion Regulation

In the study 174 data sets have been used, every one from a different participant who had but one try. For 42 of them a *positive feedback* was used in the sense that in every round, regardless of the decisions the participant took, a sum of

20.000 money units (MU) was added to the capital. For 42 participants a *negative feedback* in form of a reduction of 8000 MUs was implemented. These modifications are implemented in the model and readjusted in the a posteriori analysis, of course.

In a previous study [**1**] it was shown that participants who receive a negative feedback perform better than those who receive positive feedback. In our new study we additionally considered the ability to regulate emotion. The psychological results of this study are submitted in a separate paper [**2**] in which also details on the experimental setup can be found. As a main result, an interaction between feedback and emotion regulation could be shown: participants with a high ability of emotion regulation perform better when they get negative feedback, while those with a low ability to regulate their emotions perform bad for negative and good for positive feedback.

Acknowledgments

Financial support of the Heidelberg Graduate School of Mathematical and Computational Methods for the Sciences is gratefully acknowledged.

References

1. C.M. Barth and J. Funke. Negative affective environments improve complex solving performance. *Cognition and Emotion*, 2009. (in press).
2. C.M. Barth, J. Funke, and S. Sager. Effects of emotion regulation and affect on problem solving. *Journal of Individual Differences*. (submitted).
3. P. Belotti. Couenne: a user's manual. Technical report, Lehigh University, 2009.
4. P. Bonami, L.T. Biegler, A.R. Conn, G. Cornuéjols, I.E. Grossmann, C.D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, and A. Wächter. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5(2):186–204, 2009.
5. M.R. Bussieck. Gams performance world. http://www.gamsworld.org/performance.
6. D. Dörner. On the difficulties people have in dealing with complexity. *Simulation and Games*, 11:87–106, 1980.
7. J. Funke. Einige Bemerkungen zu Problemen der Problemlöseforschung oder: Ist Testintelligenz doch ein Prädiktor? *Diagnostica*, 29:283–302, 1983.
8. J. Funke. *Problemlösendes Denken*. Kohlhammer, 2003.
9. J. Funke. Complex problem solving: A case for complex cognition? *Cognitive Processing*, 2010. (in press).
10. S. Kolb, F. Petzing, and S. Stumpf. Komplexes Problemlösen: Bestimmung der Problemlösegüte von probanden mittels verfahren des operations research – ein interdisziplinärer Ansatz. *Sprache & Kognition*, 11:115–128, 1992.
11. W. Putz-Osterloh. Über die Beziehung zwischen Testintelligenz und Problemlöseerfolg. *Zeitschrift für Psychologie*, 189:79–100, 1981.
12. S. Sager, H. Diedam, and M. Engelhart. Tailorshop: Optimization Based Analysis and data Generation tOol. TOBAGO web site https://sourceforge.net/projects/tobago.
13. H.-M. Süß, K. Oberauer, and M. Kersting. Intellektuelle Fähigkeiten und die Steuerung komplexer Systeme. *Sprache & Kognition*, 12:83–97, 1993.

14. A. Wächter and L.T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.
15. D. Wenke and P. A. Frensch. *Is success or failure at solving complex problems related to intellectual ability?*, pages 87–126. The psychology of problem solving. Cambridge University Press, 2003.

45

MINLP model and Lagrangian heuristic for the newsvendor problem with supplier discounts

Guoqing Zhang

University of Windsor
Department of Industrial and Manufacturing Systems Engineering
Windsor, Ontario, Canada N9B 3P4

gzhang@uwindsor.ca

Abstract

We study several multi-product newsboy problems with supplier quantity discounts, including single- and multi-constraints. Different from most previous nonlinear optimization models on the topic, we illustrate that those problems are formulated as mixed integer nonlinear programming models due to price discounts. Lagrangian relaxation approaches are presented to solve the problems. Computational results on large-scale test instances indicate that the proposed algorithms are extremely effective for the problems. A comparison with GAMS/CONOPT is also reported.
Keywords: newsboy, Lagrangian heuristic, quantity discount.

1. Introduction and modeling

The newsboy problem, also known as newsvendor problem or single period problem, has a rich history. Significant number of articles has been published to address the variants or extensions of the classical newsboy problem due to its importance to both inventory theory and practice ([1]-[4]). It has been noted that this topic has attracted great attentions in recent ten years. Both a budget constraint and suppliers price discounts are very common in supplier-retailer practice, but to our knowledge, the newsboy model with taking into account both features has not been reported in the literature. Typically the optimal order quantity depends on the purchasing price (and other parameters), which is fixed in previous research. However, a more realistic situation is that the price is also dependent on the order quantity.

The purpose of this study is to investigate the affect of both a budget constraint and supplier quantity discounts on the optimal order quantities in a multi-product newsboy problem. We present a mixed integer nonlinear programming model to formulate the problem and developed a Lagrangian relaxation approach.

Indices:

$i =$	$1,\dots, n$: index of products, where n is the total number of products
$k_i =$	the number of quantity discounts for product i offered by a supplier
$j =$	$1,\dots, k_i$: index of quantity segment j for product i offered by a supplier

Parameters:

p_i	=	unit sales revenue of product i
h_{im}	=	the resource consumed per unit of product i on the resource m
H_m	=	the limitation of the newsboy on resource m
c_{ij}	=	the unit discounted price of product i on discount segment j, and c_{i1} is original unit price
d_{ij}^L	=	the lower bound of the quantity of product i on discount segment j
d_{ij}^U	=	the upper bound of the quantity of product i on discount segment j
z_i	=	the random variable of the demand for product i
$f(z_i)$	=	the probability density function followed by the demand of product i
g_i	=	the estimated understocking cost of one unit of product i
s_i	=	the estimated overstocking cost of one unit of product i

We define the following decision variables:

Q_i	:	the amount of product i purchased from suppliers
Q_{ij}	:	the amount of product i purchased on discount segment j
y_{ij}	:	1 if product i is purchased on discount segment j ; otherwise 0

Thus, a multi-product newsboy problem with supplier quantity discounts and M constraints is formulated as follows:

$$MaxR = \sum_{i=1}^{n}\{\int_0^{Q_i}[p_i z_i - s_i(Q_i - z_i)]f(z_i)dz_i + \int_{Q_i}^{\infty}[p_i Q_i - g_i(z_i - Q_i)]f(z_i)dz_i\} - \sum_{i=1}^{n}\sum_{j=1}^{k_i} c_{ij}Q_{ij} \tag{45.1}$$

The first term represents the expected revenue minus the overstocking cost when the order quantities are above the actual demand levels (or plus the salvage when si is negative). The second term is the expected revenue minus the shortage cost (including loss of goodwill if any) when the order quantities are lower than the actual market demands. The third term is the cost for purchasing.

The problem is subject to the following constraints:

- Resource constraints:

$$\sum_{i=1}^{n} h_{im}Q_i \leqslant H_m, \quad m = 1,\dots M \tag{45.2}$$

- Quantity discount constraints: The suppliers provide an all-unit quantity discount scheme.

$$Q_{ij} \leqslant d^U_{ijl} y_{ij}, \quad \forall i, j, \tag{45.3}$$

$$Q_{ij} \geqslant d^L_{ij} y_{ij}, \quad \forall i, j, \tag{45.4}$$

$$\sum_{j=1}^{k_i} y_{ij} = 1, \quad \forall i, \tag{45.5}$$

$$Q_i = \sum_{j=1}^{k_i} Q_{ij}, \quad \forall i. \tag{45.6}$$

- Other nonnegative and integral constraints:

$$Q_{ij}, Q_i \geqslant 0, \quad \forall i, j; \tag{45.7}$$

$$y_{ij} \in \{0, 1\}, \quad \forall i, j. \tag{45.8}$$

Note that the main difference between our model and the existing models is the proposed model is a mixed integer nonlinear programming while the previous constrained newsvendor models are nonlinear programming.

2. Solution approach

We present a Lagrangian relaxation heuristic to solve the problem, mainly for large scale instances.

2.1. Reformulation and upper bound computation

We relax the resource constraints (1.2) and introduce associated Lagrange multipliers λ_m ($\lambda_m \geqslant 0$), $m = 1, , M$, to construct the following Lagrangian dual problem:

$$\mathrm{M}in_{\lambda_m} \mathrm{Max}\ L = R + \sum_{m=1}^{M} \lambda_m (H_m - \sum_{i=1}^{n} h_{im} Q_i) \tag{45.9}$$

$$\text{Subject to} \quad constraint\ \ (1.3)\ \ to\ \ (1.8) \tag{45.10}$$

The Lagrangian relaxation problem for a given set of values of λ_m, $m = 1, , M$, is:

$$\mathrm{M}ax\ L = R + \sum_{m=1}^{M} \lambda_m (H_m - \sum_{i=1}^{n} h_{im} Q_i) \tag{45.11}$$

$$\text{Subject to} \quad constraint\ \ (1.3)\ \ to\ \ (1.8).$$

The relaxation problem can be decomposed into n sub-problems, SP_i, as follows, each for product i:

$$\begin{aligned}\text{Max } R_i = {} & \int_0^{Q_i} [p_i z_i - s_i(Q_i - z_i)] f(z_i) dz_i + \int_{Q_i}^{\infty} [p_i Q_i - g_i(z_i - Q_i)] f(z_i) dz_i \\ & - \sum_{j=1}^{k_i} c_{ij} Q_{ij} - \sum_{m=1}^{M} \lambda_m h_{im} Q_i \end{aligned} \tag{45.12}$$

Subject to $\quad constraint\ \ (1.3)\ \ to\ \ (1.8).$

Considering constraint (1.5), the sub-problem can be further decomposed into k_i sub-problems, SP_{ij}, each for discount segment j:

$$\begin{aligned}\text{Max } R_{ij} \quad &= \int_0^{Q_{ij}} [p_i z_i - s_i(Q_{ij} - z_i)] f(z_i) dz_i + \int_{Q_{ij}}^{\infty} [p_i Q_{ij} - g_i(z_i - Q_{ij})] f(z_i) dz_i \\ (45.13) \qquad & - \sum_{j=1}^{k_i} c_{ij} Q_{ij} - \sum_{m=1}^{M} \lambda_m h_{im} Q_{ij} \\ \text{Subject to} \quad & \textit{constraint } (1.3) \textit{ to } (1.4).\end{aligned}$$

The optimal order quantity for each price segment without considering bound constraints is

$$(45.14) \qquad Q^*_{ij} \quad = \quad F^{-1}\left(\frac{p_i + g_i - c_{ij} - \sum_{m=1}^{M} \lambda_m h_{im}}{p_i + g_i + s_i}\right).$$

Proposition 1: Let Q^+_{ij} and Q^*_{ij} be the optimal solutions of problem (SP_{ij}) with and without bound constraints, respectively. Q^*_{ij} is evaluated as formula (2.6). Then we have: (i) If the solution Q^*_{ij} satisfies the bound constraints, i.e., , then $Q^+_{ij} = Q^*_{ij}$. (ii) If $Q^*_{ij} \geqslant d^U_{ij}$, then $Q^+_{ij} = d^U_{ij}$. (iii) If $Q^*_{ij} \leqslant d^L_{ij}$, then $Q^+_{ij} = d^L_{ij}$.

Proposition 1 provides a closed form to solve the optimization problem (SP_{ij}). According to the proposition, we design an algorithm to solve the sub-problem SP_i.

2.2. Lower bound computation and subgradient method

We present two heuristic approaches to construct an initial feasible solution and an updated feasible solution at last iteration respectively, and those feasible solutions are the lower bounds of the optimal solution. We use subgradient optimization method to update the the Lagrangian multipliers.

The following heuristic gives the steps to constructing a feasible solution from the final solution of Lagrangian dual approach, if it is infeasible.

```
Step 1 Compute the MENBUC for each product $i$:
    $G_i = [p_i - g_i - (p_i + s_i + g_i)F_i(Q_i) - c_i(Q_i)]/h_i$
Step 2 Sort the MENBUCs in an increasing order
Step 3 Choose the product according to the order, and reduce the order level as the follows:
    $Q_i = max\{Q_i + (H - \sum_{l=1}^{n} h_i Q_i)/h_i, d^L_{ij}\}$
    until the budget limit is satisfied. Where $d^L_{ij}$ is the lower bound of the discount segment $j$ that $Q_i$ locates.
Step 4. Update the solution
```

3. Numerical results

Some test instances are randomly produced with the number of products from 5 to 2,000, and the number of discount segments from 2 to 5. We assume that the demand of each product follows normal distribution and set the mean from 30 to 100 and the standard deviation from 5 to 15.

We first compare our results against that from GAMS/CONOPT, one effective solver for MINLP, for some problems with the number of discount segments $k = 3$ and single constraint. The comparisons are reported in Table 1. For the small size problems with the number of products from 5 to 20, GAMS/CONOPT got the optimal solutions but took much longer time than the proposed Lagrangian relaxation method (44.7 to 0.084 seconds for the average of computational time). For the instances with more than 200 products, GAMS/CONPOT could not obtain the optimal solutions in 30 minutes, while the proposed Lagrangian relaxation method can find either optimum or extremely good approximate solution in less than one minute.

We test the Lagrangian relaxation method for multi-constraint case with M = 2 or 3. The test instances for multiple constraints are randomly generated. The preliminary computational results are reported in Table 2. As seen in Table 2, the relative gap is 0.641% in the worst case and 0.138% on average.

Table 1. The solution and running time comparisons between Lagrangian method and GAMS/CONOPT

	GAMS/CONOPT		Lagrangian method			
N	Solution	CPU (sec.)	Bound	Solution	GAP (%)	CPU (sec.)
5	285787.73	18	285787.73	285787.73	0	0.0172
10	626421.66	24	626421.64	626421.64	0	0.109
20	1010006.21	92	1010005.99	1010005.99	0	0.125
200	?	30 min	12177026.06	12177026.06	0	0.791
2000	?	30 min	123895894.48	123894835.01	8.55E-04	55.078

Table 2. Test problem size, solution, gap, and running time for multi-constraint case

N	M	Bound	Solution	Gap(%)	CPU Time (sec.)
10	2	416448.57	413779.08	0.641	0.344
200	2	10950840.91	10948128.48	0.025	9.24
2000	2	135456831.45	135448777.31	0.006	127.23
10	3	521352.85	520923.00	0.082	0.25
200	3	12120945.94	12114293.77	0.055	3.812
2000	3	116029100.00	116009900.00	0.017	213.25

4. Extensions

A future work is to combine the demand management to the newsvendor model, for example, considering both supplier discounts and pricing decision in a multi-product newsboy model with budget and other constraints.

Acknowledgments

Supported by NSERC and CFI.

References

1. Abdel-Malek, L., Areeratchakul, N., A quadratic programming approach to the multi-product newsvendor problem with side constraints. European Journal of Operational Research 2007;176; 1607-1619
2. Khouja M, The newsboy problem with multiple discounts offered by suppliers and retailers. Decision Sciences 1996;27; 589-599

3. Lau HS, Lau AHL, The newsstand problem: A capacitated multiple-product single-period inventory problem. European Journal of Operational Research 1996; 94; 29-42
4. Zhang, G.Q., Ma, L.P., Optimal acquisition policy with quantity discounts and uncertain demands, International Journal of Production Research 2009; 47(9); 2409-2425

Claire Adjiman	Imperial College London	c.adjiman@imperial.ac.uk	3
Hesham Alfares	King Fahd Univ. of Petroleum	alfares@kfupm.edu.sa	
Kurt Anstreicher	University of Iowa	kurt-anstreicher@uiowa.edu	5
Pietro Belotti	Lehigh University	belotti@lehigh.edu	13,159
Daniel Bienstock	Columbia University	dano@columbia.edu	113
Christian Bliek	IBM France	bliek@fr.ibm.com	
Pierre Bonami	LIF-CNRS	pierre.bonami@lif.univ-mrs.fr	
Sonia Cafieri	Ecole Nationale de l'Aviation Civile	sonia.cafieri@enac.fr	205
Myun-Seok Cheon	ExxonMobil Research & Engineering	myun-seok.cheon@exxonmobil.com	
Alberto Costa	LIX, Ecole Polytechnique	costa@lix.polytechnique.fr	219
Zsolt Csizmadia	FICO	ZsoltCsizmadia@gmail.com	
Claudia D'Ambrosio	DEIS, University of Bologna	c.dambrosio@unibo.it	115
Gravot David	Rostudel Operations Research	dgravot@noos.fr	
Ali Diabat	MASDAR Inst. for Science and Tech.	adiabat@masdar.ac.ae	
Holger Diedam	IWR Heidelberg	hdiedam@ix.urz.uni-heidelberg.de	261
Laureano Escudero	Univdersidad Rey Juan Carlos	laureano.escudero@urjc.es	173
Oliver Exler	University of Bayreuth	oliver.exler@uni-bayreuth.de	119,237
Bernard Fortz	Université Libre de Bruxelles	bernard.fortz@ulb.ac.be	131
Ulf Friedrich	University of Trier	friedrich@uni-trier.de	
Claudio Gentile	IASI-CNR	gentile@iasi.cnr.it	139
Ambros Gleixner	Zuse Institute Berlin	gleixner@zib.de	103
Ignacio Grossmann	Carnegie Mellon University	grossmann@cmu.edu	67,245
M. Guignard-Spielberg	University of Pennsylvania	guignard_monique@yahoo.fr	149
Susanne Heipcke	FICO	SusanneHeipcke@fico.com	
Hassan Hijazi	Orange Labs & LIF	hassan.hijazi@yahoo.com	
Hossein T. Kakhki	Ferdowsi University	taghizad@math.um.ac.ir	231
Michal Kocvara	University of Birmingham	kocvara@maths.bham.ac.uk	31
Amélie Lambert	CEDRIC-ENSIIE	lambert.amelie@gmail.com	197
Jean B. Lasserre	LAAS-CNRS	lasserre@laas.fr	33
Jon Lee	IBM TJ Watson Research Center	jonlee@us.ibm.com	43
Stephan Lemkens	RWTH Aachen University	lemkens@math2.rwth-aachen.de	
Sven Leyffer	Argonne National Laboratory	leyffer@mcs.anl.gov	45
Leo Liberti	LIX, Ecole Polytechnique	leoliberti@gmail.com	205,219
Jeff Linderoth	University of Wisconsin-Madison	linderoth@wisc.edu	27
Peter Lindroth	Chalmers University of Technology	peter.lindroth@chalmers.se	
Andrea Lodi	DEIS, University of Bologna	andrea.lodi@unibo.it	25
Nelson Maculan	Federal University of Rio de Janeiro	maculan@cos.ufrj.br	
Ashutosh Mahajan	Argonne National Lab	mahajan@mcs.anl.gov	243
Jérôme Malick	CNRS/LJK/INRIA	jerome.malick@inria.fr	61
F. J. Martin-Campo	Rey Juan Carlos University	javier.martin.campo@urjc.es	95
Frédéric Messine	ENSEEIHT-IRIT	messine@n7.fr	65
Philippe Michelon	Université d'Avignon	philippe.michelon@univ-avignon.fr	251
Andrew Miller	Université Bordeaux 1	andrew.miller@math.u-bordeaux1.fr	13
Todd Munson	Argonne National Lab	tmunson@mcs.anl.gov	243
Giacomo Nannicini	Tepper School of Business, CMU	nannicin@andrew.cmu.edu	159
Adam Ouorou	Orange labs	adam.ouorou@orange-ftgroup.com	
Hacne Ouzia	LIP6, Paris 6	h.ouzia@gmail.com	249
Veronica Piccialli	Universit di Roma Tor Vergata	piccialli@disp.uniroma2.it	57
Mustafa Pinar	Bilkent University	mustafap@bilkent.edu.tr	209
Thomas Pogiatzis	University of Cambridge	tp309@cam.ac.uk	
Michael Poss	Université Libre de Bruxelles	mposs@ulb.ac.be	131
Carlos D. Rodrigues	Université d'Avignon	carlos-diego.rodrigues@univ-avignon.fr	251
Sebastian Sager	IWR, Uni Heidelberg	sebastian.sager@iwr.uni-heidelberg.de	169,261
Annick Sartenaer	University of Namur (FUNDP)	annick.sartenaer@fundp.ac.be	
Renata Sotirov	Tilburg University	r.sotirov@uvt.nl	73
Bhagawan Subedi	Nepal College of Information Tech.	bhagawansubedi@gmail.com	
Mohit Tawarmalani	Purdue University	mtawarma@purdue.edu	77
Philippe Toint	University of Namur (FUNDP)	philippe.toint@fundp.ac.be	81
Edwin van Dam	Tilburg University	edwin.vandam@uvt.nl	
Stefan Vigerske	Humboldt University Berlin	stefan@math.hu-berlin.de	181
Andreas Wächter	IBM TJ Watson Research Center	andreasw@us.ibm.com	83
Robert Weismantel	Otto-von-Güricke Univ. Magdeburg	weismant@mail.math.uni-magdeburg.de	85
Tapio Westerlund	Åbo Akademi University	twesterl@abo.fi	89
Angelika Wiegele	Alpen-Adria-Universität Klagenfurt	angelika.wiegele@uni-klu.ac.at	19
Guoqing Zhang	University of Windsor	gzhang@uwindsor.ca	271
Yang Zhang	Lightweight Struct. Inst., TU München	zhang@llb.mw.tum.de	