Introduction to Global Optimization

Leo Liberti

LIX, École Polytechnique, Palaiseau F-91128, France

(liberti@lix.polytechnique.fr)

October 23, 2008

Abstract

Accurate modelling of real-world problems often requires nonconvex terms to be introduced in the model, either in the objective function or in the constraints. Nonconvex programming is one of the hardest fields of optimization, presenting many challenges in both practical and theoretical aspects. The presence of multiple local minima calls for the application of global optimization techniques. This paper is a mini-course about global optimization techniques in nonconvex programming; it deals with some theoretical aspects of nonlinear programming as well as with some of the current state-of-the-art algorithms in global optimization. The syllabus is as follows. Some examples of Nonlinear Programming Problems (NLPs). General description of two-phase algorithms. Local optimization of NLPs: derivation of KKT conditions. Short notes about stochastic global multistart algorithms with a concrete example (SobolOpt). In-depth study of a deterministic spatial Branch-and-Bound algorithm, and convex relaxation of an NLP. Latest advances in bilinear programming: the theory of reduction constraints.

Contents

1 Introduction

Optimization is a field of applied mathematics that deals with finding the extremal value of a function in a domain of definition, subject to various constraints on the variable values. Historically, optimization techniques first came about in conjunction with problems linked with the logistics of personnel and transportation management. Typically, the problems were modelled in terms of finding the minimum cost configuration subject to all constraints be satisfied, where both the cost and the constraints were linear functions of the decision variables. These kinds of problems are all in the class of Linear Programming (LP). The most celebrated algorithm for solving such problems is the Simplex Algorithm, proposed by Dantzig in the late 40s/early 50s [Dan63]; its worst-case complexity is exponential in the number of problem variables, but it performs extremely efficiently in most practical instances.One other technique for solving LP problems, the Ellipsoid Algorithm, exhibits polynomial complexity. Many free and commercial implementations of the Simplex Algorithm exist nowadays, which are capable of solving extremely large-scale instances with up to millions of variables and constraints.

In many cases the nature of the problem explicitly requires integer or binary decision variables. Imposing such constraints on the variables makes the problem much more difficult to solve. MILP (Mixed-Integer Linear Programming) problems are problems where some (or all) variables are constrained to take integer values, and where the objective function and constraints are linear functions of the variables. Notwithstanding the linearity of the functions in the problem, the problem itself is nonlinear, since the integrality constraints on the variables can be expressed as nonlinear functions. The techniques for solving MILPs are very different from those for solving LPs. Typically, the solution of an entire LP problem (obtained by relaxing the integrality constraints on the integer variables, and called the LP relaxation) is required at each step of an algorithm that solves a MILP. The two most common algorithms employed in

solving MILPs are the Cutting Plane algorithm and the Branch-and-Bound algorithm. Their worst-case complexity is exponential in the size of the instance.

The true discriminant between "easy" and "hard" optimization problems in mathematical programming is the convex/nonconvex issue. A problem is convex if it is a minimization of a convex function (or a maximization of a concave function) where the admissible points are in a convex set. The fundamental result in convex analysis says that a locally optimal solution of a convex problem is also globally optimal (see Theorem 3.6 below). This is practically very important: since algorithmic termination tests occur at each iteration in the algorithm, they must be computationally very efficient; thus, virtually all termination conditions of optimization algorithms are local conditions, i.e. they test whether the current solution is locally optimal with respect to a pre-defined neighbourhood. If the problem is convex this is enough to guarantee that the solution is globally optimal. Nonconvex problems, on the other hand, may have many different local optima, and choosing the best one is an extremely hard task. The field of applied mathematics that studies extremal locations of nonconvex functions subject to (possibly) nonconvex constraints is called Global Optimization. If some, or all the problem variables are constrained to take discrete values only, the problem is nonconvex even if the mathematical expressions in the problem are linear (this happens because the solution set is a discrete set of points, which is a nonconvex set). Such problems are often termed Combinatorial Optimization problem due to the combinatorial nature of a discrete decision variable.

In general, the global optimization problem is formulated in terms of finding the point x in a solution space set X (called the *feasible region*) where a certain function $f : X \to T$ (called the *objective function*), attains a minimum or a maximum. T is any ordered set (usually a subset of $\mathbb{R}$). The set X is usually a subset of $\mathbb{R}^n$ defined by constraints $g(x) \gtreqless 0$, where g is a set of m possibly nonlinear functions of x and $\gtreqless$ are relations in $\{\leq, =, \geq\}$. The extremal point x can then be written as $(x_1, \ldots, x_n)$, and the x_i's are sometimes called *decision variables*. It is often practically useful to express the variable bounds explicitly as $x^L \leq x \leq x^U$ $(x^L, x^U \in \mathbb{R}^n)$. Some of the variables may be constrained to only take integer values ($x_i \in \mathbb{Z}$ for all i in an index set $Z \subseteq \{1, \ldots n\}$). It is customary to write the global optimization Mixed-Integer Nonlinear Programming problem (MINLP) as follows:

$$\left.\begin{array}{lrcl} \min_x & f(x) & & \\ & g(x) & \gtreqless & b \\ & x^L & \leq x \leq & x^U \\ & x_i \in \mathbb{Z} & & \forall\, i \in Z. \end{array}\right\} \tag{1}$$

Notice that expressing the integrality constraints on the variables can be reduced to imposing an appropriate constraint in form of a function. E.g., x_1 only taking values 0,1 can be expressed by imposing the constraint $x_1(x_1 - 1) = 0$. The variable bounds are also called *variable ranges*. It is sometimes convenient to introduce the problem constraints in the form

$$a \leq g(x) \leq b$$

where $a, b \in \mathbb{R}^m$ are lower and upper limits on the constraint values. Equality constraints are expressed by setting $a = b$ and one-way inequality constraints are expressed by setting $a = -\infty$ or $b = +\infty$. In the rest of the paper, both forms of expressing the constraints will be employed.

1.1 Scope of this paper

This paper has a didactical purpose, and therefore includes many details which would normally be assumed as known. It is meant to give a substantial overview of the field of global optimization of nonconvex MINLPs. We focus our treatment on deterministic global optimization algorithms with an in-depth treatment of the spatial Branch-and-Bound algorithm (see Sections 2.2 and 5); however, we also give some insight to stochastic methods (see Sections 2.1, 4). Since most modern global optimization methods use local optimization as a tool, an overview of local optimization of NLPs is also given (see Section

3), which also contains much basic material about convexity and elementary multi-variate geometry and analysis. Although this is mainly a theoretical paper, some examples have been included (see Section 1.2). Section 6 covers more advanced material about an automatic way to provide very tight convex relaxations of bilinear problems.

1.2 Examples which require Global Optimization

In this section we shall see some examples of multi-extremal nonconvex programming problems. Solving these problems requires global optimization methods.

1.1 Example (Haverly's Pooling Problem)
Haverly's Pooling Problem, first introduced in [Hav78], is described visually in Fig. 1. We have three input feeds with di□erent levels of percentage of sulphur in the crude. Accordingly, the unit costs of the input feeds vary. Two of the feeds go into a mixing pool, which makes the level of sulphur percentage the same in the two streams coming out of the pool. The various streams are then blended to make two end products with di□erent requirements on the sulphur percentage and di□erent unit revenues. We have some market demands on the end products and we want to determine the quantities of input crude of each type to feed into the networks so that the operations costs are minimized. This problem can be

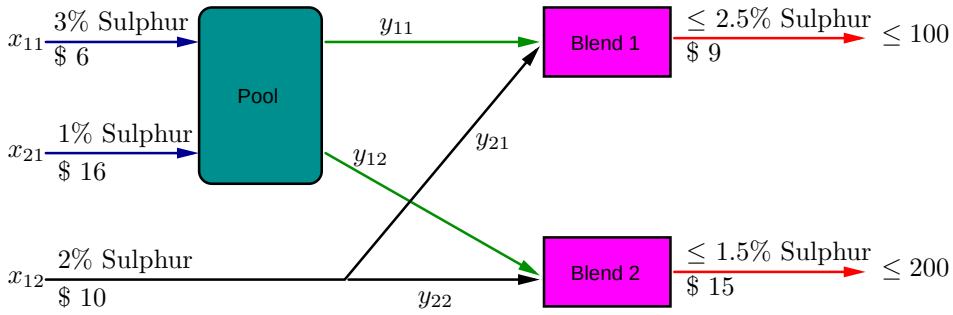


Figure 1: Haverly's Pooling Problem.

formulated in various ways. We present here the what is known as the p-formulation of the HPP.

$$
\begin{array}{lll}
\min\limits_{x,y,p} & 6x_{11} + 16x_{21} + 10x_{12} - 9(y_{11} + y_{21}) - 15(y_{12} + y_{22}) & \textbf{cost} \\
\textbf{s.t.} & x_{11} + x_{21} - y_{11} - y_{12} = 0 & \textbf{mass balance} \\
 & x_{12} - y_{21} - y_{22} = 0 & \textbf{mass balance} \\
 & y_{11} + y_{21} \leq 100 & \textbf{demands} \\
 & y_{12} + y_{22} \leq 200 & \textbf{demands} \\
 & 3x_{11} + x_{21} - p(y_{11} + y_{12}) = 0 & \textbf{sulphur balance} \\
 & py_{11} + 2y_{21} \leq 2.5(y_{11} + y_{21}) & \textbf{quality req.} \\
 & py_{12} + 2y_{22} \leq 1.5(y_{12} + y_{22}) & \textbf{quality req.}
\end{array}
$$

where x, y are the input and intermediate stream quantities (as in Fig. 1) and p is the percentage of sulphur of the streams out of the pool. Notice that this problem has three constraints involving bilinear terms, so global optimization techniques are mandatory for its solution.

1.2 Example (Pooling/Blending Problems)
The HPP (Example 1.1) is a special instance of a class of problems termed Pooling/Blending problems. Various types of crude oils of di□erent qualities, coming from di□erent feeds, are mixed together in various pools to produce several end-products subject to quality requirements and quantity demands; the objective is to minimize the costs. These problems are usually modelled as continuous NLPs involving bilinear terms. In general they may have many local optima, and so the use of global optimization

methods is required. The literature on Pooling/Blending problems is vast [ATS99, FHJ92, VF96, TSed, ABH+04, LP06]. We present the general blending problem formulation found in [ATS99], p. 1958. The formulation refers to a setting with q pools each with n_j input streams ($j \leq q$) and r end products. Each stream has l qualities.

$$\left.\begin{array}{rll} \min_{x,y,p} \sum_{j=1}^{q} \sum_{i=1}^{n_j} c_{ij} x_{ij} - \sum_{k=1}^{r} d_k \sum_{j=1}^{q} y_{jk} & & \\ \sum_{i=1}^{n_j} x_{ij} = \sum_{k=1}^{r} y_{jk} & \forall j \leq q & \\ \sum_{j=1}^{q} y_{jk} \leq S_k & \forall k \leq r & \\ \sum_{i=1}^{n_j} \lambda_{ijw} x_{ij} = p_{jw} \sum_{k=1}^{r} x_{jk} & \forall j \leq q \; \forall w \leq l & \\ \sum_{j=1}^{q} p_{jw} x_{jk} \leq z_{kw} \sum_{j=1}^{q} y_{jk} & \forall k \leq r \; \forall w \leq l & \\ x^L \leq x \leq x^U, p^L \leq p \leq p^U, y^L \leq y \leq y^U, & & \end{array}\right\} \quad (2)$$

where x_{ij} is the flow of input stream i into pool j, y_{jk} is the total flow from pool j to product k and p_{jw} is the w-th quality of pool j; $c_{ij}, d_k, S_k, Z_{kw}, \lambda_{ijw}$ are, respectively: the unit cost of the i-th stream into pool j, the unit price of product k, the demand for product k, the w-th quality requirement for product k and the w-th quality specification of the i-th stream into pool j.

Euclidean location problems can also be expressed as NLPs:

1.3 Example (Euclidean Location Problems)

There are n plants with geographical positions expressed in Euclidean coordinates (a_i, b_i) ($1 \leq i \leq n$) w.r.t. to an arbitrary point $(0, 0)$. Daily, the i-th plant needs r_i tons of raw material, which can be delivered from m storage points each with storage capacity c_j ($1 \leq j \leq m$). For security reasons, each storage point must be located at least at $D = 1Km$ distance from the plant. The cost of raw material transportation between each storage point and each plant is directly proportional to their distance as well as the quantity of raw material. Find geographical positions where to place each storage point so that the transportation costs are minimized. The mathematical formulation is as follows:

$$\left.\begin{array}{lll} \min\limits_{x,y,d,w} & \sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij} d_{ij} & \\ \textbf{s.t.} & \sum_{i=1}^{n} w_{ij} \leq c_j \quad \forall j \leq m & \textbf{storage capacity} \\ & \sum_{j=1}^{m} w_{ij} \geq r_i \quad \forall i \leq n & \textbf{plant requirement} \\ & d_{ij} = \sqrt{(x_j - a_i)^2 + (y_j - b_i)^2} \quad \forall i \leq m, j \leq n & \textbf{Euclidean distance} \\ & d_{ij} \geq D \quad \forall i \leq n, j \leq m & \textbf{minimum distance} \end{array}\right\}$$

where (x_j, y_j) is the geographical position of the j-th storage point, d_{ij} is the distance between the i-th plant and the j-th storage point, w_{ij} is the quantity of raw material transported from the j-th storage point to the i-th plant ($i \leq n$, $j \leq m$).

Lastly, we present an example which shows that global optimization techniques can be a research aid to questions in pure mathematics.

1.4 Example (Kissing Number Problem)

When rigid balls touch each other, in technical terms, they "kiss" (this is the etimology of the term "kissing number"). In mathematical terms, the *kissing number* in D dimensions is the number of D-spheres of radius R that can be arranged around a central D-sphere of radius R so that each of the surrounding spheres touches the central one without overlapping. Determining the maximum kissing number in various dimensions has become a well-known problem in Combinatorial Geometry. Notationally, we indicate the Kissing Number Problem in D dimensions by KNP(D). In $\mathbb{R}^2$ the result is trivial: the maximum kissing number is 6 (see Fig. 2). The situation is far from trivial in $\mathbb{R}^3$. The problem earned its fame because, according to Newton, the maximum kissing number in 3D is 12, whereas according to his contemporary fellow mathematician David Gregory, the maximum kissing number in 3D is 13 (this conjecture was stated without proof). This question was settled, at long last, more than 250 years after having been stated, when J. Leech finally proved that the solution in 3D is 12 [Lee56]. Given parameters D (number of dimensions) and N (number of spheres), the variables $x^i = (x_1^i, \ldots, x_D^i)$, $1 \leq i \leq N$ determine the

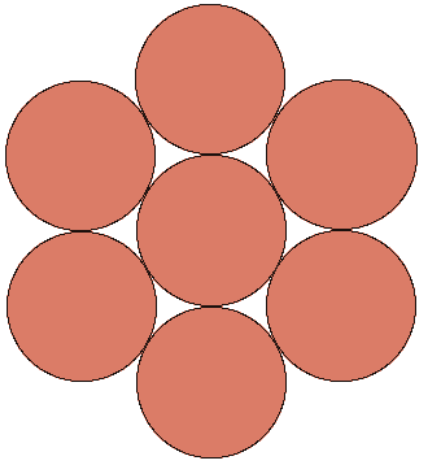


Figure 2: The Kissing Number Problem in 2D.

position of the center of the i-th sphere around the central one. We maximize a decision variable $\alpha \geq 0$ which represents the minimum pairwise sphere separation distance in the N-sphere configuration being tested, subject to the necessary geometric constraints. Since the constraints are nonconvex, there may be multiple local minima. If the global optimization of the model determines that the global maximum is at $\alpha \geq 1$, then there is enough space for N spheres; if the globally optimal α is strictly less than 1, it means that the N configuration has overlapping spheres, hence the kissing number is $N - 1$. By solving this decision problem repeatedly for di□erent values of N, we are able to quickly pinpoint the maximum N for which $\alpha > 1$. The following formulation correctly models the problem:

$$\begin{aligned} \max \quad & \alpha && (3)\\ \forall i \leq N \quad & ||x^i||_2 = 2R && (4)\\ \forall i < j \leq N \quad & ||x^i - x^j||_2 \geq 2R\alpha && (5)\\ & \alpha \geq 0 && (6)\\ \forall i \leq N \quad & x^i \in \mathbb{R}^D && (7) \end{aligned}$$

Constraints (4) ensure that the centers of the N spheres all have distance $2R$ from the center of the central sphere (i.e., the N spheres kiss the central sphere). Constraints (5) make the N spheres non-overlapping. This problem has been solved correctly up to $D = 4$ (and $N = 24$, which is KNP(4)) with the SobolOpt algorithm (see Sect. 4) [LMK04].

1.3 A brief history of global optimization

Generic optimization problems have been important throughout history in engineering applications. The first significant work in optimization was carried out by Lagrange in 1797 [Lag97]. Nevertheless, before the introduction and extensive usage of electronic computers, the requirement for global optimization was not even an issue given the tremendous amount of computational effort it requires. Global optimality of solutions was guaranteed only when locally optimizing convex functions, a rather limited class of problems.

The methods that were first used in global optimization were deterministic techniques, mostly based on the divide-and-conquer principle. This was introduced in the late 1950s with the advent of the first electronic computers into the research environment. In contrast with the computational principles employed before the introduction of computers, whereby people would try to minimize the amount of actual computations to perform with respect to the theoretical framework, the divide-and-conquer principle

applies to the intrinsic iterative nature of methods used when working with electronic computers: keep the complexity of the theoretical structures involved to a minimum, and rely on intensive computation to explore the solution space. Thus, instead of trying to locate a minimum by solving a set of equations by symbolic/algebraic methods, one would try to construct a sequence of approximate solutions which converges to the solution by dividing the problem into smaller subproblems. One typical algorithm which embodies the divide-and-conquer principle is the Branch-and-Bound algorithm (BB) [PS98]. Because of the nature of the algorithm, where the subproblems are produced by branching a problem entity (e.g. variable) into its possible instances, the BB algorithm applies very well to cases where problem entities are discrete in nature. Thus, the first applications of BB to global optimization problems were devoted to discrete problems such as the Travelling Salesman Problem (TSP).

The first paper concerning continuous global optimization with a BB (deterministic) technique dates from 1969 [FS69]. In the 1970s and 1980s, work on continuous or mixed-integer deterministic global optimization was scarce. Most of the papers published in this period dealt either with applications of global optimization to very specific cases, or with theoretical results concerning convergence proofs. One notable exception was the work of McCormick [McC76] who considered symbolic transformations of problems. His methods are such that they can, in theory, be carried out automatically by a computer.

A major reason for the slow pace of progress in continuous global optimization is that it is computationally very expensive, and it was not until the 1990s that computer hardware with the necessary power became available. Toward the end of the 1980s, however, the first elementary textbooks began to appear on the subject. They explained the basics of local optimization (Lagrange multipliers and Karush-Kuhn-Tucker conditions) and some of the early techniques in deterministic [PR87] and stochastic [Tu89] global optimization. Early topics in deterministic global optimization include convex optimization [Pin86], Branch-and-Bound techniques restricted to particular classes of problems (e.g. concave minimization problems [Hor86]) and some theoretical and complexity-related studies [Ž85, RR87]. Stochastic algorithms based on adaptive random search appeared between the 1970s and early 1980s [LJ73, MBS80]. It is worth noting that the first deterministic global optimization technique that was able to deal with generic nonconvex continuous NLPs was Interval Optimization [Han80, Han92]. Unfortunately, it often has slow convergence due to the fact that Interval Arithmetic generally provides very wide intervals for the objective function value.

Toward the end of the 1980s, many papers on deterministic global optimization started to appear in the literature. They were mostly concerned with iterative methods applied to particular classes of problems [Par88, MM88, Hor88, HDT88, TH88, KG88, MM89], but there were also theoretical studies [HJL88, Hor89] and parallel implementation studies and reports [Par89, Ge89, ES89]. In the area of stochastic optimization, some of the first algorithms employing "tunnelling" [Yao89] began to appear in the same period.

Since the beginning of the 1990s, the optimization research community has witnessed an explosion of papers, books, algorithms, software packages and resources concerning deterministic and stochastic global optimization. In the early 1990s, most of the articles were still concerned with applications of global optimization or algorithms which perform well on particular classes of problems. It is significant that one of the first and most widely used book in global optimization [HT96], first published in 1990 and successively re-published in 1993 and 1996, does not even mention nonconvex NLPs in general form (1).

The first method that was able to deal directly with the generic nonconvex NLPs in the form (1) was Ryoo and Sahinidis' Branch-and-Reduce algorithm which appeared in a paper published in May 1995 [RS95, RS96]. Shortly afterwards, Floudas' team published their first article on the αBB branch-and-bound method [AMF95] which was then thoroughly explored and analysed in several subsequent papers [AF96, AAMF96, AAF97, ADFN98, AAF98, ASF98]. The first αBB variant that addressed problems in the form (1) appeared in 1997 [AAF97]. One notable limitation of the αBB algorithm is that it relies on the functions being twice differentiable in the continuous variables.

Since the inception of the αBB algorithm, a number of Branch-and-Select algorithms geared toward

the most generic nonconvex MINLP formulation appeared in the literature, like Smith and Pantelides' symbolic reformulation approach [Smi96, SP97a, SP99], Pistikopoulos' Reduced Space Branch-and-Bound approach [EP97] (which only applies to continuous NLPs), Grossmann's Branch-and-Contract algorithm [ZG99] (which also only applies to continuous NLPs) and Barton's Branch-and-Cut framework [KB00]. Within the Branch-and-Select strategy, we can also include modern Interval Analysis based global optimization methods [VEH96, OBF01, uB03].

Branch-and-Select is by no means the only available technique for deterministic global optimization, but it seems to be the method which least relies on the problem having a particular structure: this is an advantage insofar as one needs to implement software which solves problems in the general form (1). For example, d.c. optimization and the Extended Cutting Planes (ECP) method [WSHP98, WP02] are structurally different from Branch-and-Select approaches; however, both make use of a problem which is already in a special form. It must be said, however, that "special forms" are sometimes general enough to accommodate large classes of problems. For instance, the ECP method solves mixed-integer NLPs (MINLPs) in the following form: $\min\{f(z) \mid g(z) \leq 0 \wedge Az \leq a \wedge Bz = b \wedge z \in \mathbb{R}^n \times \mathbb{Z}^m\}$, where f is a pseudo-convex function, g is a set of pseudo-convex functions, A, B are matrices and a, b are vectors. Although not all functions are pseudo-convex, the latter do form quite a large class. It is the pseudo-convexity condition that allows the ECP method to derive a new valid cutting plane at each iteration.

In the stochastic global optimization field, recent advances include Simulated and Nested Annealing [Raj00, Loc02], Tabu Search (TS) [CS00, KVvA$^+$99], Multi-Level Single Linkage (MLSL) [Sch02, KS04], Variable Neighbourhood Search (VNS) [MPKVv03], Differential Evolution [SP97b], Adaptive Lagrange-Multiplier Methods [WW99], Ant Colony Simulation [MKP$^+$00, CDM91], Quantum Dynamics in complex biological evolution [HH00], Quantum Thermal Annealing [LB00], Ruin and Recreate Principle [SSSWD00]. In particular, the application of the meta-heuristics simulated annealing, tabu search and variable neighbourhood search has yielded very good results.

2 The structure of Global Optimization Algorithms

Algorithmic methods for globally solving MINLPs are usualy divided into two main categories: deterministic and stochastic.

An algorithm is deterministic if it can be simulated by a *deterministic finite automaton*, that is, a 5-tuple $(\Sigma, Q, q_0, F, \delta)$ where Σ is the input alphabet, Q the set of states, $q_0 \in Q$ the start state, $F \subseteq Q$ the set of final states, and $\delta : Q \times \Sigma \to Q$ the transition function. The transition function is a device that given the current state of the automaton and a word from the input alphabet, produces the next state the automaton is going to be in. The fact that δ is a well-defined function embodies the principle of determinism. To any pair (current state, input word) there corresponds a unique next state. On the other hand, if δ is a relation between $Q \times \Sigma$ and Q, that is, $\delta(q, \sigma)$ is a subset of Q, then to any pair (current state, input word) there may correspond more than one possible next states. In this case the automaton is a *nondeterministic finite automaton*.

Since algorithms are instruction sequences that make a computer perform certain steps, and since computers are theoretically deterministic finite automata[1], there is no doubt that all algorithms are inherently deterministic. Moreover, it can be shown that any nondeterministic finite automaton can be simulated by a deterministic finite automaton. Thus, there can be algorithms running on deterministic computers which are based on transition relations, rather than functions. Notice that nondeterminism is not synonymic with random choices. If at any stage in the computation the next step is not unique we have nondeterminism; but since any enumerable set can be well-ordered (i.e. for any subset there is a unique minimum element) and computers are finite state machines, there is always a choice function that

[1]This statement does not apply to quantum computers; in practice, it is also debatable whether it can be applied to parallel computers in full generality.

allows the next step to be selected in a unique way. The convergence proofs for this type of algorithms do not involve probability theory.

Stochastic Global Optimization, on the other hand, deals with situations where there is a random element in the choice of the next step of computation. Deterministic automata like computers are very bad at performing random choices, hence these are simulated with pseudo-random number sequences or similar computational devices. When true random data are required, a random external input (like for example the decaying time of some radioactive atoms) is employed. The convergence proofs for this type of algorithms involve the use of probability theory.

In most global optimization algorithms (both deterministic and stochastic) it is possible to identify two separate phases (see [Sch02] for a review of two-phase global optimization methods, although in this paper we do not define the two-phase classification quite in the same way). The exhaustive exploration of the search space is delegated to the *global phase*, which may be deterministic or stochastic. At each iteration in the global phase, a local optimization procedure is called to identify a locally optimal point; this is known as the *local phase*, which is usually deterministic. The global phase calls the local optimization procedure as a "black-box": a reliable and robust local optimization procedure is therefore of paramount importance for a fast convergence, so very advanced local optimization algorithms are employed. Most of the actual number-crunching is performed in the local phase, which turns out to be the most computationally expensive step of the global optimization algorithm.

Local optimization of NLPs is an NP-hard problem in itself [PS88], so finding the global optimum of most nonconvex problems is also NP-hard. Although the local optimization phase is used as a "black-box" function call, a good knowledge of the local optimization technique used is important to fine-tune the global phase, so we shall give some attention to the theory of local optimization of NLPs in Sect. 3.

2.1 Stochastic global phase

Stochastic methods are based on an element of random choice. Because of this, one has to sacrifice the possibility of an absolute guarantee of success within a finite amount of computation. For a survey of these methods, see [Sch02, GBR95, PRT00]. See [Ree93] for a survey of stochastic techniques for combinatorial (discrete) problems. Roughly speaking, there are two approaches in designing a stochastic global phase: the sampling and the escaping approach.

In the sampling approach, followed for example by Multi-Level Single Linkage (MLSL) and Variable Neighbourhood Search (which is actually a mixture of both sampling and escaping approaches), the emphasis is on extensive sampling: many local optimization procedures are initiated from many different starting points selected with various rules (see Fig. 3); the best among these local optima is "labelled" the global optimum. Various different algorithm termination rules exist in the literature. One of the most common is the Bayesian stopping rule, which is based on the expected number of local minima in the sampling set. Exhaustive sampling set exploration or a more practical CPU time limit are also employed. Sampling stochastic algorithms are guaranteed to converge in infinite time with probability 1, but in practice there is no guarantee that the located solution will actually be the global optimum, or by how far the algorithm has failed to locate the true global optimum. These algorithms usually work very well for small and medium-sized problems, but the chance of finding the global optimum worsens considerably as the number of dimensions increases, due to the limitation on the sampling set size.

In the escaping approach (see e.g. Tabu Search, Simulated Annealing and tunnelling methods), various methods are devised for "escaping from local optima", so that after the local optimization procedure has terminated with a local optimum, the global phase is capable of reaching another feasible point from which to re-start a new local optimization procedure (see Fig. 4). Termination conditions for these methods are usually based on the number of local optimization calls or on a maximum time limit. As in the sampling approach, there are no definite guarantees on global optimality (apart from convergence in infinite time with probability 1).

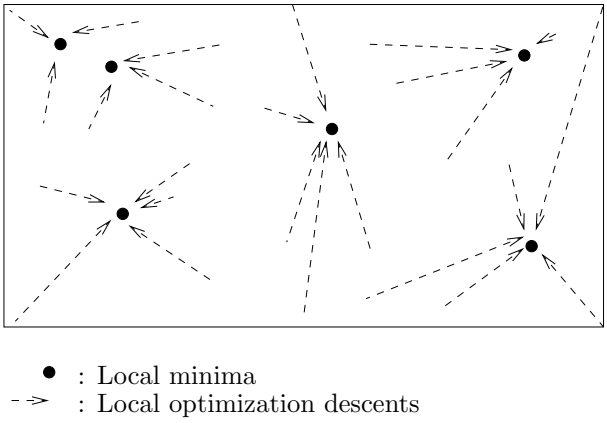


Figure 3: Sampling approach of the stochastic global phase. The arrows indicate a local optimization descent path from a starting point to the target local minimum.

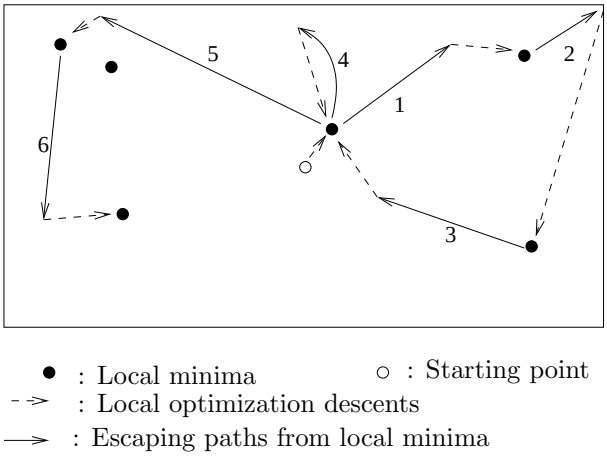
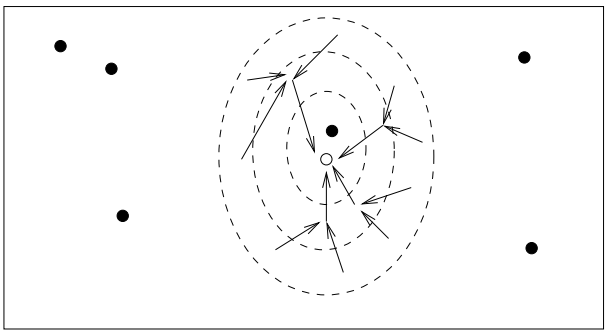


Figure 4: Escaping approach of the stochastic global phase. The trail consists of local optimization descents alternating with random escaping paths. Observe that certain escaping paths (e.g. arc 4) might position the new starting point in the same basin of attraction as the previous local optimum.

In both approaches, one cosiderable problem is given by multiple local procedure calls converging to the same local optimum. This is a waste of computational resources, as each local solution is very expensive to compute, and should ideally be found only once. The idea of *clustering* for the stochastic global phase suggests that the sampling of starting points should be grouped into clusters of nearby points, and only one local search from each cluster should be performed (see [Sch02], p. 159, and [KS04]). One particularly interesting idea for clustering is the linkage method: a point x is clustered together with a point y if x is not too far from y and the objective function value at y is better than that at x. The clusters are then represented by a directed tree, the root of which is the designated starting point from where to start the local optimization procedure (see Fig. 5).

Multi-Level Single Linkage can be described as a multi-start algorithm with clustering. One example of a good implementation of an MLSL algorithm is given in Section 4. A Variable Neighbourhood Search approach to global optimization can be found in [MPKVv03]. VNS escapes from the current local minimum by initiating other local searches from a sampling set which increases its size iteratively. As soon as a local search identifies a local minimum different from the current one, the new local minimum is marked as "current" and the algorithm progresses. In Tabu Search [KVvA$^+$99], successive candidate



● : Local minima
○ : Starting point selected in the cluster

Figure 5: Linkage clustering in the stochastic global phase. The points in the cluster are those incident to the arcs; each arc (x, y) expresses the relation "x is clustered to y". The arcs point in the direction of objective function descent. The root of the tree is the "best starting point" in the cluster.

points are located by performing moves in the search space. The inverse of the move is then put into a tabu list (which is a FIFO queue) and forbidden from being applied for as long as it stays in the queue. This way, if the current candidate point is at a local minimum and a move is applied which takes it out of the local minimum, the inverse move will not be applied at an immediately successive iteration. This makes it unlikely to fall back on the same local minimum we are escaping from. In Simulated Annealing, the search procedure is allowed to escape from local optima with a certain (decreasing) probability [Loc02]. Tunnelling methods derive from the quantum-mechanical idea of a particle in a potential well which escapes from the well with a certain probability [RSC00].

2.2 Deterministic global phase

This section contains a very general treatment of a class of algorithms (called Branch-and-Select) to which the spatial Branch-and-Bound algorithm (which will be analysed in detail later) belongs. Although, as has been remarked in Sect. 1.3, Branch-and-Select is not the only deterministic global phase approach to global optimization, it is nonetheless the most widely used.

Branch-and-Select algorithms include well-known techniques such as Branch-and-Cut and Branch-and-Bound, and are among the most effective methods for the deterministic solution of global optimization problems. They started out as divide-and-conquer type algorithms to solve combinatorial optimization problems like the Travelling Salesman Problem [AHU83] but were very soon applied to continuous and mixed-integer nonlinear optimization [FS69, HT96, PR87]. In this section, we present a general theory of such algorithms based on material from [Tuy98], together with the necessary convergence proofs.

Branch-and-Select algorithms can be used to solve the widest possible class of optimization problems (1) to global optimality, even when objective function and constraint values are provided by black-box procedures. In fact, they can be designed to rely on no particular feature of the problem structure. However, in their basic form and without any acceleration devices (such as pre-processing steps or improved algorithmic steps), they tend to be very inefficient. Furthermore, their performance may depend strongly on the formulation of the problem, meaning that a different algebraic form of the same equations might lead to quite different algorithmic performance.

Let $\Omega \subseteq \mathbb{R}^n$. A finite family of sets $\mathcal{S}$ is a *net* for Ω if it is pairwise disjoint and it covers Ω, that is, $\forall s, t \in \mathcal{S}$ $(s \cap t = \emptyset)$ and $\Omega \subseteq \bigcup_{s \in \mathcal{S}} s$. A net $\mathcal{S}'$ is a *refinement* of the net $\mathcal{S}$ if $\mathcal{S}'$ has been obtained from $\mathcal{S}$ by finitely partitioning some set s in $\mathcal{S}$ and then replacing s by its partition; that is, given $s \in \mathcal{S}$ with

$s = \bigcup_{i \in I} s_i$ where I is an index subset of $\{1, \ldots, |\mathcal{S}'|\}$ and each $s_i \in \mathcal{S}'$, we have $\mathcal{S}' = (\mathcal{S} \backslash s) \cup \{s_i \mid i \in I\}$.

Let $\langle \mathcal{S}_n \rangle$ be an infinite sequence of nets for Ω such that, for all $i \in \mathbb{N}$, $\mathcal{S}_i$ is a refinement of $\mathcal{S}_{i-1}$, and let $\langle M_n \rangle$ be an infinite sequence of subsets of Ω such that $M_i \in \mathcal{S}_i$. $\langle M_n \rangle$ is a *filter* for $\langle \mathcal{S}_n \rangle$ if $\forall i \in \mathbb{N}$ we have $(M_i \subseteq M_{i-1})$. Let $M_\infty = \bigcap_{i \in \mathbb{N}} M_i$ be the *limit* of the filter.

We will now present a general framework for Branch-and-Select algorithms that globally solve the generic problem $\min\{f(x) \mid x \in \Omega\}$. Given any net $\mathcal{S}$ for Ω and any objective function value $\gamma \in \mathbb{R}$, we consider a selection rule that determines: (a) a distinguished point x^*_M for every $M \in \mathcal{S}$, (b) a subfamily of qualified sets $\mathcal{R} \subset \mathcal{S}$ such that, for all $x \in \bigcup_{s \in \mathcal{S} \backslash \mathcal{R}} s$, we have $f(x) \geq \gamma$, (c) a distinguished set $B_{\mathcal{R}}$ of $\mathcal{R}$. Basically, regions which cannot contain any global optimum are excluded from consideration starting from the current iteration. For each set of $\mathcal{R}$, it iteratively calculates a distinguished point (for example, by applying a local optimization procedure to the specified region); $B_{\mathcal{R}}$ is then picked for further net refinement. The prototype algorithm below also relies on a selection rule used to partition a region.

1. (Initialization) Start with a net $\mathcal{S}_1$ for Ω, set $x_0 = \emptyset$ and let γ_0 be any strict upper bound for $f(\Omega)$. Set $\mathcal{P}_1 = \mathcal{S}_1$, $k = 1$, $\sigma_0 = \emptyset$.
2. (Evaluation) Let $\sigma_k = \{x^*_M \mid M \in \mathcal{P}_k\}$ be the set of all distinguished points.
3. (Incumbent) Let x_k be the point in $\{x_{k-1}\} \cup \sigma_k$ such that $\gamma_k = f(x_k)$ is lowest; set $x^* = x_k$.
4. (Screening) Determine the family $\mathcal{R}_k$ of qualified sets of $\mathcal{S}_k$ (in other words, from now on ignore those sets that can be shown not to contain a solution with objective value lower than γ_k).
5. (Termination) If $\mathcal{R}_k = \emptyset$, terminate. The problem is infeasible if $\gamma_k \geq \gamma_0$; otherwise x^* is the global optimum.
6. (Selection) Select the distinguished set $M_k = B_{\mathcal{R}_k} \in \mathcal{R}_k$ and partition it according to a pre-specified branching rule. Let $\mathcal{P}_{k+1}$ be a partition of $B_{\mathcal{R}_k}$. In $\mathcal{R}_k$, replace M_k by $\mathcal{P}_{k+1}$, thus obtaining a new refinement net $\mathcal{S}_{k+1}$. Set $k \leftarrow k + 1$ and go back to Step 2.

A Branch-and-Select algorithm is *convergent* if γ^*, defined as $\lim_{k \to \infty} \gamma_k$, is such that $\gamma^* = \inf f(\Omega)$. A selection rule is *exact* if: (i) $\inf(\Omega \cap M) \geq \gamma^*$ for all M such that $M \in \mathcal{R}_k$ for all k (i.e. each region that remains qualified forever in the solution process is such that $\inf(\Omega \cap M) \geq \gamma^*$); (ii) the limit M_∞ of any filter $\langle M_k \rangle$ is such that $\inf f(\Omega \cap M_\infty) \geq \gamma^*$. In this theoretical set-up, it is easy to show that a Branch-and-Select algorithm using an exact selection rule converges.

2.1 Theorem
A Branch-and-Select algorithm using an exact selection rule converges.

Proof. Suppose, to get a contradiction, that there is $x \in \Omega$ with $f(x) < \gamma^*$. Let $x \in M$ with $M \in \mathcal{R}_n$ for some $n \in \mathbb{N}$. Because of condition (i) above, M cannot remain qualified forever; furthermore, unqualified regions may not, by hypothesis, include points with better objective function values than the current incumbent γ_k. Hence M must necessarily be split at some iteration $n' > n$. So x belongs to every M_n in some filter $\{M_n\}$, thus $x \in \Omega \cap M_\infty$. By condition (2) above, $f(x) \geq \inf f(\Omega \cap M_\infty) \geq \gamma^*$. The result follows. □

We remark that this algorithmic framework does not provide a guarantee of convergence in a finite number of steps. Consequently, most Branch-and-Select implementations make use of the concept of ε-optimality, rather than the usual definition of optimality. Recall that $x^* \in \Omega$ is a global optimum if, for all $x \in \Omega$, we have $f(x^*) \leq f(x)$. Given $\varepsilon > 0$, $\bar{x} \in \Omega$ is ε-globally optimal if there exist bounds $m \leq f(x^*) \leq M$ such that $f(\bar{x}) \in [m, M]$ and $M - m < \varepsilon$. By employing this notion and finding converging lower and upper bounds sequences to the incumbent at each step of the algorithm, it is easier to ensure a finite termination with an ε-global optimum. For a theoretically proven finite termination,

we would need some additional regularity assumptions (see for example [SS98, AKS00]); ε-optimality, however, is sufficient for most practical purposes.

2.2.1 Fathoming

Let $\mathcal{S}_k$ be the net at iteration k. For each region $M \in \mathcal{S}_k$, find a lower bounding solution x^*_M for $f(x)$ defined on $\Omega \cap M$ and the corresponding lower bounding objective function value $l(M)$. A set $M \in \mathcal{S}_k$ is qualified if $l(M) \leq \gamma_k$. The distinguished region is usually selected as the one with the lowest $l(M)$.

The algorithm is accelerated if one also computes an upper bound $u(M)$ on $\inf f(x)$ on $\Omega \cap M$ and uses it to bound the problem from above by rejecting any M for which $l(M)$ exceeds the best upper bound $u(M)$ that has been encountered so far. In this case, we say that the rejected regions have been *fathomed* (see Fig. 6). This acceleration device has become part of most Branch-and-Bound algorithms in the literature. Usually, the upper bound $u(M)$ is computed by solving the problem to local optimality in the current region: this also represents a practical alternative to the implementation of the evaluation step (step (2) in the algorithm of Section 2.2), since we can take the distinguished points $\omega(M)$ to be the local solutions $u(M)$ of the problem in the current regions. With a good numerical local solver, the accuracy of ε-global optimum is likely to be better than if we simply use $l(M)$ to define the distinguished points.

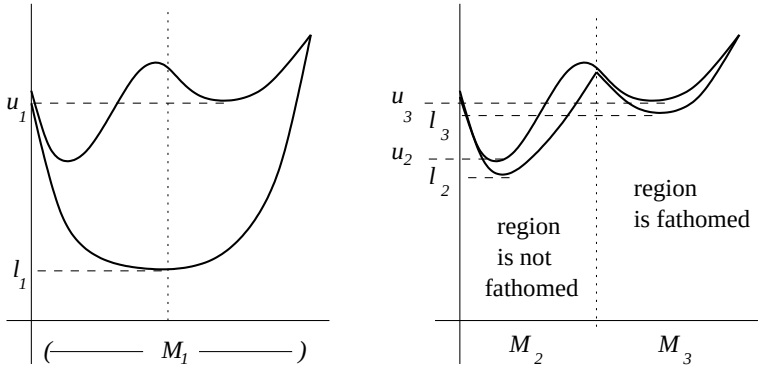


Figure 6: Fathoming via upper bound computation.

The convergence of the Branch-and-Bound algorithm is ensured if every filter $\{M_k | k \in K\}$ contains an infinite nested sequence $\{M_k | k \in K_1 \subseteq K\}$ such that:

$$\lim_{\substack{k \to \infty \\ k \in K_1}} l(M_k) = \gamma^*. \tag{8}$$

To establish this, we will show that under such conditions the selection procedure is exact. Let $\{M_k | k \in K\}$ be any filter and M_∞ its limit. Because of equation (8), $\inf f(\Omega \cap M_k) \geq l(M_k) \to \gamma^*$, hence $\inf f(\Omega \cap M_\infty) \geq \gamma^*$. Furthermore, if $M \in \mathcal{R}_k$ for all k then $\inf f(\Omega \cap M) \geq l(M) \geq l(M_k) \to \gamma^*$ as $k \to \infty$, i.e. $\inf f(\Omega \cap M) \geq \gamma^*$. Thus the selection procedure is exact and, by theorem 2.1, the Branch-and-Bound algorithm converges.

2.3 Example of solution by Branch-and-Select

We present a very simple example together with the solution obtained by using a Branch-and-Select algorithm.

2.2 Example

The problem $\min\{f(x) = \frac{1}{4}x + \sin(x) \mid x \geq -3, x \leq 6\}$ **is a very simple constrained nonlinear problem with two minima, only one of which is global (see Fig. 7). We solve it (graphically) with a spatial Branch-and-Bound approach (at each iteration we find lower and upper bounds to the optimum within a subregion of space; if these bounds are not su□ciently close, say to within a** $\varepsilon > 0$ **tolerance, we split the subregion in two, repeating the process on each subregion until all the regions have been examined). In this example we set** ε **to 0.15 (this is not realistic; in practice** ε **is set to values between** 1×10^{-6} **and** 1×10^{-3}**). At the first iteration we consider the region consisting of the whole** x **variable range**

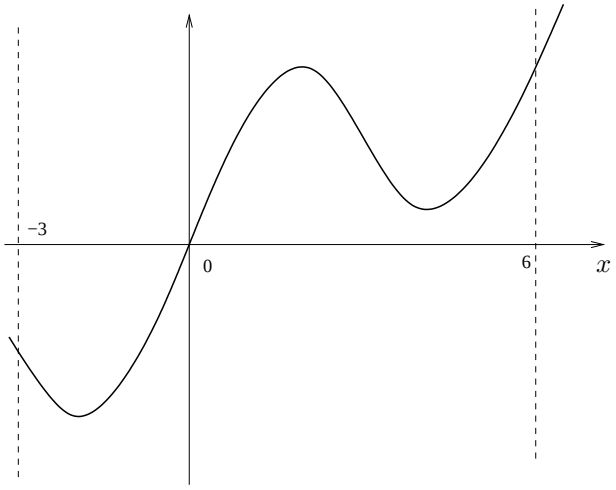


Figure 7: The problem $\min\{\frac{1}{4}x + \sin(x) \mid x \geq -3, x \leq 6\}$.

$-3 \leq x \leq 6$**. We calculate a lower bound by underestimating the objective function with a convex function: since for all** x **we have** $-1 \leq \sin(x) \leq 1$**, the function** $\frac{1}{4}x - 1$ **is a convex underestimator of the objective function. Limited to this example only, We can draw the tangents to** $f(x)$ **at the range endpoints** $x = -3$ **and** $x = 6$ **to make the underestimator tighter, as in Fig. 8 (the whole underestimator is the pointwise maximum of the three linear functions emphasized in the picture). Algebraically, the**

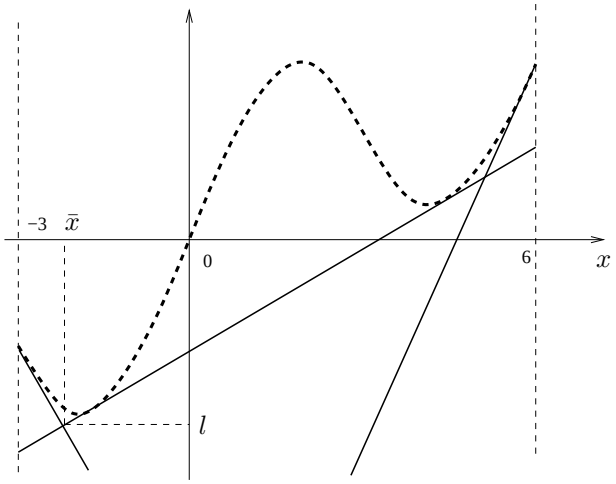


Figure 8: Convex underestimation of the objective function in Example 2.2 and optimal solution of the convex underestimation problem.

tangent to the objective function in the point $x = -3$ **is** $-0.74x - 3.11$**, whereas in the point** $x = 6$ **it**

is $1.21x - 6.04$. Thus the convex underestimator is $\bar{f}(x) = \max\{-0.74x - 3.11, \frac{1}{4}x - 1, 1.21x - 6.04\}$. Finding the minimum of $\bar{f}(x)$ in the region $-3 \leq x \leq 6$ yields a solution $\bar{x} = -2.13$ with objective function value $l = -1.53$ which is a valid lower bound for the original problem (indeed, the value of the original objective function at $\bar{x}$ is -1.42). Now we find an upper bound to the original objective function by locally solving the original problem. Suppose we pick the range endpoint $x = 6$ as a starting point and we employ Newton's method in one dimension: we find a solution $\tilde{x} = 4.46$ with objective function value $u = 0.147$. Since $|u - l| = |0.147 - (-1.53)| = 1.67 > \varepsilon$ shows that u and l are not su□ciently close to be reasonably sure that $\tilde{x}$ is the global optimum of the region under examination, we split this region in two. We choose the current $\tilde{x} = 4.46$ as the branch point, we add two regions $-3 \leq x \leq 4.46$, $4.46 \leq x \leq 6$ to a list of "unexamined regions" and we discard the original region $-3 \leq x \leq 6$. At the second iteration we pick the region $-3 \leq x \leq 4.46$ for examination and compute lower and upper bounds as in Fig. 9. We find

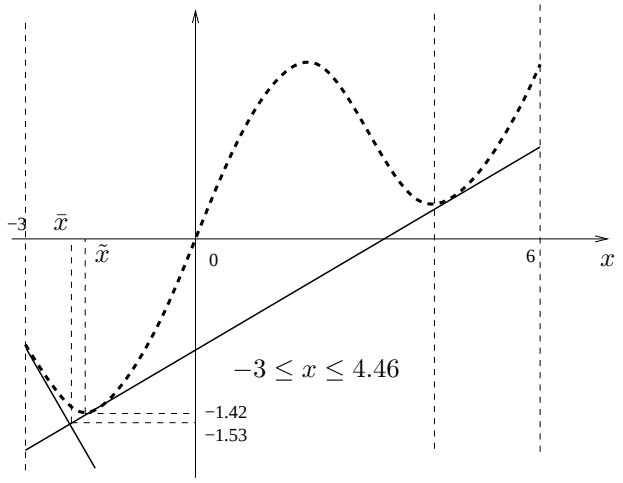


Figure 9: Second iteration of the sBB algorithm.

the same convexification solution $\bar{x}$ as before (with $l = -1.53$) and by starting the Newton search from the endpoint $x = -3$ we locate the local optimum $\tilde{x} = -1.823$ with $u = -1.42$. Since $|u - l| = 0.11 < \varepsilon$, we accept $\tilde{x}$ as the global optimum for the region under examination, and we update the "best solution found" $x^* = -1.823$ with associated function value $U = -1.42$. Since we have located the global optimum for this region, no branching occurs. This leaves us with just one region to examine, namely $4.46 \leq x \leq 6$. In the third iteration, we select this region and we compute lower and upper bounds (as in Fig. 10) to find $\bar{x} = \tilde{x} = 4.46$ with $l = 1.11$ and $u = 1.147$. Since $|u - l| = 0.04 < \varepsilon$, we have located the global optimum for this region, so there is no need for branching. Since $U = -1.42 < 1.147 = u$, we do not update the "best solution found", so on discarding this region the region list is empty. The algorithm terminates with global optimum $x^* = -1.823$. This example shows most of the important features of a typical spatial Branch-and-Bound algorithm run, save three: (a) the convex underestimation can (and usually is) modified in each region, so that it is tighter; (b) in multi-variable problems, the choice of the branch variable is a crucial task. This algorithm will be discussed in detail in Sect. 5; (c) in practice, pruning of region occurs when the lower bound in a region is higher than the current best solution. In this example, there was no pruning, as the region $4.46 \leq x \leq 6$ did not need to be branched upon. Had there been need for branching in that region, at the subsequent iterations two new regions would have been created with an associated lower bound $l = 1.11$. Since the current best solution has an objective function value of -1.42, which is strictly less than 1.11, both regions would have been pruned before being processed.

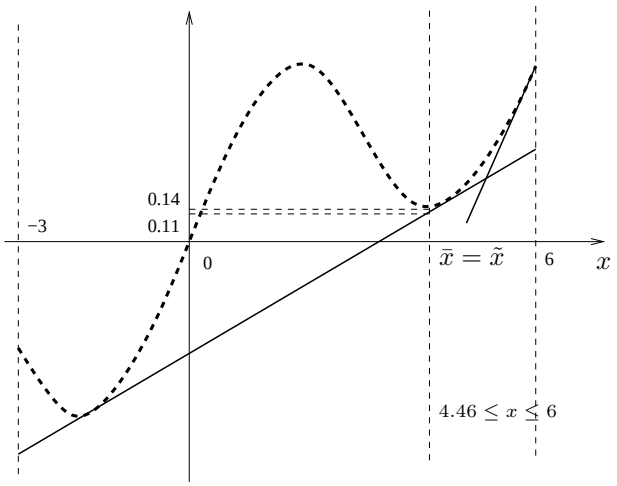


Figure 10: Third iteration of the sBB algorithm.

3 Local Optimization of NLPs

In this section we shall give an overview of local optimization. We shall introduce some basic notions of convex analysis, explain in some details the necessary and sufficient conditions for local optimality, and introduce an iterative approach to finding a local minimum of a given NLP.

3.1 Fundamental notions of convex analysis

3.1 Definition
A set $S \subseteq \mathbb{R}^n$ is *convex* if for any two points $x, y \in S$ the segment between them is wholly contained in S, that is, $\forall \lambda \in [0,1]\ (\lambda x + (1-\lambda)y \in S)$.

A linear equation $a^\top x = b$ where $a \in \mathbb{R}^n$ defines a hyperplane in $\mathbb{R}^n$. The corresponding linear inequality $a^\top x \leq b_0$ defines a closed half-space. Both hyperplanes and closed half-spaces are convex sets. Since any intersection of convex sets is a convex set, the subset of $\mathbb{R}^n$ defined by the system of closed half-spaces $Ax \geq b, x \geq 0$ is convex (where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$).

3.2 Definition
A subset $S \subseteq \mathbb{R}^n$ defined by a finite number of closed halfspaces is called a *polyhedron*. A bounded, non-empty polyhedron is a *polytope*.

Polyhedra and polytopes are very important in optimization as they are the geometrical representation of a feasible region expressed by linear constraints.

3.3 Definition
Let $S = \{x_i \mid i \leq n\}$ be a finite set of points in $\mathbb{R}^n$.

1. **The set span(S) of all linear combinations $\sum_{i=1}^n \lambda_i x_i$ of vectors in S is called the *linear hull* (or *linear closure*, or *span*) of S.**
2. **The set a(S) of all a□ne combinations $\sum_{i=1}^n \lambda_i x_i$ of vectors in S such that $\sum_{i=1}^n \lambda_i = 1$ is called the *affine hull* (or *affine closure*) of S.**

3. **The set cone**(S) **of all conic combinations** $\sum_{i=1}^{n} \lambda_i x_i$ **of vectors in** S **such that** $\forall i \leq n \ (\lambda_i \geq 0)$ **is called the** *conic hull* **(or** *conic closure***, or** *cone***) of** S**. A conic combination is** *strict* **if for all** $i \leq n$ **we have** $\lambda_i > 0$**.**

4. **The set conv**(S) **of all convex combinations** $\sum_{i=1}^{n} \lambda_i x_i$ **of vectors in** S **such that** $\sum_{i=1}^{n} \lambda_i = 1$ **and** $\forall i \leq n \ (\lambda_i \geq 0)$ **is called the** *convex hull* **(or** *convex closure***) of** S**. A convex combination is** *strict* **if for all** $i \leq n$ **we have** $\lambda_i > 0$**.**

3.4 Definition
Consider a polyhedron P **and a closed half-space** H **in** $\mathbb{R}^n$**. Let** $\overline{H \cap P}$ **be the a□ne closure of** $H \cap P$ **and** $d = \dim(\overline{H \cap P})$**. If** $d < n$ **then** $H \cap P$ **is a** *face* **of** P**. If** $d = 0$**,** $H \cap P$ **is called a** *vertex* **of** P**, and if** $d = 1$**, it is called an** *edge***. If** $d = n - 1$ **then** $H \cap P$ **is a** *facet* **of** P**.**

Having defined convex sets, we now turn our attention to convex functions.

3.5 Definition
A function $f : X \subseteq \mathbb{R}^n \to \mathbb{R}$ **is** *convex* **if for all** $x, y \in X$ **and for all** $\lambda \in [0, 1]$ **we have:**

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda) f(y).$$

Note that for this definition to be consistent, $\lambda x + (1 - \lambda)y$ **must be in the domain of** f**, i.e.** X **must be convex. This requirement can be formally relaxed if we extend** f **to be defined outside** X**.**

The main theorem in convex analysis, and the fundamental reason why convex functions are useful, is that a local minimum of a convex function over a convex set is also a global minimum, which we prove in Thm. 3.6. The proof is described graphically in Fig. 11.

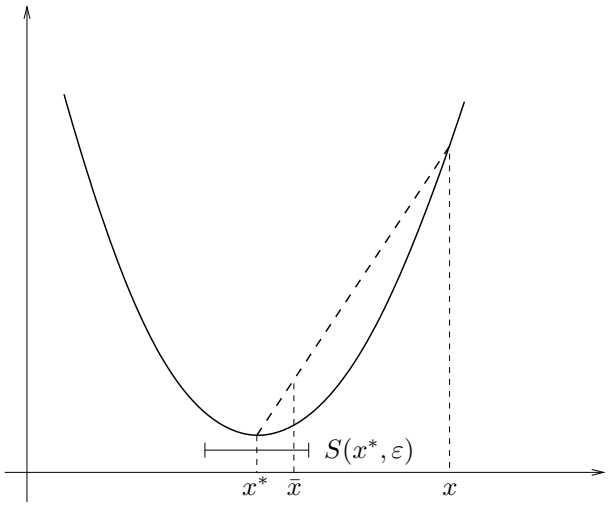


Figure 11: Graphical sketch of the proof of Thm. 3.6. The fact that $f(x^*)$ is the minimum in $S(x^*, \varepsilon)$ is enough to show that for any point $x \in X$, $f(x^*) \leq f(x)$.

3.6 Theorem
Let $X \subseteq \mathbb{R}^n$ **be a convex set and** $f : X \to \mathbb{R}$ **be a convex function. Given a point** $x^* \in X$**, suppose that there is a ball** $S(x^*, \varepsilon) \subset X$ **such that for all** $x \in S(x^*, \varepsilon)$ **we have** $f(x^*) \leq f(x)$**. Then** $f(x^*) \leq f(x)$ **for all** $x \in X$**.**

Proof. Let $x \in X$. Since f is convex over X, for all $\lambda \in [0, 1]$ we have $f(\lambda x^* + (1 - \lambda)x) \leq \lambda f(x^*) + (1 - \lambda) f(x)$. Notice that there exists $\bar{\lambda} \in (0, 1)$ such that $\bar{\lambda} x^* + (1 - \bar{\lambda})x = \bar{x} \in S(x^*, \varepsilon)$. Consider $\bar{x}$: by

convexity of f we have $f(\bar{x}) \leq \bar{\lambda} f(x^*) + (1-\bar{\lambda}) f(x)$. After rearrangement, we have

$$f(x) \geq \frac{f(\bar{x}) - \bar{\lambda} f(x^*)}{1-\bar{\lambda}}.$$

Since $\bar{x} \in S(x^*, \varepsilon)$, we have $f(\bar{x}) \geq f(x^*)$, thus

$$f(x) \geq \frac{f(x^*) - \bar{\lambda} f(x^*)}{1-\bar{\lambda}} = f(x^*),$$

as required. □

3.2 Necessary and sufficient conditions for local optimality

In this section we shall give necessary and sufficient conditions for a feasible point x^* to be a locally optimal point. Most of the work deals with deriving necessary conditions. The sufficient conditions are in fact very close to requiring convexity of the objective function and feasible region, as we shall see later.

As for the necessary conditions, We deal first with the case of an optimization problem as in Eq. (1) where the relations R are all equalities, the variables are unbounded ($x^L = -\infty, x^U = \infty$) and the index set Z is empty.

For a scalar function $f : \mathbb{R}^n \to \mathbb{R}$ we define the function vector ∇f as $(\nabla f)(x) = \left(\frac{\partial f(x)}{\partial x_1}, \ldots, \frac{\partial f(x)}{\partial x_n}\right)^\top$, where $x = (x_1, \ldots, x_n)^\top$. We denote by $\nabla f(x')$ the evaluation of ∇f at x'. If g is a vector-valued function $g(x) = (g_1(x), \ldots, g_m(x))$, then ∇g is the set of function vectors $\{\nabla g_1, \ldots, \nabla g_m\}$, and $\nabla g(x')$ is the evaluation of ∇g at x'.

Consider the following nonlinear, possibly nonconvex programming problem with continuous variables defined over $\mathbb{R}^n$:

$$\left.\begin{array}{rll} \min\limits_{x \in \mathbb{R}^n} & f(x) & \\ \text{s.t.} & g(x) & = \; 0, \end{array}\right\} \tag{9}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}^m$ are C^1 (i.e., continuously differentiable) functions.

A *constrained critical point* of problem (9) is a point $x^* \in \mathbb{R}^n$ such that $g(x^*) = 0$ and the directional derivative[2] of f along g is 0 at x^*. It is easy to show that such an x^* is a maximum, or a minimum, or a saddle point of $f(x)$ subject to $g(x) = 0$.

3.7 Theorem (Lagrange Multiplier Method)
If x^* is a constrained critical point of problem (9), $m \leq n$, and $\nabla g(x^*)$ is a linearly independent set of vectors, then $\nabla f(x^*)$ is a linear combination of the set of vectors $\nabla g(x^*)$.

Proof. Suppose, to get a contradiction, that $\nabla g(x^*)$ and $\nabla f(x^*)$ are linearly independent. In the case $\nabla f(x^*) = 0$, the theorem is trivially true, so assume $\nabla f(x^*) \neq 0$. Choose vectors $w_{m+2}, \ldots, w_n$ such that the set $J = \{\nabla g_1(x^*), \ldots, \nabla g_m(x^*), \nabla f(x^*), w_{m+2}, \ldots, w_n\}$ is a basis of $\mathbb{R}^n$. For $m+2 \leq i \leq n$ define $h_i(x) = \langle w_i, x \rangle$. Consider the map

$$F(x) = (F_1(x), \ldots, F_n(x)) = (g_1(x), \ldots, g_m(x), f(x), h_{m+2}(x), \ldots, h_n(x)).$$

Since the Jacobian (i.e., the matrix of the first partial derivatives) of F evaluated at x^* is J, and J is nonsingular, by the inverse function theorem F is a local diffeomorphism of $\mathbb{R}^n$ to itself. Thus, the equations $y_i = F_i(x)$ $(i \leq n)$ are a local change of coordinates in a neighbourhood of x^*. With respect

[2] A formal definition is given in [Mil69], p. 7-8.

to coordinates y_i, the surface S defined by $g(x) = 0$ is the coordinate hyperplane $0 \times \mathbb{R}^{n-m}$. Notice that the $(k+1)$-st coordinate, $y_{k+1} = f(x)$, cannot have a critical point on the coordinate plane $0 \times \mathbb{R}^{n-m}$, so x^* is not a constrained critical point, which is a contradiction. □

In the proof of the above theorem, we have implicitly used the fact that the criticality of points is invariant with respect to diffeomorphism (this can be easily established by showing that the derivatives of the transformed functions are zero at the point expressed in the new coordinates). The classical proof of the Lagrange Multiplier Theorem 3.7 is much longer but does not make explicit use of differential topology concepts (see [Apo61], p. 153). The proof given above has been taken almost verbatim from [Pug02].

By Theorem (3.7) there exist scalars $\lambda_1, \ldots, \lambda_m$, such that

$$\nabla f(x^*) + \sum_{i=1}^{m} \lambda_i \nabla g_i(x^*) = 0.$$

The above condition is equivalent to saying that if x^* is a constrained critical point of f s.t. $g(x^*) = 0$, then x^* is a critical point of the following (unconstrained) function:

$$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x). \tag{10}$$

Function (10) is called the *Lagrangian* of f w.r.t. g, and $\lambda_1, \ldots, \lambda_m$ are called the *Lagrange multipliers*. Intuitively, when we are optimizing over a subspace defined by $Ax = b$, the optimization direction must be a linear combination of the vectors which are normal to the hyperplane system $Ax = b$. The situation is depicted in Fig. 12.

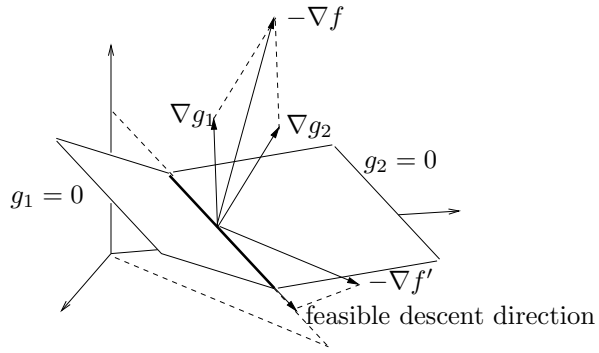


Figure 12: If ∇f is linearly dependent on the constraint gradients, there is no feasible descent direction. $\nabla f'$, which does not have this property, identifies a feasible descent direction when projected on the feasible space (the thick line).

Hence, in order to find the solution of problem (9), by Theorem 3.7, one should find all critical points of $L(x, \lambda)$ and then select the one with minimum objective function value. This approach is of limited use on all but the simplest problems. It does, however, provide at least necessary conditions for the characterization of a constrained minimum.

Notice, however, that problem (9) only caters for equality constraints; in particular, the range of each variable is $\mathbb{R}$ and there is no general way to reformulate an inequality constraint to an equation constraint without introducing other inequality constraints in the process (e.g. $g_1(x) \leq 0$ might be reformulated to $g_1(x) + t = 0$ by adding a variable t and a constraint $t \geq 0$). Furthermore, the proof of Theorem 3.7 is heavily based on the assumption that the only constraints of the problem are equality constraints, so an extension of the proof to encompass inequality constraints seems unlikely.

In order to take into account inequality constraints, we introduce the following simple example.

3.8 Example
Consider the problem $\min\{-x_1 - x_2 \mid x_1 - 1 \leq 0, x_2 - 1 \leq 0\}$. The minimum is obviously at $x^* = (1,1)$ as depicted in Fig. 13 (the figure only shows the positive quadrant; the feasible region actually extends to the other quadrants). By inspection, it is easy to see that at the point $x^* = (1,1)$ any further movement

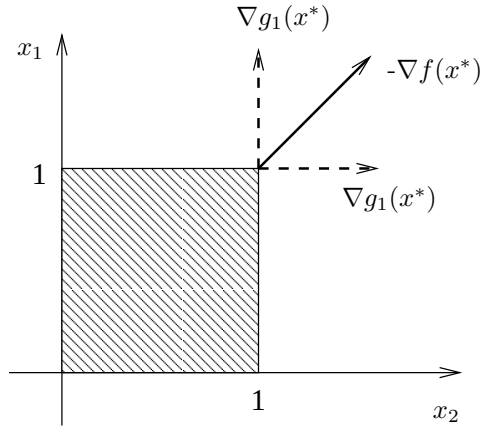


Figure 13: The problem of Example 3.8.

towards the direction of objective function decrease would take x^* outside the feasible region, i.e. no feasible direction decreases the objective function value. Notice that the descent direction at x^* is in the cone defined by the normal vectors to the constraints in x^*. In this particular case, the descent direction of the objective function is the vector $-\nabla f(x^*) = (1,1)$. The normal vector to $g_1(x) = x_1 - 1$ at x^* is $\nabla g_1(x^*) = (1,0)$ and the normal vector to $g_2(x) = x_2 - 1$ at x^* is $\nabla g_2(x^*) = (0,1)$. Notice that we can express $-\nabla f(x^*) = (1,1)$ as $\lambda_1(1,0) + \lambda_2(0,1)$ with $\lambda_1 = 1 > 0$ and $\lambda_2 = 1 > 0$. In other words, $(1,1)$ is a *conic combination* of $(1,0)$ and $(0,1)$.

If we now consider the problem of minimizing $\bar{f}(x) = x_1 + x_2$ subject to the same constraints as above, it appears clear that $x^* = (1,1)$ cannot be a local minimum, as there are many feasible descent directions. Take for example the direction vector $y = (-1,-1)$. This direction is feasible w.r.t. the constraints: $\nabla g_1(x^*)y = (1,0)(-1,-1) = (-1,0) \leq 0$ and $\nabla g_2(x^*)y = (0,1)(-1,-1) = (0,-1) \leq 0$. Moreover, it is a nonzero descent direction: $-\nabla \bar{f}(x^*)y = -(1,1)(-1,-1) = (1,1) > 0$.

In summary, either the descent direction at x^* is a conic combination of the gradients of the constraints (and in this case x^* may be a local minimum), or there is a nonzero feasible descent direction (and in this case x^* cannot be a local minimum).

The example above is an application of a well known theorem of alternatives called Farkas' Lemma. The following three results are necessary to introduce Farkas' Lemma, which will then be used in the proof of Theorem 3.13.

3.9 Theorem (Weierstraß)
Let $S \subseteq \mathbb{R}^n$ be a non-empty, closed and bounded set and let $f : S \to \mathbb{R}$ be continuous on S. Then there is a point $x^* \in S$ such that f attains its minimum value at x^*.

Proof. Since f is continuous on S and S is closed and bounded, then f is bounded below on S. Since S is non-empty, there exists a greatest lower bound α for the values of f over S. Let ε be such that $0 < \varepsilon < 1$ and consider the sequence of sets $S_k = \{x \in S \mid \alpha \leq f(x) \leq \alpha + \varepsilon^k\}$ for $k \in \mathbb{N}$. By the definition of infimum, S_k is non-empty for each k, so we can select a point $x^{(k)} \in S_k$ for each k. Since

S is bounded, there exists a convergent subsequence of $x^{(k)}$ which converges to a point x^*. Since S is closed, $x^* \in S$. By continuity of f, and because $\alpha \leq f(x^{(k)}) \leq \alpha + \varepsilon^k$, we have that the values $f(x^{(k)})$ (taken on the convergent subsequence) converge to α. Thus x^* is a point where f attains its minimum value $f(x^*) = \alpha$ (see Fig. 14). □

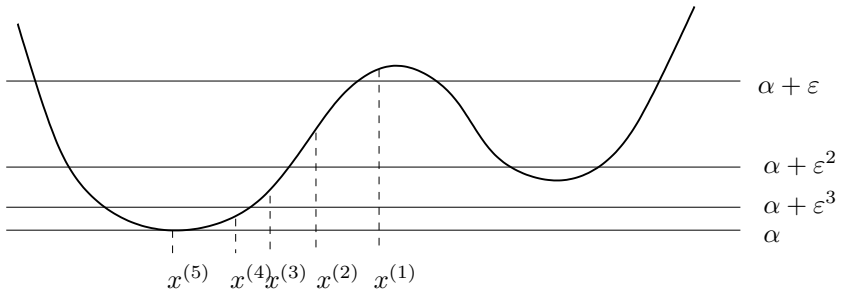


Figure 14: Weierstraß Theorem 3.9.

3.10 Proposition
Given a non-empty, closed convex set $S \subseteq \mathbb{R}^n$ and a point $x^* \notin S$, there exists a unique point $x' \in S$ with minimum distance from x^*. Furthermore, x' is the minimizing point if and only if for all $x \in S$ we have $(x^* - x')^\top (x - x') \leq 0$.

Proof. Let $x' \in S$ be the point minimizing $f(x) = ||x^* - x||$ subject to $x \in S$ (which exists by Weierstraß Theorem 3.9). To show that it is unique, notice first that f is convex: for $\lambda \in [0,1]$ and $x_1, x_2 \in S$ we have $f(\lambda x_1 + (1-\lambda)x_2) \leq f(\lambda x_1) + f((1-\lambda)x_2) = \lambda f(x_1) + (1-\lambda) f(x_2)$ by triangular inequality and homogeneity of the norm. Suppose now $y \in \mathbb{R}^n$ such that $f(y) = f(x')$. Since $x', y \in S$ and S is convex, the point $z = \frac{x'+y}{2}$ is in S. We have $f(z) \leq \frac{f(x')}{2} + \frac{f(y)}{2} = f(x')$. Since $f(x')$ is minimal, $f(z) = f(x')$. Furthermore, $f(z) = ||x^* - z|| = ||x^* - \frac{x'+y}{2}|| = ||\frac{x^*-x'}{2} + \frac{x^*-y}{2}||$. By the triangle inequality, $f(z) \leq \frac{1}{2}||x^* - x'|| + \frac{1}{2}||x^* - y||$. Since equality must hold as $f(z) = f(x') = f(y)$, vectors $x^* - x'$ and $x^* - y$ must be collinear, i.e. there is $\theta \in \mathbb{R}$ such that $x^* - x' = \theta(x^* - y)$. Since $f(x') = f(y)$, $|\theta| = 1$. Supposing $\theta = -1$ we would have $x^* = \frac{x'+y}{2}$, which by convexity of S would imply $x^* \in S$, contradicting the hypothesis. Hence $\theta = 1$ and $x' = y$ as claimed. For the second part of the Proposition, assume x' is the minimizing point in S and suppose there is $x \in S$ such that $(x^* - x')^\top (x - x') > 0$. Take a point $y \neq x'$ on the segment $\overline{x, x'}$. Since S is convex, $y \in S$. Furthermore, since $y - x'$ is collinear to $x - x'$, $(x^* - x')^\top (y - x') > 0$. Thus, the angle between $y - x'$ and $x^* - x'$ is acute. Choose y close enough to x' so that the largest angle of the triangle T having x^*, x', y as vertices is the angle in y (such a choice is always possible) as in Fig. 15. By elementary geometry, the longest side of such a triangle is the segment $\overline{x^*, x'}$ opposite to the angle in y. This implies $||x^* - y|| < ||x^* - x'||$, which is a contradiction, as x' was chosen to minimize the distance from x^*. Conversely, suppose for all $x \in S$ we have $(x^* - x')^\top (x - x') \leq 0$. This means that the angle in x' is obtuse (and hence it is the largest angle of T), and consequently the opposite side $\overline{x, x^*}$ is longer than the side $\overline{x', x^*}$. □

3.11 Proposition
Given a non-empty, closed convex set $S \subseteq \mathbb{R}^n$ and a point $x^* \notin S$, there exists a separating hyperplane $h^\top x = d$ (with $h \geq 0$) such that $h^\top x \leq d$ for all $x \in S$ and $h^\top x^* > d$.

Proof. Let x' be the point of S having minimum (strictly positive, since $x^* \notin S$) distance from x^*. Let $y = \frac{x'+x^*}{2}$ be the midpoint between x', x^*. The plane normal to the vector $x^* - y$ and passing through y separates x^* and S (see Fig. 16). □

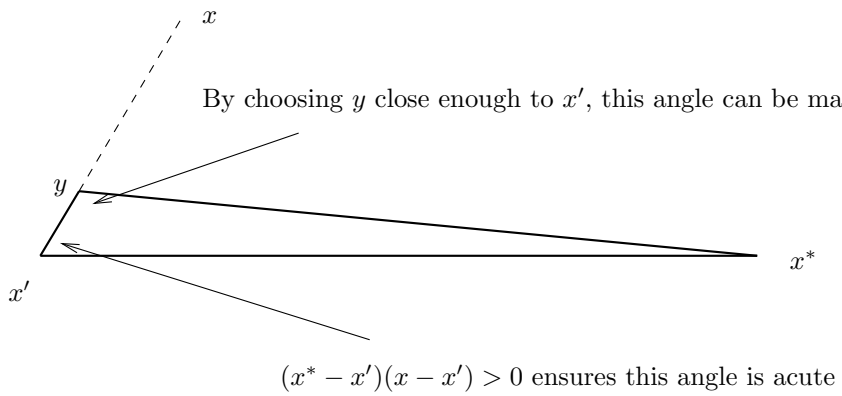


Figure 15: Second part of the proof of Prop. 3.10: the point y can always be chosen close enough to x' so that the angle in the vertex y is obtuse. The figure also shows that it is impossible for x' to be the minimum distance point from x^* if the angle in x' is acute and the set S containing points x', x, y is convex.

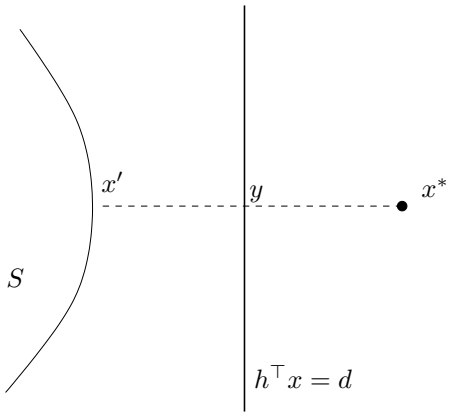


Figure 16: Prop. 3.11: separating hyperplane between a point x^*

minimum x^* of $f(x)$ such that $g(x^*) \leq 0$. It can be shown that if x^* is a constrained minimum then there is no nonzero feasible descent direction at x^* (see [BSS93], p. 141). We shall define a *feasible direction at* x^* as a direction vector y such that $(\nabla g(x^*))^\top y \leq 0$, and a *nonzero descent direction at* x^* as a direction vector y such that $(-\nabla f(x^*))^\top y > 0$.

3.13 Theorem (Karush-Kuhn-Tucker Conditions)
If x^* is a constrained minimum of problem (11), I is the maximal subset of $\{1, \ldots, m\}$ such that $g_i(x^*) = 0$ for all $i \in I$, and $\nabla \bar{g}$ is a linearly independent set of vectors (where $\bar{g} = \{g_i(x^*) \mid i \in I\}$), then $-\nabla f(x^*)$ is a conic combination of the vectors in $\nabla \bar{g}$, i.e. there exist scalars λ_i for all $i \in I$ such that the following conditions hold:

$$\nabla f(x^*) + \sum_{i \in I} \lambda_i \nabla g_i(x^*) = 0 \tag{12}$$

$$\forall i \in I \quad (\lambda_i \geq 0). \tag{13}$$

Proof. Since x^* is a constrained minimum and $\nabla \bar{g}$ is linearly independent, there is no nonzero feasible descent direction at x^* such that $(\nabla \bar{g}(x^*))^\top y \leq 0$ and $-\nabla f(x^*) y > 0$. By a direct application of Farkas' Lemma (3.12), there is a vector $\lambda \in \mathbb{R}^{|I|}$ such that $\nabla(\bar{g}(x^*))\lambda = -\nabla f(x^*)$ and $\lambda \geq 0$. □

The Karush-Kuhn-Tucker (KKT) necessary conditions (12), (13) can be reformulated to the following:

$$\nabla f(x^*) + \sum_{i=1}^{m} \lambda_i \nabla g_i(x^*) = 0 \tag{14}$$

$$\forall i \leq m \quad (\lambda_i g_i(x^*) = 0) \tag{15}$$

$$\forall i \leq m \quad (\lambda_i \geq 0). \tag{16}$$

This is easily verified by defining $\lambda_i = 0$ for all $i \notin I$. Conditions (15) are called *complementary slackness conditions*, and they express the fact that if a constraint is not active at x^* then its corresponding Lagrange multiplier is 0. A point x^* satisfying the KKT conditions is called a *KKT point*, or *KKT solution*.

Consider now a general NLP with both inequality and equality constraints:

$$\left.\begin{array}{lrcl} \min\limits_{x \in \mathbb{R}^n} & f(x) & & \\ \text{s.t.} & g(x) & \leq & 0 \\ & h(x) & = & 0, \end{array}\right\} \tag{17}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$ and $h : \mathbb{R}^n \to \mathbb{R}^p$ are C^1 functions. A *constrained minimum* of problem (17) is a minimum x^* of $f(x)$ such that $g(x^*) \leq 0$ and $h(x) = 0$.

By applying theorems 3.7 and 3.13, we can define the Lagrangian of problem (17) by the following:

$$L(x, \lambda, \mu) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + \sum_{i=1}^{p} \mu_i h_i(x), \tag{18}$$

and the corresponding KKT conditions as:

$$\left.\begin{array}{rcll} \nabla f(x^*) + \sum\limits_{i=1}^{m} \lambda_i \nabla g_i(x^*) + \sum\limits_{i=1}^{p} \mu_i \nabla h_i(x^*) & = & 0 & \\ \lambda_i g_i(x^*) & = & 0 & \forall i \leq m \\ \lambda_i & \geq & 0 & \forall i \leq m. \end{array}\right\} \tag{19}$$

In order to derive the general KKT conditions (19), it is sufficient to sum Eq. (12) and (14) and then divide the sum by the scalar 2.

This completes the discussion as regards the necessary conditions for local optimality. If a point x^* is a KKT point and the objective function is convex in a neighbourhood of x^* then x^* is a local minimum. These are the sufficient conditions which are used in practice in most cases. It turns out, however, that they are not the most stringent conditions possible.

3.14 Definition
Consider a function $f : X \subseteq \mathbb{R}^n \to \mathbb{R}^n$.

1. f **is** *quasiconvex* **if for all** $x, x' \in X$ **and for all** $\lambda \in [0,1]$ **we have:**

$$f(\lambda x + (1-\lambda)x') \leq \max\{f(x), f(x')\}.$$

2. f **is** *pseudoconvex* **if for all** $x, x' \in X$ **such that** $f(x') < f(x)$ **we have:**

$$(\nabla f(x))^\top (x' - x) < 0.$$

We state the sufficiency conditions in the following theorem; the proof can be found in [BSS93], p. 155.

3.15 Theorem
Let x^* be a KKT point of problem (11), I be as in Theorem 3.13, and S be the feasible region of a relaxation of problem (17) where all constraints g_i such that $i \notin I$ have been dropped. If there is a ball $B(x^*, \varepsilon)$ with $\varepsilon > 0$ such that f is pseudoconvex over $B(x^*, \varepsilon) \cap S$ and g_i are di□erentiable at x^* and quasiconvex over $N(x) \cap S$ for all $i \in I$, then x^* is a local minimum of problem (17).

3.3 A local optimization algorithm: SQP

There are many different algorithmic approaches to local optimization of NLPs in the form (17), most of which are based on the theoretical results of Section 3.2. Currently, the most widely employed of these algorithms is Sequential Quadratic Programming (SQP). The SQP method tries to solve the KKT conditions (14)-(16) iteratively by finding the minimizing solution of quadratic approximations of the objective function subject to linearized constraints.

We start with an arbitrary initial point $x \in X$ and generate a sequence $x^{(k)}$ using the relation

$$x^{(k+1)} = x^{(k)} + \alpha d,$$

where α is the steplength and d the search direction vector (this means that at each iteration we proceed in the direction of the vector d by an amount αd). Both α and d are updated at each iteration. The steplength α (in $(0,1]$) is adjusted at each iteration using a line search optimization (in practice, the point on the line in the d direction yielding the best reduction in objective function value is selected). The search direction vector d is obtained at each step by solving the following quadratic optimization problem (m, p, f, g and h are as in problem (17)):

$$\left.\begin{array}{lrcll} \min\limits_{d} & (\nabla f(x^{(k)}))^\top d + \frac{1}{2} d^\top H(x^{(k)}) d & & & \\ \text{s.t.} & (\nabla g_i(x^{(k)}))^\top d & \leq & 0 & \forall i \leq m \\ & (\nabla h_i(x^{(k)}))^\top d & = & 0 & \forall i \leq p, \end{array}\right\} \tag{20}$$

where $H(x^{(k)})$ is a positive semi-definite approximation to the Hessian of the objective function (i.e. the square $n \times n$ matrix whose (i,j)-th element is $\frac{\partial^2 f}{\partial x_i \partial x_j}$), evaluated at $x^{(k)}$. It can be shown that necessary and sufficient conditions for a function to be convex is that its Hessian is positive semi-definite, so problem (20) above actually corresponds to the best convex quadratic approximation of the original problem (17). Thus, a KKT point for (20) is a globally optimal solution of (20) by Theorem 3.15.

The KKT conditions for each subproblem (20) are solved directly (e.g. by Newton's method) to obtain the solution of (20). If $||d||$ is less than a pre-defined ε tolerance, the current point $x^{(k)}$ solves the KKT conditions for the original problem (17) and the process terminates. Notice that solving the KKT conditions for problem (20) also provides current solution values for the Lagrange multipliers μ, λ (which upon termination are also valid for the original problem (17)).

The SQP approach is one of the most promising in the field of local optimization of NLPs, and there are many variants that improve convergence speed and robustness. Recall from Sect. 2 that most global optimization algorithms use local NLP solvers as "black-box" procedures. From the point of view of global optimization the weakest link in the chain is the inherent unreliability of most local optimization algorithms, including SQP. In practice, there are many cases where the SQP method fails to converge on even the most innocent-looking NLPs, as the subproblems (20) can be defective for many reasons, the main ones being:

- the linearized constraints of (20) may be infeasible, even though the original constraints of (17) are feasible;
- the Jacobian of the constraints may not have full rank.

Establishing a cause for these occurrences is generally not an easy task. The subproblem may be defective locally (because the approximation to the original problem is not a good one) or because the original problem is ill-posed. Some of these problems may be overcome by simply changing the starting point: in particular, selecting a feasible starting point is a definite help for the accuracy of the quadratic approximation. However, finding a feasible starting point actually requires the solution of another nonconvex problem whose related SQP subproblems simply minimize the norm of the search direction d subject to

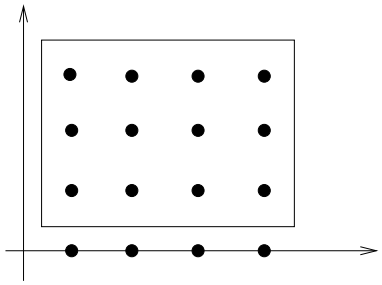


Figure 17: Projecting a grid distribution in $\mathbb{R}^2$ on the coordinate axes reduces the number of projected points. In this picture, $N = 12$ but the projected points are just 4.

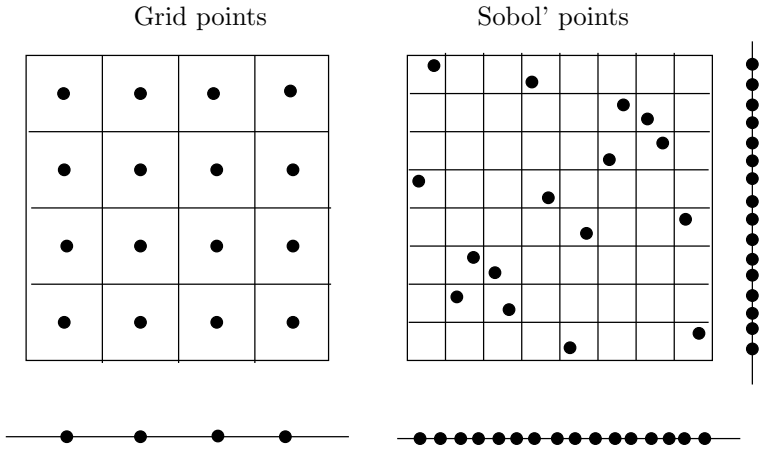


Figure 18: Comparison between projected distribution of grid points and Sobol' points in $\mathbb{R}^2$.

2. (Termination) If a pre-determined termination condition is verified, stop.

3. (Sampling) Sample a point q_k from a Sobol' sequence; add $(q_k, f(q_k))$ to Q.

4. (Clustering distance) Compute a distance r_k (which is a function of k and n; there are various ways to compute this distance, so this is considered as an "implementation detail" — one possibility is $r_k = \beta k^{-\frac{1}{n}}$, where β is a known parameter).

5. (Local phase) If there is no previously sampled point $q_j \in Q$ (with $j < k$) such that $||q_k - q_j|| < r_k$ and $f(q_j) \leq f(q_k) - \varepsilon$, solve problem (1) locally with q_k as a starting point to find a solution y with value $f(y)$. If $y \notin S$, add y to S. Set $k \leftarrow k + 1$ and repeat from step 2.

The algorithm terminates with a list S of all the local minima found. Finding the global minimum is then a trivial matter of identifying the minimum with lowest objective function value $f(y)$. Two of the most common termination conditions are (a) maximum number of sampled points and (b) maximum time limit exceeded.

A discussion of how Sobol' sequences can be generated is beyond the scope of this paper. A good reference is [PTVF97], p.311.

5 The spatial Branch-and-Bound algorithm

The spatial Branch-and-Bound (sBB) algorithm described in this section is a deterministic algorithm that solves NLPs in form (1). The general mathematical structure and properties of sBB algorithms aimed at solving nonconvex NLPs are explained in Section 2.2. The convergence proof theorem 2.1 also covers the sBB algorithm described in this section. The Branch-and-Reduce method [RS95] is an sBB algorithm with strong emphasis on variable range reduction. The αBB algorithm [AMF95, Adj98, AAMF96, AAF98] is an sBB whose main feature is that the convex underestimators for general twice-differentiable nonconvex terms can be constructed automatically. The reduced-space Branch-and-Bound algorithm [EP97] identifies *a priori* a reduced set of branching variables so that less branching is required. The generalized Branch-and-Cut framework proposed in [KB00] derives cuts from violated constraints in three sub-problems related to the original problem. In this exposition, we follow a particular type of sBB algorithm called *spatial Branch-and-Bound with symbolic reformulation* [Smi96, SP99, Lib04b].

Central to the each sBB algorithm is the concept of a *convex relaxation* of the original nonconvex problem; this is a convex problem whose solution is guaranteed to provide an underestimation for the objective function optimal value in the original problem. At each iteration of the algorithm, restrictions of the original problem and its convex relaxations to particular sub-regions of space are solved, so that a lower and an upper bound to the optimal value of the objective function can be assigned to each sub-region; if the bounds are very close together, a global optimum relative to the subregion has been identified. The particular selection rule of the sub-regions to examine makes it possible to exhaustively explore the search space rather efficiently. Since the sub-regions of space are usually hyper-rectangles defined by restrictions of the variable ranges, it is very important to start with the smallest variable ranges guaranteeing feasibility of the problem. This problem is addressed in Sect. 5.1.

Most sBB algorithms for the global optimization of nonconvex NLPs conform to a general framework of the following form:

1. (Initialization) Initialize a list of regions to a single region comprising the entire set of variable ranges. Set the convergence tolerance $\varepsilon > 0$, the best objective function value found up to the current step as $U := \infty$ and the corresponding solution point as $x^* := (\infty, \ldots, \infty)$. Optionally, perform optimization-based bounds tightening (see Sect. 5.1.1).

2. (Choice of Region) If the list of regions is empty, terminate the algorithm with solution x^* and objective function value U. Otherwise, choose a region R (the "current region") from the list (see Sect. 5.2). Delete R from the list. Optionally, perform feasibility-based bounds tightening on R (see Sect. 5.1.2).

3. (Lower Bound) Generate a convex relaxation of the original problem in the selected region R (see Sect. 5.3) and solve it to obtain an underestimation l of the objective function with corresponding solution $\bar{x}$. If $l > U$ or the relaxed problem is infeasible, go back to step 2.

4. (Upper Bound) Attempt to solve the original (generally nonconvex) problem in the selected region to obtain a (locally optimal) solution $\tilde{x}$ with objective function value u (see Sect. 5.4). If this fails, set $u := +\infty$ and $\tilde{x} = (\infty, \ldots, \infty)$.

5. (Pruning) If $U > u$, set $x^* = \tilde{x}$ and $U := u$. Delete all regions in the list that have lower bounds bigger than U as they cannot possibly contain the global minimum.

6. (Check Region) If $u - l \leq \varepsilon$, accept u as the global minimum for this region and return to step 2. Otherwise, we may not yet have located the region global minimum, so we proceed to the next step.

7. (Branching) Apply a branching rule to the current region to split it into sub-regions (see Sect. 5.5). Add these to the list of regions, assigning to them an (initial) lower bound of l. Go back to step 2.

The most outstanding feature of sBB algorithm described in this section is the automatic construction of the convex relaxation via symbolic reformulation. This involves identifying all the nonconvex terms in the problem and replacing them with the respective convex relaxations. The algorithm that carries out this task is symbolic in nature as it has to recognize the nonconvex operators in any given function.

Below, we consider some of the key steps of the algorithm in more detail.

5.1 Bounds tightening

These procedures appear in steps 1 and 2 of the algorithm structure outlined above. They are optional in the sense that the algorithm will, in principle, converge even without them. Depending on how computationally expensive and how effective these procedures are, in some cases convergence might actually be faster if these optional steps are not performed. In the great majority of cases, however, the bounds tightening steps are essential to achieve fast convergence.

Two major bounds tightening schemes have been proposed in the literature: optimization-based and feasibility-based (both are described below).

5.1.1 Optimization-based bounds tightening (step 1)

The optimization-based bounds tightening procedure identifies the smallest range of each variables subject to the convex relaxation of the problem to remain feasible. This ensures that the sBB algorithm will not have to explore hyper-rectangles which do not actually contain any feasible point.

Unfortunately, this is a computationally expensive procedure which involves solving at least $2n$ convex NLPs (or LPs if a linear convex relaxation is employed) where n is the number of problem variables. Let $\alpha \leq \bar{g}(x) \leq \beta$ be the set of constraints in the relaxed (convex) problem (α, β are the constraint limits). The following procedure will construct sequences $x^{L,k}, x^{U,k}$ of lower and upper variable bounds which converge to new variable bounds that are at least as tight as, and possibly tighter than x^L, x^U.

1. Set $x^{L,0} \leftarrow x^L$, $x^{U,0} \leftarrow x^U$, $k \leftarrow 0$.
2. Repeat

$$\begin{aligned} x_i^{L,k} &\leftarrow \min\{x_i \mid \alpha \leq \bar{g}(x) \leq \beta \wedge x^{L,k-1} \leq x \leq x^{U,k-1}\}, \quad \forall i \leq n; \\ x_i^{U,k} &\leftarrow \max\{x_i \mid \alpha \leq \bar{g}(x) \leq \beta \wedge x^{L,k-1} \leq x \leq x^{U,k-1}\}, \quad \forall i \leq n; \\ k &\leftarrow k+1. \end{aligned}$$

   until $x^{L,k} = x^{L,k-1}$ and $x^{U,k} = x^{U,k-1}$.

Because of the associated cost, this type of tightening is normally performed only once, at the first step of the algorithm.

5.1.2 Feasibility-based bounds tightening (step 2)

This procedure is computationally cheaper than the one described above, and as such it can be applied at each and every region considered by the algorithm. Variable bounds are tightened by using the problem constraints to calculate extremal values attainable by the variables. This is done by isolating a variable on the left hand side of a constraint and evaluating the right hand side extremal values by means of interval arithmetic.

Feasibility-based bounds tightening is trivially easy for the case of linear constraints. Given linear constraints in the form $l \leq Ax \leq u$ where $A = (a_{ij})$, it can be shown that, for all $1 \leq j \leq n$:

$$\begin{aligned} x_j \in \quad & \left[\max\left(x_j^L, \min_i \left(\tfrac{1}{a_{ij}} \left(l_i - \sum_{k \neq j} \max(a_{ik}x_k^L, a_{ik}x_k^U) \right) \right) \right) \right. \quad , \\ & \left. \min\left(x_j^U, \max_i \left(\tfrac{1}{a_{ij}} \left(u_i - \sum_{k \neq j} \min(a_{ik}x_k^L, a_{ik}x_k^U) \right) \right) \right) \right] \quad \text{if } a_{ij} > 0 \\ x_j \in \quad & \left[\max\left(x_j^L, \min_i \left(\tfrac{1}{a_{ij}} \left(l_i - \sum_{k \neq j} \min(a_{ik}x_k^L, a_{ik}x_k^U) \right) \right) \right) \right. \quad , \\ & \left. \min\left(x_j^U, \max_i \left(\tfrac{1}{a_{ij}} \left(u_i - \sum_{k \neq j} \max(a_{ik}x_k^L, a_{ik}x_k^U) \right) \right) \right) \right] \quad \text{if } a_{ij} < 0. \end{aligned}$$

As pointed out by Smith ([Smi96], p.202), feasibility-based bounds tightening can also be carried out for certain types of nonlinear constraints.

5.2 Choice of region (step 2)

The region selection at step 2 follows the simple policy of choosing the region in the list with the lowest lower objective function bound as the one which is most promising for further consideration (recall that the lower bound l calculated in each region is associated to the subregions after branching — see step 7 of the sBB algorithm).

5.3 Convex relaxation (step 3)

The convex relaxation solved at step 3 of the algorithm aims to find a guaranteed lower bound to the objective function value. In most global optimization algorithms, convex relaxations are obtained for problems in special (e.g. factorable) forms. In the sBB with symbolic reformulation the convex relaxation is calculated automatically for a problem in the most generic form (1) provided this is available in closed analytic form, which allows symbolic manipulation to be applied to each term of the mathematical expressions present in the problem.

The convex relaxation is built in two stages: first the problem is reduced to a *standard form* where the nonlinear terms are linearized. This means that each nonlinear term is replaced by an added variable, and a constraint of type "added variable = nonlinear term" is added to the problem formulation. Such constraints are called *defining equations*, or *defining constraints* because they define the meaning of the added variables. In the second stage of the convex relaxation each nonlinear term is replaced by the corresponding convex under- and overestimators.

5.3.1 Reformulation to standard form

This is the first stage towards the construction of the convex relaxation of the original problem via symbolic manipulation of the variables and constraints. An optimization problem (1) is in standard form when all the nonlinear terms have been linearized. The nonlinearities are then isolated from the rest of the problem and thus are easy to tackle by symbolic and numerical procedures.

There is no unique way to linearize nonlinear terms. For example, the term $x^3 + y^3 + y^2$ might be linearized as $w_1 + w_2 + w_3$ together with the defining equations $w_1 = x^3$, $w_2 = y^3$, $w_3 = y^2$. Recall however that the aim of this linearization is to provide tight convex underestimators and concave overestimators

to the nonlinear terms. Thus, were we able to construct an extremely tight convex relaxation of the term $y^3 + y^2$ as a whole, a better linearization would be $w_1 + w_4$ where $w_4 = y^3 + y^2$. In general, we follow the principle that we identify the simplest possible nonlinear term. This, however, is a guideline rather than a rule.

The following formulation describes a problem in a standard form that recognizes nonlinear operators with one and two operands.

$$\min x_\omega \tag{21}$$
$$l \leq Ax \leq u \tag{22}$$
$$x_k = x_i x_j \qquad \forall (i,j,k) \in \mathcal{M} \tag{23}$$
$$x_k = \frac{x_i}{x_j} \qquad \forall (i,j,k) \in \mathcal{D} \tag{24}$$
$$x_k = x_i^\nu \qquad \forall (i,k,\nu) \in \mathcal{P} \tag{25}$$
$$x_k = f_\mu(x_i) \qquad \forall (i,k,\mu) \in \mathcal{U} \tag{26}$$
$$x^L \leq x \leq x^U \tag{27}$$

where $x = (x_1, \ldots, x_n)$ are the problem variables, $\omega \leq n$, A is a constant, usually sparse, matrix, l, u are the linear constraint bounds, and x^L, x^U are the variable bounds. The defining constraints (23)–(26) include all the problem nonlinearities: $\mathcal{M}, \mathcal{D}$ are sets of triplets of variable indices which define bilinear and linear fractional terms respectively. $\mathcal{P}$ is a set of triplets defining power terms; each triplet comprises two variable indices i, k and the corresponding power exponent $\nu \in \mathbb{R}$. $\mathcal{U}$ is a set of triplets which define terms involving univariate functions f_μ; each triplet comprises two variable indices i, k and a third index μ that identifies the type of the univariate function being applied (e.g. $\exp(\cdot)$, $\ln(\cdot)$).

Each of the constraints (23)-(26) has one of the forms:

$$\begin{aligned} \textit{added variable} &= \textit{operand (binary operator) operand} \\ \textit{added variable} &= \textit{(unary operator) operand} \end{aligned}$$

where *operand* is an original or an "added" variable i.e. one added to the original set of variables by the standardization procedure. The objective function is also replaced by the added variable x_ω, and a constraint of the form:

$$x_\omega = objective\ function$$

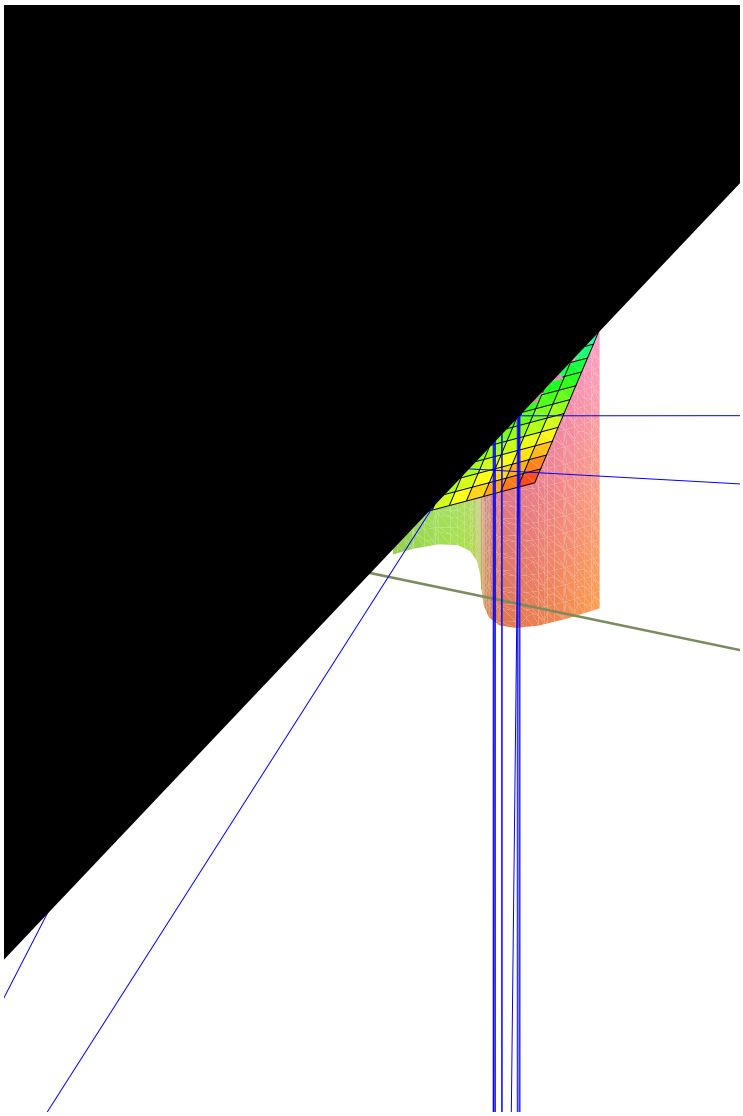
is added to the linear constraints.

5.1 Example

In this example we show the reformulation obtained for Example 2.2, together with its graphical representation. Since there is only one nonconvex term in the problem (namely, $\sin(x)$), the reformulation is immediate:

$$\left.\begin{array}{ll} \min\limits_{x,y} & \frac{1}{4}x + y \\ \textbf{s.t.} & y = \sin(x) \\ & -3 \leq x \leq 6 \\ & -1 \leq y \leq 1. \end{array}\right\}$$

The reformulation described in this section is called a *lifting*, as introducing new variables lifts the problem into a higher dimensional space. This is immediately apparent in this simple example. Compare Fig. 7 (the original problem) and its reformulation shown in Fig. 19.



5.3.2 Convexification

It is in this second stage of the process that the actual convex relaxation of the original problem is built. The convex relaxation is relative to the current region $[x^L, x^U]$, meaning that it depends on x^L, x^U. The algorithm for convexification is entirely symbolic (as opposed to numeric) and hence performs very efficiently even in the presence of very complicated mathematical expressions.

Having reduced a problem to standard form, we replace every nonconvex term with a convex envelope consisting of convex over- and underestimating inequality constraints. The rules we follow to construct over- and underestimators are as follows:

1. $x_i = x_j x_k$ is replaced by four linear inequalities (McCormick's envelopes, [McC76]):

$$\begin{aligned} x_i &\geq x_j^L x_k + x_k^L x_j - x_j^L x_k^L && (28)\\ x_i &\geq x_j^U x_k + x_k^U x_j - x_j^U x_k^U && (29)\\ x_i &\leq x_j^L x_k + x_k^U x_j - x_j^L x_k^U && (30)\\ x_i &\leq x_j^U x_k + x_k^L x_j - x_j^U x_k^L && (31) \end{aligned}$$

2. $x_i = x_j / x_k$ is reformulated to $x_i x_k = x_j$ and the convexification rules for bilinear terms (28)-(31) are applied.

3. $x_i = f_\mu(x_j)$ where f_μ is concave univariate is replaced by two inequalities: the function itself and the secant:

$$\begin{aligned} x_i &\leq f_\mu(x_j) && (32)\\ x_i &\geq f_\mu(x_j^L) + \frac{f_\mu(x_j^U) - f_\mu(x_j^L)}{x_j^U - x_j^L}(x_j - x_j^L) && (33) \end{aligned}$$

4. $x_i = f_\mu(x_j)$ where f_μ is convex univariate is replaced by:

$$\begin{aligned} x_i &\leq f_\mu(x_j^L) + \frac{f_\mu(x_j^U) - f_\mu(x_j^L)}{x_j^U - x_j^L}(x_j - x_j^L) && (34)\\ x_i &\geq f_\mu(x_j) && (35) \end{aligned}$$

5. $x_i = x_j^\nu$ where $0 < \nu < 1$ is treated as a concave univariate function in the manner described in point 3 above.

6. $x_i = x_j^{2m}$ for any $m \in \mathbb{N}$ is treated as a convex univariate function in the manner described in point 4 above.

7. $x_i = x_j^{2m+1}$ for any $m \in \mathbb{N}$ can be convex, concave, or piecewise convex and concave with a turning point at 0. If the range of x_j does not include 0, the function is convex or concave and falls into a category described above. A complete discussion of convex underestimators and concave overestimators for this term when the range includes 0 can be found in [LP03].

5.4 Local solution of the original problem (step 4)

The local solution to the original nonconvex problem restricted to the current region is obtained by calling a local solver code as a "black-box" procedure. As has been remarked in Sect. 3.3, even the most advanced local solution techniques for nonconvex NLPs are inherently very fragile and can fail for various reasons even when a local optimum exists. This fragility is unfortunately partly inherited by the global

solution algorithm, even though the sBB algorithm in itself is very robust (i.e. it fails only when the local solver consistently fails to find a local optimum on each sub-region).

The most computationally expensive step in the sBB algorithm is typically the call to the local optimization procedure to find the upper bound to the problem at hand. This normally involves the numerical solution of a general non-convex NLP, which can take a very long time. If a good upper bound can be found for a region without resorting to the local optimization procedure, then it should be used without question.

The two methods described below should at least halve the number of upper bounding problems that are solved during the sBB algorithm. Note that a distinction is made between the variables that are present in the original NLP ("original variables") and those added by the standardization procedure ("added variables", cf. section 5.3.1).

5.4.1 Branching on added variables

Suppose that in the sBB algorithm an added variable w is chosen as the branch variable (see Sect. 5.5). The current region is then partitioned into two sub-regions along the w axis, the convex relaxations are modified to take the new variable ranges into account, and lower bounds are found for each sub-region. The upper bounds, however, are found by solving the original problem which is not dependent on the added variables. Thus the same exact original problem is solved at least three times in the course of the algorithm (i.e. once for the original region and once for each of its two sub-regions).

The obvious solution is for the algorithm to record the objective function upper bounds in each region. Whenever the branch variable is an added variable, avoid solving the original (upper bounding) problem and use the stored values instead.

5.4.2 Branching on original variables

Even when the branching occurs on an original problem variable, there are some considerations that help avoid solving local optimization problems unnecessarily. Suppose that the original variable x is selected for branching in a certain region. Then its range $[x^L, x^U]$ is partitioned into $[x^L, x']$ and $[x', x^U]$. If the solution of the upper bounding problem in $[x^L, x^U]$ is x^*, and $x^* \in [x^L, x']$, then it is unnecessary to solve the upper bounding problem again in the sub-region $[x^L, x']$ as an upper bound is already available at x^*. Of course, the upper bounding problem still needs to be solved for the other subregion $[x', x^U]$ (see Fig. 20).

5.5 Branching (step 7)

There are many branching strategies [Epp95] available for use in spatial Branch-and-Bound algorithms. Generally, branching involves two steps, namely determining the point (i.e. set of variable values) on which to branch, and finding the variable whose domain is to be sub-divided by the branching operation. Here, we use the solution $\tilde{x}$ of the upper bounding problem (step 4) as the branching point, if such a solution is found; otherwise the solution of the lower bounding problem $\bar{x}$ (step 3) is used. We then use the standard form to identify the nonlinear term with the largest error with respect to its convex relaxation. By definition of the standard form (21) (see Sect. 5.3.1), this is equivalent to evaluating the defining constraints (23)-(26) at $\bar{x}$ and choosing the one giving rise to the largest error in absolute value. In case the chosen defining constraint represents a unary operator, the only variable operand is chosen as the branch variable; if it represents a binary operator, the branch variable is chosen as the one whose value at the branching point is nearest to the midpoint of its range (see [Smi96], p. 205-207).

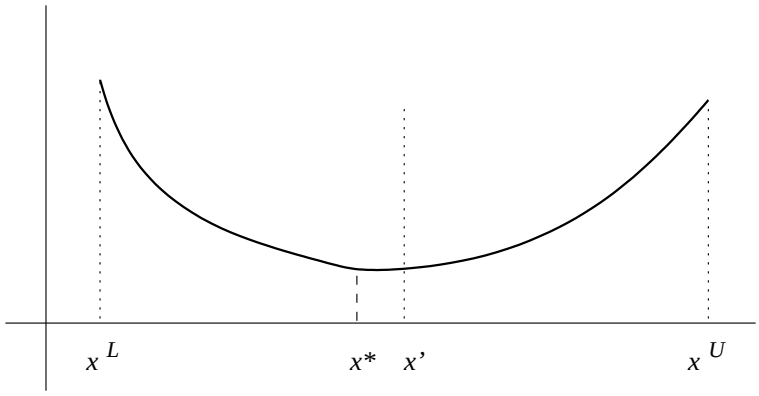


Figure 20: If the locally optimal solution in $[x^L, x^U]$ has already been determined to be at x^*, solving in $[x^L, x']$ is unnecessary.

6 Tight convex relaxations of bilinear problems

This section describes some very recent research being undertaken by the author as regards the application of the sBB algorithm to nonconvex problems involving bilinear terms and at least one linear inequality constraint. This particular structure (shared by many practically important classes of problems) sometimes makes it possible to considerably reduce the number of bilinear terms in the problem without ever changing the feasible region. In other words, an automatic exact reformulation of the problem, which involves less bilinear terms than the original problem formulation, is sometimes possible. A more complete treatment of this topic (together with very encouraging numerical results obtained on problems drawn from the oil industry) is given in [Lib04b, Lib04a, Lib05, LP06]. The following is a summary of the most important idea behind this exact reformulation.

Consider the *generalized bilinear problem*:

$$\min\{x^\top Q x + f(x) \mid Ax = b, x \in X \subseteq \mathbb{R}^n\} \tag{36}$$

where Q is $n \times n$ upper-triangular matrix, f is a function $\mathbb{R}^n \to \mathbb{R}$, A is an $m \times n$ matrix with full rank m, $b \in \mathbb{R}^m$ and X is a definition set for the variables which might include variable bounds, linear or nonlinear constraints, integrality constraints on the variables and so on. By substituting each bilinear term $x_i x_j$ with a linearizing variable w_i^j we obtain an exact reformulation:

$$\left.\begin{array}{rrcl} \min\limits_{x \in X} & p^\top w & + & f(x) \\ & Ax & = & b \\ \forall i \leq j \leq n & w_i^j & = & x_i x_j. \end{array}\right\} \tag{37}$$

for a suitable constant vector $p \in \mathbb{R}^{\frac{1}{2}n(n+1)}$ which can be easily obtained from the matrix Q. Multiply the system $Ax = b$ by each problem variable x_k to obtain the following *reduction constraint system*:

$$\forall k \leq n \ (Aw_k - bx_k = 0), \tag{38}$$

where $w_k = (w_k^1, \ldots, w_k^n)^\top$. Substitute $b = Ax$ in (38) to obtain:

$$\forall k \leq n \ (A(w_k - x_k x) = 0);$$

now substitute $z_i^j = w_i^j - x_i x_j$ for all $i \leq j \leq n$ to get the following *companion system*:

$$\forall k \leq n \ (Az_k = 0), \tag{39}$$

where $z_k = (z_k^1, \ldots, z_k^n)^\top$. System (39) can be written as $Mz = 0$ for a suitable matrix M obtained from A (z is the vector of all z_i^j for $i \le j \le n$). Partition z in a set B and N of basic and nonbasic variables w.r.t. system $Mz = 0$. By setting all the nonbasic variables to zero, for $Mz = 0$ to hold, the basic variables must also be zero. Thus, by setting $w_i^j = x_i x_j$ for all (i,j) such that $z_i^j \in N$, the reduction constraint system (38) implies that $w_i^j = x_i x_j$ for all (i,j) such that $z_i^j \in B$. In other words, the system (38) of linear equations replaces those bilinear constraints in (37) corresponding to basic variables of system (39). Thus, the following reformulation of (36), involving more linear constraints but less bilinear terms than the original problem, is exact:

$$\left.\begin{array}{rrcl} \min\limits_{x\in X} & p^\top w & + & f(x) \\ & Ax & = & b \\ \forall k \le n & Aw_k - bx_k & = & 0 \\ \forall (i,j) : z_i^j \in N & w_i^j & = & x_i x_j. \end{array}\right\} \tag{40}$$

We shall now see an example of the application of the above theory.

6.1 Example

Consider the problem

$$\left.\begin{array}{ll} \min_x & x_1^2 + x_2^2 + x_3^2 - x_1x_2 - x_2x_3 - x_3 \\ & x_1 + x_2 + 2x_3 = 1 \\ & x_1 + 2x_2 + x_3 = 1. \end{array}\right\}$$

After the reformulation, this becomes

$$\left.\begin{array}{ll} \min_{x,w} & w_1^1 + w_2^2 + w_3^3 - w_2^1 - w_3^2 - x_3 \\ & x_1 + x_2 + 2x_3 = 1 \\ & x_1 + 2x_2 + x_3 = 1 \\ & w_1^1 = x_1^2 \\ & w_2^1 = x_1x_2 \\ & w_2^2 = x_2^2 \\ & w_3^2 = x_2x_3 \\ & w_3^3 = x_3^2. \end{array}\right\}$$

Notice that the bilinear constraint $w_3^1 = x_1x_3$ is missing from the formulation, but we shall need it when considering the reduction constraint systems. The linear constraints of this problem correspond to the system $Ax = b$, where

$$A = \begin{pmatrix} 1 & 1 & 2 \\ 1 & 2 & 1 \end{pmatrix} \tag{41}$$

$x = (x_1, x_2, x_3)$**, and** $b = (1,1)$**. Multiplying this by x_1, x_2, x_3 we get the reduction constraint systems**

$$\begin{array}{rcl} w_1^1 + w_2^1 + 2w_3^1 - x_1 &=& 0 \\ w_1^1 + 2w_2^1 + w_3^1 - x_1 &=& 0 \\ w_2^1 + w_2^2 + 2w_3^2 - x_2 &=& 0 \\ w_2^1 + 2w_2^2 + w_3^2 - x_2 &=& 0 \\ w_3^1 + w_3^2 + 2w_3^3 - x_3 &=& 0 \\ w_3^1 + 2w_3^2 + w_3^3 - x_3 &=& 0. \end{array}$$

If we let

$$\begin{array}{rcl} z_1^1 &=& w_1^1 - x_1^2 \\ z_2^1 &=& w_2^1 - x_1x_2 \\ z_3^1 &=& w_3^1 - x_1x_3 \\ z_2^2 &=& w_2^2 - x_2^2 \\ z_3^2 &=& w_3^2 - x_2x_3 \\ z_3^3 &=& w_3^3 - x_3^2, \end{array}$$

the reduction constraint systems above correspond to the companion system $Bz = 0$, where

$$B = \begin{pmatrix} 1 & 1 & 2 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 2 & 0 \\ 0 & 1 & 0 & 2 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 2 \\ 0 & 0 & 1 & 0 & 2 & 1 \end{pmatrix}$$

and $z = (z_1^1, z_2^1, z_3^1, z_2^2, z_3^2, z_3^3)$. This matrix has row echelon form (obtained without row or column permutations)

$$\begin{pmatrix} 1 & 1 & 2 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 2 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 2 & -2 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

and thus rank 5. This means that the reduction constraint systems can replace 5 out of 6 w-defining constraints. In particular, we should keep the one that corresponds to the nonbasic variable z_3^3, i.e. $w_3^3 = x_3^2$.

Whilst the idea of multiplying linear constraints by problem variables is not new [SA92], the analysis that shows the reduction in the number of bilinear terms is, to the best of our knowledge, completely original. The citations listed above extend this idea in many directions, including application to large-scale sparse bilinear problems, where a fundamental modification of this idea is necessary in order for it to be useful. Efficient algorithms for reduction constraints generation for problems with dense and sparse matrix Q have been proposed and successfully tested on various problems, including the pooling and blending problems from the oil industry (see Example 1.2), where the computational improvements w.r.t. an sBB algorithm without reduction constraints went up to 5 orders of magnitude.

7 Conclusion

This paper is a brief (mostly theoretical) overview of the main concepts in global and local nonconvex optimization, with a particular slant on deterministic methods for global optimization. Section 1 provides an introductory exposition of the field of optimization and includes a short history of global optimization. In section 2 we classify global optimization algorithms according to their algorithmic behaviour and convergence properties; we then explain the two-phase structure underlying most global optimization algorithm, and introduce the classes of stochastic global optimization algorithms and deterministic global optimization algorithms. Section 3 is about local nonlinear optimization, with basic material including convexity, necessary and sufficient conditions for local optimality, and a short description of the SQP algorithm. In Section 4 we describe a very successful stochastic optimization algorithm called SobolOpt. Section 5 contains an in-depth description and analysis of a deterministic global optimization algorithm called spatial Branch-and-Bound with symbolic reformulation. The important concept of convex relaxation is introduced therein. Finally, section 6 describes recent advances which allow to automatically generate a very tight convex relaxation for a large class of nonconvex problems containing bilinear terms.

Acknowledgements

We wish to express gratitude to Tanja Davidović, Sergei Kucherenko, Fabio Schoen, Pietro Belotti, Federico Malucelli and Ercam Atam for taking the time of reading and correcting this paper.

References

[AAF97] C.S. Adjiman, I.P. Androulakis, and C.A. Floudas. Global optimization of MINLP problems in process synthesis and design. *Computers & Chemical Engineering*, 21:S445–S450, 1997.

[AAF98] 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(9):1159–1179, 1998.

[AAMF96] C.S. Adjiman, I.P. Androulakis, C.D. Maranas, and C.A. Floudas. A global optimization method, αBB, for process design. *Computers & Chemical Engineering*, 20:S419–S424, 1996.

[ABH$^+$04] C. Audet, J. Brimberg, P. Hansen, S. Le Digabel, and N. Mladenović. Pooling problem: Alternate formulations and solution methods. *Management Science*, 50(6):761–776, 2004.

[ADFN98] 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 & Chemical Engineering*, 22(9):1137–1158, 1998.

[Adj98] C.S. Adjiman. *Global Optimization Techniques for Process Systems Engineering*. PhD thesis, Princeton University, June 1998.

[AF96] C.S. Adjiman and C.A. Floudas. Rigorous convex underestimators for general twice-differentiable problems. *Journal of Global Optimization*, 9(1):23–40, July 1996.

[AHU83] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *Data Structures and Algorithms*. Addison-Wesley, Reading, MA, 1983.

[AKS00] F.A. Al-Khayyal and H.D. Sherali. On finitely terminating branch-and-bound algorithms for some global optimization problems. *SIAM Journal of Optimization*, 10(4):1049–1057, 2000.

[AMF95] I. P. Androulakis, C. D. Maranas, and C. A. Floudas. *alpha*BB: A global optimization method for general constrained nonconvex problems. *Journal of Global Optimization*, 7(4):337–363, December 1995.

[Apo61] T.M. Apostol. *Mathematical Analysis*. Addison-Wesley, Reading, MA, 1961.

[ASF98] C.S. Adjiman, C.A. Schweiger, and C.A. Floudas. Mixed-integer nonlinear optimization in process synthesis. In D.-Z. Du and P.M. (Eds.) Pardalos, editors, *Handbook of Combinatorial Optimization, vol. I*, volume 1, pages 1–76. Kluwer Academic Publishers, Dordrecht, 1998.

[ATS99] N. Adhya, M. Tawarmalani, and N.V. Sahinidis. A lagrangian approach to the pooling problem. *Industrial and Engineering Chemistry Research*, 38:1956–1972, 1999.

[BCCS97] L.T. Biegler, T.F. Coleman, A.R. Conn, and F.N. Santosa, editors. *Large-scale Optimization with Applications*. Springer, 1997.

[BSS93] M.S. Bazaraa, H.D. Sherali, and C.M. Shetty. *Nonlinear Programming: Theory and Algorithms*. Wiley, Chichester, second edition, 1993.

[CDM91] A. Colorni, M. Dorigo, and V. Maniezzo. Distributed optimization by ant colonies. In F. Varela and P. Bourgine, editors, *Proceedings of the European Conference on Artificial Life*, pages 134–142, Amsterdam, 1991. ECAL, Paris, France, Elsevier.

[CS00] R. Chelouah and P. Siarry. Tabu search applied to global optimization. *European Journal of Operational Research*, 123:256–270, 2000.

[Dan63] G.B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, NJ, 1963.

[EP97] T.G.W. Epperly and E.N. Pistikopoulos. A reduced space branch and bound algorithm for global optimization. *Journal of Global Optimization*, 11:287:311, 1997.

[Epp95] T.G.W. Epperly. *Global Optimization of Nonconvex Nonlinear Programs using Parallel Branch and Bound*. PhD thesis, University of Winsconsin – Madison, 1995.

[ES89] E. Eskow and R. B. Schnabel. Mathematical modeling of a parallel global optimization algorithm. *Parallel Computing*, 12(3):315–325, December 1989.

[FHJ92] L.R. Foulds, D. Haughland, and K. Jornsten. A bilinear approach to the pooling problem. *Optimization*, 24:165–180, 1992.

[FS69] J.E. Falk and R.M. Soland. An algorithm for separable nonconvex programming problems. *Management Science*, 15:550–569, 1969.

[GBR95] C. Guus, E. Boender, and H.E. Romeijn. Stochastic methods. In Horst and Pardalos [HP95], pages 829–869.

[Ge89] R. P. Ge. A parallel global optimization algorithm for rational separable-factorable functions. *Applied Mathematics and Computation*, 32(1):61–72, July 1989.

[Gro96] I.E. Grossmann, editor. *Global Optimization in Engineering Design*. Kluwer Academic Publishers, Dordrecht, 1996.

[Han80] E.R. Hansen. Global constrained optimization using interval analysis. In K. Nickel, editor, *Interval Mathematics 1980*, pages 25–47, New York, 1980. Proceedings of the International Symposium, Freiburg, Academic Press.

[Han92] E. Hansen. *Global Optimization Using Interval Analysis*. Marcel Dekker, Inc., New York, 1992.

[Hav78] C.A. Haverly. Studies of the behaviour of recursion for the pooling problem. *ACM SIGMAP Bulletin*, 25:19–28, 1978.

[HDT88] R. Horst, J. Devries, and N. V. Thoai. On finding new vertices and redundant constraints in cutting plane algorithms for global optimization. *Operations Research Letters*, 7(2):85–90, April 1988.

[HH00] Masayuki Hirafuji and Scott Hagan. A global optimization algorithm based on the process of evolution in complex biological systems. *Computers and Electronics in Agriculture*, 29:125–134, 2000.

[HJL88] P. Hansen, B. Jaumard, and S. H. Lu. Analytical approach to global optimization. *Comptes Rendus de L'Academie Des Sciences Serie I-Mathematique*, 306(1):29–32, January 1988.

[Hor86] R. Horst. A general-class of branch-and-bound methods in global optimization with some new approaches for concave minimization. *Journal of Optimization Theory and Applications*, 51(2):271–291, November 1986.

[Hor88] R. Horst. Deterministic global optimization with partition sets whose feasibility is not known: Application to concave minimization, reverse convex constraints, d.c. programming and lipschitzian optimization. *Journal of Optimization Theory and Applications*, 58(1):11–37, July 1988.

[Hor89] R. Horst. On consistency of bounding operations in deterministic global optimization. *Journal of Optimization Theory and Applications*, 61(1):143–146, April 1989.

[HP95] R. Horst and P.M. Pardalos, editors. *Handbook of Global Optimization*, volume 1. Kluwer Academic Publishers, Dordrecht, 1995.

[HT96] R. Horst and Hoang Tuy. *Global Optimization: Deterministic Approaches*. Springer, Berlin, third edition, 1996.

[KB00] P. Kesavan and P.I. Barton. Generalized branch-and-cut framework for mixed-integer nonlinear optimization problems. *Computers & Chemical Engineering*, 24:1361–1366, 2000.

[KG88] G. R. Kocis and I. E. Grossmann. Global optimization of nonconvex mixed-integer nonlinear-programming (MINLP) problems in process synthesis. *Industrial & Engineering Chemistry Research*, 27(8):1407–1421, August 1988.

[KS04] S. Kucherenko and Yu. Sytsko. Application of deterministic low-discrepancy sequences in global optimization. *Computational Optimization and Applications*, 30(3):297–318, 2004.

[KVvA+99] V. Kovačević-Vujčić, M. Čangalović, M. Ašić, L. Ivanović, and M. Dražić. Tabu search methodology in global optimization. *Computers and Mathematics with Applications*, 37:125–133, 1999.

[Lag97] J.L. Lagrange. *Théorie des fonctions analytiques*. Impr. de la République, Paris, 1797.

[LB00] Y.H. Lee and B.J. Berne. Global optimization: Quantum thermal annealing with path integral monte carlo. *Journal of Physical Chemistry A*, 104:86–95, 2000.

[Lee56] J. Leech. The problem of the thirteen spheres. *Mathematical Gazette*, 40:22–23, 1956.

[Lib04a] L. Liberti. Reduction constraints for the global optimization of NLPs. *International Transactions in Operational Research*, 11(1):34–41, 2004.

[Lib04b] L. Liberti. *Reformulation and Convex Relaxation Techniques for Global Optimization*. PhD thesis, Imperial College London, UK, March 2004.

[Lib05] L. Liberti. Linearity embedded in nonconvex programs. *Journal of Global Optimization*, 33(2):157–196, 2005.

[LJ73] R. Luus and T.H.I. Jaakola. Optimization by direct search and systematic reduction of the size of the search region. *AIChE Journal*, 19:760–766, 1973.

[LMK04] L. Liberti, N. Maculan, and S. Kucherenko. The kissing number problem: a new result from global optimization. In L. Liberti and F. Maffioli, editors, *CTW04 Workshop on Graphs and Combinatorial Optimization*, volume 17 of *Electronic Notes in Discrete Mathematics*, pages 203–207, Amsterdam, 2004. Elsevier.

[Loc02] M. Locatelli. Simulated annealing algorithms for global optimization. In Pardalos and Romeijn [PR02], pages 179–229.

[LP03] L. Liberti and C.C. Pantelides. Convex envelopes of monomials of odd degree. *Journal of Global Optimization*, 25:157–168, 2003.

[LP06] 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.

[MBS80] S. F. Masri, G. A. Bekey, and F. B. Safford. A global optimization algorithm using adaptive random search. *Applied Mathematics and Computation*, 7(4):353–375, 1980.

[McC76] G.P. McCormick. Computability of global solutions to factorable nonconvex programs: Part I — Convex underestimating problems. *Mathematical Programming*, 10:146–175, 1976.

[Mil69] J. Milnor. *Topology from the differentiable viewpoint*. University Press of Virginia, Charlottesville, 1969.

[MKP+00] M. Mathur, S.B. Karale, S. Priye, V.K. Jayaraman, and B.D. Kulkarni. Ant colony approach to continuous function optimization. *Industrial and Engineering Chemistry Research*, 39:3814–3822, 2000.

[MM88] C. C. Meewella and D. Q. Mayne. An algorithm for global optimization of lipschitz continuous functions. *Journal of Optimization Theory and Applications*, 57(2):307–322, May 1988.

[MM89] C. C. Meewella and D. Q. Mayne. Efficient domain partitioning algorithms for global optimization of rational and lipschitz continuous functions. *Journal of Optimization Theory and Applications*, 61(2):247–270, May 1989.

[MPKVv03] N. Mladenović, J. Petrović, V. Kovačević-Vujčić, and M. Čangalović. Solving a spread-spectrum radar polyphase code design problem by tabu search and variable neighbourhood search. *European Journal of Operations Research*, 151:389–399, 2003.

[OBF01] A.R.F. O'Grady, I.D.L. Bogle, and E.S. Fraga. Interval analysis in automated design for bounded solutions. *Chemicke Zvesti*, 55(6):376–381, 2001.

[Par88] P.M. Pardalos. Enumerative techniques for solving some nonconvex global optimization problems. *Or Spektrum*, 10(1):29–35, February 1988.

[Par89] P.M. Pardalos. Parallel search algorithms in global optimization. *Applied Mathematics and Computation*, 29(3):219–229, February 1989.

[Pin86] J. Pinter. Global optimization on convex sets. *Or Spektrum*, 8(4):197–202, 1986.

[PR87] P.M. Pardalos and J.B. Rosen. *Constrained Global Optimization: Algorithms and Applications*. Springer, Berlin, 1987.

[PR02] P.M. Pardalos and H.E. Romeijn, editors. *Handbook of Global Optimization*, volume 2. Kluwer Academic Publishers, Dordrecht, 2002.

[PRT00] P.M. Pardalos, H.E. Romeijin, and H. Tuy. Recent development and trends in global optimization. *Journal of Computational and Applied Mathematics*, 124:209–228, 2000.

[PS88] P.M. Pardalos and G. Schnitger. Checking local optimality in constrained quadratic programming is np-hard. *Operations Research Letters*, 7(1):33–35, February 1988.

[PS98] C.H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover, New York, 1998.

[PTVF97] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C, Second Edition*. Cambridge University Press, Cambridge, 1992, reprinted 1997.

[Pug02] C.C. Pugh. *Real Mathematical Analysis*. Springer, Berlin, 2002.

[Raj00] S. Rajasekaran. On simulated annealing and nested annealing. *Journal of Global Optimization*, 16:43–56, 2000.

[Ree93] C.R. Reeves, editor. *Modern Heuristic Techniques for Combinatorial Problems*. Blackwell, Oxford, 1993.

[RR87] H. Ratschek and J. G. Rokne. Efficiency of a global optimization algorithm. *Siam Journal On Numerical Analysis*, 24(5):1191–1201, October 1987.

[RS95] H.S. Ryoo and N.V. Sahinidis. Global optimization of nonconvex NLPs and MINLPs with applications in process design. *Computers & Chemical Engineering*, 19(5):551–566, 1995.

[RS96] H. S. Ryoo and N. V. Sahinidis. A branch-and-reduce approach to global optimization. *Journal of Global Optimization*, 8(2):107–138, March 1996.

[RSC00] P. RoyChowdury, Y.P. Singh, and R.A. Chansarkar. Hybridization of gradient descent algorithms with dynamic tunneling methods for global optimization. *IEEE Transactions on Systems, Man and Cybernetics—Part A: Systems and Humans*, 30(3):384–390, 2000.

[SA92] H.D. Sherali and A. Alameddine. A new reformulation-linearization technique for bilinear programming problems. *Journal of Global Optimization*, 2:379–410, 1992.

[Sch02] F. Schoen. Two-phase methods for global optimization. In Pardalos and Romeijn [PR02], pages 151–177.

[Smi96] E.M.B. Smith. *On the Optimal Design of Continuous Processes*. PhD thesis, Imperial College of Science, Technology and Medicine, University of London, October 1996.

[SP97a] E.M.B. Smith and C.C. Pantelides. Global optimisation of nonconvex MINLPs. *Computers & Chemical Engineering*, 21:S791–S796, 1997.

[SP97b] R. Storn and K. Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359, 1997.

[SP99] E.M.B. Smith and C.C. Pantelides. A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex MINLPs. *Computers & Chemical Engineering*, 23:457–478, 1999.

[SS98] J.P. Shectman and N.V. Sahinidis. A finite algorithm for global minimization of separable concave programs. *Journal of Global Optimization*, 12:1–36, 1998.

[SSSWD00] G. Schrimpf, J. Schneider, H. Stamm-Wilbrandt, and G. Dueck. Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, 159:139–171, 2000.

[TH88] H. Tuy and R. Horst. Convergence and restart in branch-and-bound algorithms for global optimization: Application to concave minimization and d.c. optimization problems. *Mathematical Programming*, 41(2):161–183, July 1988.

[TSed] M. Tawarmalani and N.V. Sahinidis. Convexification and global optimization of the pooling problem. *Mathematical Programming*, (submitted).

[Tu89] A. Törn and A. Žilinskas. *Global Optimization*. Springer, Berlin, 1989.

[Tuy98] H. Tuy. *Convex Analysis and Global Optimization*. Kluwer Academic Publishers, Dordrecht, 1998.

[uB03] J. Žilinskas and I.D.L. Bogle. Evaluation ranges of functions using balanced random interval arithmetic. *Informatica*, 14(3):403–416, 2003.

[Ž85] A. Žilinskas. Axiomatic characterization of a global optimization algorithm and investigation of its search strategy. *Operations Research Letters*, 4(1):35–39, 1985.

[VEH96] R. Vaidyanathan and M. El-Halwagi. Global optimization of nonconvex MINLPs by interval analysis. In Grossmann [Gro96], pages 175–193.

[VF96] V. Visweswaran and C. A. Floudas. New formulations and branching strategies for the gop algorithm. In Grossmann [Gro96].

[WP02] T. Westerlund and R. Pörn. Solving pseudo-convex mixed integer optimization problems by cutting plane techniques. *Optimization and Engineering*, 3:235–280, 2002.

[WSHP98] T. Westerlund, H. Skrifvars, I. Harjunkoski, and R. Pörn. An extended cutting plane method for a class of non-convex MINLP problems. *Computers & Chemical Engineering*, 22(3):357–365, 1998.

[WW99] B.W. Wah and T. Wang. Efficient and adaptive lagrange-multiplier methods for nonlinear continuous global optimization. *Journal of Global Optimization*, 14:1:25, 1999.

[Yao89] Y. Yao. Dynamic tunnelling algorithm for global optimization. *IEEE Transactions On Systems Man and Cybernetics*, 19(5):1222–1230, sep-oct 1989.

[ZG99] J. M. Zamora and I. E. Grossmann. A branch and contract algorithm for problems with concave univariate, bilinear and linear fractional terms. *Journal of Global Optimization*, 14:217:249, 1999.

Index