Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers & Operations Research 32 (2005) 2419–2434
computers & operations research
www.elsevier.com/locate/dsw

Reformulation descent applied to circle packing problems

Nenad Mladenović[a,b,*], Frank Plastria[c], Dragan Urošević[a]

[a] *Mathematical Institute, Serbian Academy of Sciences, Knez Mihajlova 35, Belgrade 11000, Serbia, Montenegro, Yugoslavia*
[b] *GERAD and HEC Montreal, 3000 ch. de la Cote-Sainte-Catherine, Montreal H3T 2A7, Canada*
[c] *Vrije Universiteit Brussel, Pleinlaan 2, Brussel B-1050, Belgium*

Abstract

Several years ago classical Euclidean geometry problems of densest packing of circles in the plane have been formulated as nonconvex optimization problems, allowing to find heuristic solutions by using any available NLP solver. In this paper we try to improve this procedure. The faster NLP solvers use first order information only, so stop in a stationary point. A simple switch from Cartesian coordinates to polar or vice versa, may destroy this stationarity and allow the solver to descend further. Such formulation switches may of course be iterated. For densest packing of equal circles into a unit circle, this simple feature turns out to yield results close to the best known, while beating second order methods by a time-factor well over 100.

This technique is formalized as a general reformulation descent (RD) heuristic, which iterates among several formulations of the same problem until local searches obtain no further improvement. We also briefly discuss how RD might be used within other metaheuristic schemes.
© 2004 Elsevier Ltd. All rights reserved.

Keywords: Global optimization; Circle packing; Metaheuristics; Reformulation descent; Minos

1. Introduction

What is the smallest circle in which one can pack n unit circles? Chapter 10 of [1] about tangent circles reproduces the solutions given by [2] for $n = 2, \ldots, 10$, claiming their optimality, as allegedly proven by [3].

Their solution for $n = 10$ is shown at the left of Fig. 1, but the better solution obtained in [4] depicted on the right shows that the previous claim was incorrect. Since then this latter packing has been shown to be optimal, see [5].

* Correponding author. Fax: +381-11-186-105.
E-mail addresses: nenad@crt.umontreal.ca (N. Mladenović), frank.plastria@vub.ac.be (F. Plastria), draganu@mi.sanu.ac.yu (D. Urošević).

0305-0548/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2004.03.010

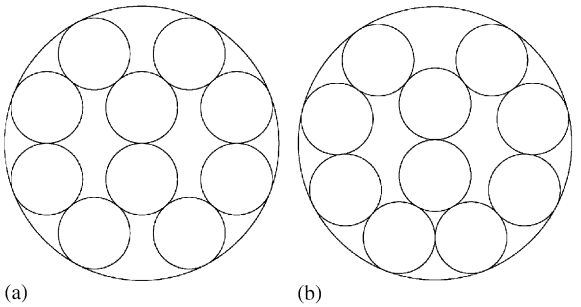

Fig. 1. Packing 10 unit circles into a circle. (a) Radius: 3.8284271; (b) Radius: 3.8130256.

In fact the solution on the left is not even a local optimum: moving all circles downwards along the boundary, while pushing the two central circles upwards, one may see that the radius of the enclosing circle may be continuously decreased, until the situation shown at the right is reached. However, this descent path is curved with respect to the eight outer unit circles, and there exists no fully linear path out of the situation at the left along which the outer radius may be decreased. Therefore, this solution corresponds to a stationary point, i.e. where no linear descent direction exists. Since (sub)gradients are defined using information along half-lines issued from the point of evaluation only, any optimization code based solely on gradient-type information will be blocked there.

However, along this descent curve all small circle centers either move radially or circularly with respect to the large circle's center, whereas the large radius reduction is a radial decrease with respect to this same center. All these movements are linear when expressed in polar coordinates centered at the outer circle's center. So in terms of polar coordinates there exists a linear path along which the objective decreases. In other words, in polar coordinate terms the solution (a) at the left is not stationary, and gradient-based optimization codes ought to be able to move on towards the right-hand solution (b).

This observation led to the idea of using nonlinear reformulations of a problem, in order to try to move on after a standard optimization code stops at a stationary point, but without the guarantee of having reached a local minimum. This paper reports on the first results obtained with this idea when tested on the Problem of Packing Circles into the unit Circle (PPCC), similar to Fig. 1. In these problems, each time a stationary point is reached, the problem is transformed non linearly from one coordinate system to the other; that is, from Cartesian to Polar or from Polar to Cartesian.

The paper is organized as follows. In the next section, we describe our RD heuristic in the case of nonlinear programming. In Section 3, the RD heuristic is applied to the circle packing problem and computational results are reported. Section 4 gives some results obtained with RD for packing circles into a square, and suggests two general ways how RD may be used in more general meta-heuristic schemes. In a final section, we give some concluding remarks and suggestions for future work.

2. RD for nonlinear non-convex optimization

2.1. Stationarity vs. local optimality

A nonlinear programming problem (NLP) may be formulated as

$$\min_x \{f(x)|x \in D\}, \tag{1}$$

where f is a function that maps $\mathbb{R}^n$ to $\mathbb{R}$ and D is a feasible set, usually defined by a set of equality and/or inequality constraints. NLP is *non-convex* if $f(x)$ is a non convex function and/or D is not a convex set. NLP is unconstrained if $D = \mathbb{R}^n$; when $D \subsetneq \mathbb{R}^n$, a constrained optimization problem is defined. The usual way for solving constrained problems is to transform them first to an unconstrained form and to solve it using a technique that best takes advantage of the theoretical properties of this unconstrained form.

Simple gradient methods use gradients for the search, and stop when no further descent direction can be found using (approximate) gradients. Theoretically, the point is then *stationary*, i.e. with a zero gradient.

Any local minimum is necessarily a stationary point, at least when the objective is differentiable at that point. Indeed, any non-stationary point admits a nonzero gradient at this point, and in the negative gradient direction the function will then strictly decrease *along a* (short enough) *straight line segment*, showing that the point is not a local minimum.

The emphasized text above is the clue to the RD idea for global optimization: gradient techniques and stationarity only take into account what happens when one leaves the stationary point along straight line segments. This means that a stationary point which is not a local minimum, can always be left along a (short enough) *curve* while decreasing the objective, but not along a *straight* one.

Now, using a reformulation of the problem applying a linear (or rather affine) transformation changes line segments into line segments, so any stationary point remains stationary, and the gradient method will also remain stuck after transformation. But if we can find some nonlinear transformation, which rectifies (makes straight) some descent curve, then after this reformulation the point is not stationary anymore, and the gradient method can proceed to find better points.

2.2. Pseudo-code of RD

The following pseudo-code describes in more detail how we propose to exploit the availability of several formulations of the problem.

RD

(1) Construct at least two not linearly related formulations of the problem.
(2) Choose a starting solution x.
(3) Repeat over all formulations.
- Using an off-the-shelf nonlinear optimization code, find a stationary point x' starting from x.
- If x' is better than x, set $x := x'$ and restart the full loop at 3, i.e. start again from the first formulation.
- otherwise loop, i.e. change formulation (if such exists, otherwise move to (4)) and repeat.

(4) Store x as a possible local optimum, since x is a stationary point for all formulations.

MRD

(5) Restart at step 2, with another starting solution, or stop if some stopping criterium is met.

In order to obtain solutions of better quality, our RD method may be restarted several times, as indicated in step 5 above. We will call this variant Multistart-RD (MRD).

Stopping criteria for MRD will typically depend on factors such as calculation time, frequency of improved solutions, or quality of found solution. This quality may be measured either in terms of simply being satisfactory, or, in case quality bounds are available, in terms of being sufficiently close to optimal.

2.3. Parameters of RD

RD may be seen as a general heuristic for local optimization with several parameters, as shortly discussed below, and therefore allows many variants in implementation.

2.3.1. Set of formulations

The choice of the formulations of the problem is of course important. When only one formulation is available, RD simply reduces to a single local optimization step. When at least two formulations are available, the main loop in RD becomes non-trivial. Any two linearly related formulations will in principle have the same stationary points, so should not be considered both, since this would never lead to new solutions and thus be totally ineffective.

Also the RD strategy will not be of any use if (approximate) stationarity and (approximate) local optimality are equivalent. In convex problems this is well known to be the case. Therefore, as soon as one of the formulations used is convex (i.e. minimization of a convex function under convex constraints) RD becomes ineffective.

Therefore RD should be reserved for problems for which several non-linearly related non-convex formulations are available, but no convex formulation exists (or is known). On the other hand, it is well-known that unconstrained non-convex models, obtained by Lagrangian, interior or exterior point reformulations from constrained problems often exhibit numerous stationary points. Thus, RD should be successfully applied for constrained optimization problems.

It was observed higher that a reformulation may only be expected to be effective if it rectifies some descent curve at the stationary point. Therefore, the choice of formulations is quite crucial: any formulation used should be chosen carefully to reflect the problem's geometry sufficiently so as to rectify some new but natural curved feasible paths (hopefully of descent). In other words, one should first have a good understanding of the objective's behavior before choosing the set of formulations.

2.3.2. Initial solution(s)

The choice of the starting solution is well known to be important in nonlinear optimization, although it is quite difficult, if not impossible, to say exactly how. Usually, one cannot say much more than a vague "if the starting solution is feasible and 'close' to a local optimum, the method will converge to it or an even closer local optimum".

For the multi-start MRD, one should of course aim at diversity of the successive starting points. Also, most codes allow starting at non-feasible solutions, which may be another way of diversification. Two strategies may be considered: randomly chosen starting points (possibly guided for diversification) or grid-like chosen starting points.

2.3.3. Nonlinear optimization code

Many quite different nonlinear optimization methods exist, and it is notorious that their results may be of quite different quality. Observe that the NLP-code used does not need to contain elaborate features to evade stationary, non-locally optimal points, since this is exactly what RD aims at overcoming. In view of potentially numerous repetitions, the code should better be *simple* and *fast*.

Note also that codes that use second-order conditions, like Newton-type methods should not be and cannot be used with RD for solving nonlinear non-convex optimization problems, since they do not necessarily stop in a stationary point. RD methods try to reach a precision that second order condition methods already have, but with much less computational efforts, i.e. in much less CPU time. Hence, an important question when testing RD is how it compares with Newton-type methods. That is why in the computational results section of this paper we compare our RD with a truncated Newton method, called Spenbar.

2.3.4. Sequence of chosen formulations

The order in which the formulations are tried out in step (3) might be thought to be irrelevant since the final solution obtained in step (4) is a stationary point for all formulations. But there might be many such stationary points, and a different order might result in another one. In practice, this turns out to happen; even with two formulations the choice of which one to start with does to matter, as shown by the results described in Section 3.4. In case there are more formulations one might even try to optimize in advance the order of the formulations aiming at 'maximizing' the 'degree of nonlinearity' of the transformation of one to the next.

Notice that the full loop over all formulations in step (3) is restarted after each better solution found. This feature has turned out to be important, to make sure that each time a better solution is found *all* (other) formulations are tried out again before stopping.

3. Packing equal circles in the unit circle

We have tested the RD and MRD strategies on circle packing problems.

In circle packing a given number of circular disks must be positioned without any overlap within a given planar shape, the *container*. The packing is densest when no smaller copy of the container can contain the same circles. The search for a densest packing may be expressed in two ways: either the classical way, by fixing the radii of the disks to 1 and minimizing the size of the container, as was suggested in the introduction for packing 10 unit circles in a smallest containing circle, or by fixing the container size and maximizing the common radius of the packed disks. In what follows the last option has been taken, but both models are clearly equivalent (although the transformation from one into the other is nonlinear, in fact).

Our approach may be applied to circle packing problems into several different container shapes. Here we consider first in detail the problem of packing equal circles in the unit circle (PPCC), and in the next section we briefly discuss the square container case (PPCS). We first give the two formulations used, then discuss the descent NLP method we used for solving PPCC w.r.t. both formulations and also which second order NLP method we selected for comparative testing, and finally discuss our computational setup and results.

3.1. Formulations

As announced in the introduction we have considered the two following nonlinear programming formulations of PPCC.

3.1.1. Cartesian coordinates

The circular container is the unit radius circle with center (0,0). The disks to be packed within it are given by their centers (x_i, y_i) $(i=1,\ldots,n)$, and their common radius r, which is to be maximized. This may be formulated as

$$\begin{aligned} &\max r, \\ &(x_i - x_j)^2 + (y_i - y_j)^2 - 4r^2 \geqslant 0, \quad && 1 \leqslant j < i \leqslant n, \\ &x_i^2 + y_i^2 \leqslant (1 - r)^2, && 1 \leqslant i \leqslant n, \\ &r \geqslant 0, \quad x_i, y_i \in \mathbb{R}, && 1 \leqslant i \leqslant n. \end{aligned} \tag{2}$$

The first set of inequalities express that any two disks should be disjoint: the squared Euclidean distance between their centers must be at least $(2r)^2$. The second set state that the disks must fully lie within the unit circle. We have preferred this smoother quadratic form, rather than the more standard constraint

$$\sqrt{x_i^2 + y_i^2} + r \leqslant 1. \tag{3}$$

3.1.2. Polar coordinates

The circular container is centered at the pole and has unit radius. The disks to be packed within it are given by their centers at polar coordinates (ρ_i, ϕ_i) $(i = 1,\ldots,n)$, and their common radius r, which is to be maximized. This may be formulated as

$$\begin{aligned} &\max r, \\ &\rho_i^2 + \rho_j^2 - 4\rho_i\rho_j \cos(\phi_i - \phi_j) - 4r^2 \geqslant 0, \quad && 1 \leqslant j < i \leqslant n, \\ &\rho_i + r \leqslant 1, && 1 \leqslant i \leqslant n, \\ &\rho_i, r \geqslant 0, \quad \phi_i \in [0, 2\pi], && 1 \leqslant i \leqslant n. \end{aligned} \tag{4}$$

Note that, unlike the Cartesian formulation, the second constraint set, expressing inclusion of the disks inside the container, are now linear.

3.2. Off-the-shelf NLP solver

The local minimizer used in our RD method should stop when the Jacobian (or gradient vector) is sufficiently close to zero. The obtained solution will then be a stationary point. We used Minos ([6–8]), a quite popular method of this type, which we briefly describe below.

Minos is a software package designed to solve large-scale optimization problems expressed in the following form:

$$\min \quad f(x) + c^{\mathrm{T}}x + d^{\mathrm{T}}y, \tag{5}$$

$$\text{s.t.} \quad g(x) + A_1 y = b_1, \tag{6}$$

$$A_2 x + A_3 y = b_2, \tag{7}$$

$$\underline{b} \leqslant \begin{pmatrix} x \\ y \end{pmatrix} \leqslant \bar{b}, \tag{8}$$

where the vectors c, d, b_1, b_2, $\underline{b}$ and $\bar{b}$ and the matrices A_1, A_2, A_3 are constant, $f(x)$ is a smooth scalar function and $g(x)$ is a vector of smooth functions $g_i(x)$. x and y are called the nonlinear and the linear variables, respectively.

A sequence of linearly constrained NLP subproblems is obtained in the following way: the nonlinear functions $g_i(x)$ in (6) are replaced by their linear approximation at the current point x_k:

$$\tilde{g}(x, x_k) = g(x_k) + J(x_k)(x - x_k) \tag{9}$$

or shortly

$$\tilde{g} = g_k + J_k(x - x_k), \tag{10}$$

where k is an iteration counter and J is the Jacobian or gradient of g. Moreover, an augmented Lagrangian objective function is constructed instead of (5)

$$\min_{x,y} f(x) + c^{\mathrm{T}}x + d^{\mathrm{T}}y - \lambda_k^{\mathrm{T}}(g - \tilde{g}) + \frac{1}{2}\rho(g - \tilde{g})^{\mathrm{T}}(g - \tilde{g}). \tag{11}$$

The vector λ_k is an estimate of λ, the Lagrange multipliers for the nonlinear constraints. The scalar ρ is a penalty parameter. Therefore, using (10) we see that the linearized constraints (6) and (7) take a form

$$\begin{pmatrix} J_k & A_1 \\ A_2 & A_3 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} = \begin{pmatrix} J_k x_k - g_k \\ 0 \end{pmatrix}, \tag{12}$$

where s_1 and s_2 are new slack variables. For solving (11), (12) and (8), Minos uses the reduced gradient algorithm (see e.g. [9]).

Table 1 shows our first experiments for $n = 35$ circles when solving PPCC using RD with Minos as off-the-shelf code. Minos requires setting a limit to its number of internal iterations, and for use in RD we always set it to 60, after some experimentation. Note that this means that some Minos steps may be stopped before a stationary point or local minimum is reached.

For all 15 trials, always starting from a random solution with zero radius and the Cartesian formulation, the first solution obtained was improved when moving to polar coordinates. Then, in all cases, the obtained solutions were improved again after moving back to Cartesian coordinates; after which in all but 2 cases the local searches in polar coordinates were successful again, etc. This example was encouraging in the sense that reformulation turned out to be useful in many cases. Moreover, a solution equivalent to the best known was obtained at our 10th trial. This confirmed the

Table 1
Packing $n = 35$ equal circles in the unit circle by RD; best known $r = 0.1493167765$ [11]

	Radius							% dev.	Aver. impr.	Run. time
	Iter. 1	Iter. 2	Iter. 3	Iter. 4	Iter. 5	Iter. 6	Iter. 7			
1	0.13768	0.14379	0.14795	0.14897	0.14898			0.23	2.01	59.33
2	0.13470	0.14601	0.14707	0.14896				0.24	3.47	48.6
3	0.14090	0.14660	0.14767	0.14771				1.08	1.60	47.95
4	0.09958	0.13051	0.13889	0.14766	0.14883	0.14921		0.07	8.97	73.12
5	0.13808	0.14662	0.14880	0.14922				0.06	2.65	51.16
6	0.14017	0.14666	0.14860					0.48	2.98	37.63
7	0.13136	0.14551	0.14679	0.14725				1.38	3.99	50.65
8	0.13102	0.14204	0.14573	0.14648	0.14682			1.67	2.94	63.05
9	0.14273	0.14750	0.14759					1.16	1.70	37.03
10	0.11201	0.11430	0.11701	0.12079	0.12152	0.14882	0.14931	0.00	2.06	50.89
11	0.13536	0.14081	0.14484	0.14723	0.14822	0.14844		0.59	2.30	74.08
12	0.13390	0.14383	0.14578	0.14579				2.36	2.93	47.77
13	0.12898	0.13902	0.14344	0.14706	0.14772	0.14787		0.97	3.48	74.95
14	0.13413	0.14613	0.14722	0.14776	0.14814	0.14835	0.14840	0.61	2.58	83.47
15	0.14094	0.14720	0.14821	0.14849				0.55	1.77	48.89
Average								0.77	3.17	56.57

conclusion of [4] that use of an off-the-shelf NLP code can lead to good results without resorting to sophisticated special purpose search algorithms. However, the solution was obtained after the 7th iteration only, showing that without the RD strategy we would probably not have found it. The behavior of our RD for $n = 35$ during this 10th trial is illustrated in Fig. 2. Note the early stops of Minos during the first five steps due to the 60 inner iteration limit.

We also observed that any time Minos stopped with the message

> "*Alternative optimum possible*. The slack is nonbasic, but its reduced gradient is essentially zero. This means that if the slack were allowed to start moving away from its bound, there would be no change in the value of the objective function."

the next reformulation step of RD turned out to be able to obtain a further improvement. This message therefore seems to indicate a stationary point of (11) which is not a local optimum.

3.3. Choice of competing NLP solver

Some constrained NLP methods do not stop when a stationary point is reached. Such methods use second-order conditions or the Hessian matrix H (or its approximation), and they stop in x only when $H(x)$ is positive definite. The best-known NLP method of that kind is the Newton method.

Thus a natural challenging question is how our RD heuristic compares with some Newton type method. The Spenbar (Spare penalty-barrier) software [10] implements the penalty-barrier method.

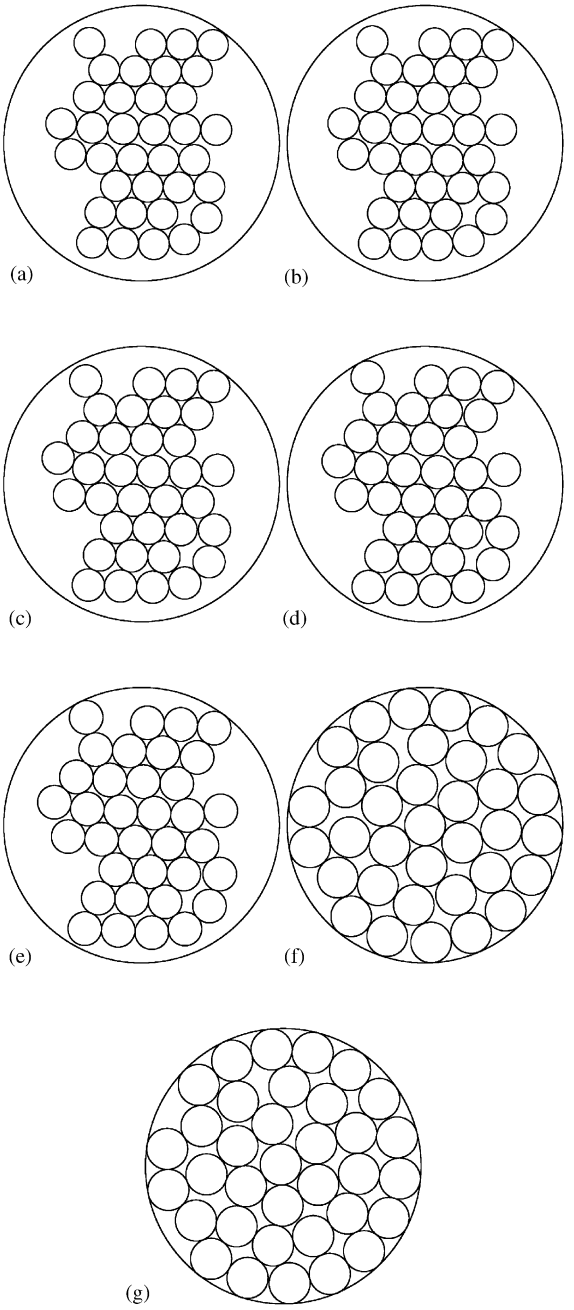

Fig. 2. Iterations of RD in solving PPCC for $n = 35$. (a) $r_1 = 0.112010$ (*Cartesian*); (b) $r_2 = 0.114301$ (*Polar*); (c) $r_3 = 0.117013$ (*Cartesian*); (d) $r_4 = 0.120787$ (*Polar*); (e) $r_5 = 0.121525$ (*Cartesian*); (f) $r_6 = 0.148819$ (*Polar*); (g) $r_7 = 0.149310$ (*Cartesian*).

In Section 3.4.3 we compare RD with Spenbar. For the sake of completeness, we briefly describe Spenbar in Appendix A. Note again that we do not use Spenbar in our RD, since it is a second order condition method and hence, does not always stop when a stationary point is reached.

3.4. Computational results

In this subsection we first examine the influence of two parameters of RD on the quality of the final objective function value: the initial solutions and the order of formulations. Then we compare the performance of our RD with some single formulation techniques, as well as with the Spenbar method.

All tests in this paper were performed on a Pentium 4, 1800 MHz with 1024 Mb RAM. For each problem instance (n-value), 50 different initial solutions are generated, from which each method is started. Traditionally circle packing is studied as packing unit disks and results are expressed in terms of the radius of the smallest container, i.e. in terms of our $1/r$. Therefore, we present our results also in the same terms. The '% deviation of the best' is calculated as

$$\frac{f_{\text{method}} - f_{\text{bestknown}}}{f_{\text{bestknown}}} \times 100,$$

where $f_{\text{bestknown}}$ denotes the best known value of $1/r$ from the literature [11], and f_{method} similarly denotes the inverse of the best objective function value obtained during the 50 trials by the tested method. The value '% deviation of the average' is obtained similarly, but now f_{method} denotes the average inverse objective function value obtained during the 50 trials by the tested method.

3.4.1. Effect of the initial solution

The influence of three different initialization methods on the quality of the final solutions obtained by RD have been tested.

The simplest method (I_1) worked as follows: the polar coordinates of the circle centers ρ_i and ϕ_i, $i=1,\ldots,n$ were chosen at random, uniformly from $[0, 0.95]$ and, $[0, 2\pi]$, respectively. This, however, yields a high concentration of centers around the origin. In the second initialization method (I_2) the circle centers were uniformly distributed within a circle of radius $(1 - 1/\sqrt{n})$; this is achieved by taking ϕ_i uniform in $[0, 2\pi]$ and $\rho_i = (1 - 1/\sqrt{n})\sqrt{rnd}$, where rnd is uniformly distributed on (0,1). The third method (I_3) consisted of the following steps: build n unit squares as much as possible symmetrically around origin; generate Cartesian coordinates (x_i, y_i) uniformly from $[-0.25, 0.25]$ around the center of each square; reduce all coordinates proportionally, assuming that the big circle has a unit radius.

In all three initialization methods the initial radius r was obtained as half of the smallest pairwise distance between circle centers.

Some results are given in Table 2. It appears that the initialization does not influence the best final solution much among 50 random initial solutions, with a slight advantage to I_1, although I_3 seems to give more stable results and uses much less time.

3.4.2. Effect of the formulation order

We also investigated how the order of the two formulations influences the final solutions obtained by RD, i.e. start with Cartesian coordinates followed by polar (C-P for short) or the other way

Table 2
Effect of the initialization method on RD solution

n	% dev. of best			% dev. of av.			Running time (s)		
	I_1	I_2	I_3	I_1	I_2	I_3	I_1	I_2	I_3
40	0.00	0.00	0.00	3.47	1.35	0.97	1.51	0.76	1.06
45	0.16	0.04	0.10	3.01	0.84	0.69	3.00	1.87	1.40
50	0.00	0.01	0.06	0.93	2.87	0.79	5.15	3.51	2.71
55	0.00	0.17	0.00	2.24	2.36	2.09	9.17	5.69	3.54
60	0.00	0.00	0.03	3.49	1.03	1.40	12.96	7.80	5.09
65	0.00	0.00	0.00	1.01	1.07	1.33	18.91	11.00	16.09
70	0.10	0.22	0.22	1.49	1.03	0.99	27.21	18.10	19.78
75	0.02	0.12	0.10	0.85	0.89	0.77	44.60	30.71	27.20
Avg.	0.03	0.07	0.06	2.06	1.43	1.13	15.31	9.93	9.61

Table 3
Effect of different formulation order on RD solution

n	% deviation of the best		% dev. of aver. (50)		Running time (s)	
	C-P	P-C	C-P	P-C	C-P	P-C
10	0.00	0.57	1.03	0.91	0.00	0.00
15	0.13	0.13	0.49	0.76	0.01	0.01
20	0.00	0.00	1.15	0.99	0.04	0.02
25	0.00	0.00	0.62	0.70	0.08	0.07
30	0.00	0.00	0.97	1.17	0.16	0.14
35	0.00	0.01	0.73	0.60	0.90	0.41
40	0.00	0.00	0.97	1.36	1.11	0.87
45	0.10	0.25	0.69	0.78	1.47	1.32
50	0.06	0.00	0.79	0.85	3.19	1.72
55	0.00	1.15	2.09	2.08	3.37	2.10
60	0.03	0.44	1.40	1.52	4.71	3.29
65	0.00	0.01	1.33	1.33	16.24	6.82
70	0.22	0.26	0.99	0.84	19.56	9.73
75	0.10	0.26	0.77	0.72	26.46	11.93
80	0.10	0.10	0.93	0.93	39.15	18.74
85	0.72	0.96	1.75	1.89	38.79	20.01
90	0.02	0.02	1.27	1.06	96.82	49.29
95	0.18	0.18	0.93	0.96	147.35	71.15
100	0.30	0.38	1.01	1.00	180.32	83.14
Av.	0.10	0.25	1.05	1.08	30.51	14.78

around (P-C for short). Our results are given in Table 3. It appears that slightly better results are obtained in terms of solution quality when implementing the C-P order, but in twice the time needed for the P-C order.

Table 4
Comparison of $1/r$ values for: RD—reformulation descent; M_C—Minos with Cartesian formulation; M_P—Minos with polar formulation, SP—Spenbar

n	Best known	% dev. of best				% dev. of average				Average running time (s)			
		RD	M_C	M_P	SP	RD	M_C	M_P	SP	RD	M_C	M_P	SP
10	3.813026	0.00	0.00	0.00	0.00	1.03	2.01	0.88	0.08	0.00	0.02	0.01	0.29
15	4.521357	0.00	0.13	0.13	0.00	0.49	0.65	0.77	0.89	0.01	0.03	0.02	1.87
20	5.122307	0.00	0.00	0.00	0.00	1.15	2.80	2.49	0.33	0.04	0.11	0.08	5.21
25	5.752824	0.00	0.00	0.00	0.00	0.62	5.07	3.21	0.44	0.08	0.37	0.19	17.14
30	6.197741	0.00	0.00	0.00	0.00	0.97	2.49	1.40	0.71	0.16	0.52	0.29	41.69
35	6.697171	0.00	0.01	0.02	0.03	0.73	12.27	2.17	0.45	0.90	1.84	1.73	81.98
40	7.123847	0.00	0.00	0.00	0.00	0.97	9.36	4.21	0.58	1.11	2.92	1.91	179.69
45	7.572912	0.10	0.11	0.04	0.07	0.69	3.75	2.31	0.32	1.47	3.08	2.19	300.41
50	7.947515	0.06	0.03	0.00	0.02	0.79	6.90	4.26	0.39	3.19	5.16	4.41	503.78
55	8.211102	0.00	1.13	1.57	1.56	2.09	4.80	2.40	1.93	3.37	6.73	5.15	902.59
60	8.646220	0.03	0.10	0.57	0.00	1.40	1.58	1.78	0.45	4.71	7.54	6.00	1526.40
65	9.017397	0.00	0.47	0.44	0.31	1.33	5.86	2.79	0.39	16.24	12.94	10.43	2118.60
70	9.346660	0.10	0.55	0.32	0.27	0.99	7.83	2.15	0.67	19.56	17.61	14.54	3484.63
Av.		0.02	0.19	0.24	0.17	1.02	5.03	2.37	0.59	3.91	4.53	3.61	704.94
75	9.678344	0.10	0.22	0.44		0.77	4.56	1.69		26.46	22.67	17.16	
80	9.970588	0.10	0.41	0.29		0.93	3.38	1.69		39.15	30.99	23.62	
85	10.163112	0.72	1.43	1.10		1.75	3.31	1.90		38.79	29.85	24.04	
90	10.546069	0.02	0.02	0.45		1.27	10.59	4.32		96.82	47.19	47.70	
95	10.840205	0.18	0.26	0.48		0.93	11.55	6.87		147.35	59.51	41.84	
100	11.082528	0.30	0.52	0.38		1.01	8.39	3.39		180.32	64.96	45.02	
Average		0.08	0.28	0.32		1.05	5.61	2.65		29.18	15.93	12.50	

3.4.3. Comparison with single formulation methods

Now we compare the performance of RD+Minos with three standard codes applied to a single formulation: Minos with the Cartesian formulation (M_C); Minos with the polar formulation (M_P) and Spenbar (SP) with the Cartesian formulation. The purpose of comparing RD with M_C and M_P is to study the advantage of using two instead of only one formulation, always using the same NLP-solver, Minos. Spenbar is included in order to see how RD compares with more sophisticated methods that use second order information.

Initial solutions were generated by the method I_3 described before. As before in RD the limit on the number of inner iterations in Minos was set to 60, while for M_C and M_P that number was set to 100, in order to give them more chance to approximate a stationary point. Higher iteration limits (we went up to 1000) turned out not to have much effect on the objective values, but unnecessarily increased the calculation times, thus leading to a rather unfair time-comparison.

To save space we compare the four methods in Table 4 only for $n=10,15,\ldots,100$. More complete results may be found in [12].

It appears that: (i) solutions obtained by our RD are of better quality than those obtained by the single formulation methods M_C and M_P; (ii) solutions obtained by RD and Spenbar are of similar quality, but RD is about 150 times faster; (iii) problems larger than $n = 70$ cannot be solved by Spenbar in reasonable time (within 1 h);

In our more detailed experiments [12] with numbers of circles $n = 10, 11, \ldots, 100$, RD found the best-known results from the literature in 40% of the cases, while in all other cases the error never exceeded 1%.

4. Possible extensions

We briefly discuss here the use of RD for a slightly different problem, and indicate some possible extensions of RD.

4.1. Packing circles into a square

Replacing the second sets of constraints in (2) and (4), respectively, by $r \leqslant x_i \leqslant 1 - r$, $r \leqslant y_i \leqslant 1 - r$ and $r \leqslant \rho_i \cos \phi_i \leqslant 1 - r$, $r \leqslant \rho_i \sin \phi_i \leqslant 1 - r$ we obtain two formulations for packing equal circles into the unit square (PPCS). We also compared RD, M_C and M_P for these problems. Our results are given in Table 5. As before, average values are obtained after 50 runs with each method.

As was to be expected, RD did not show a clear advantage over single formulation methods. Contrary to PPCC the polar coordinate formulation is much less efficient in PPCS. Indeed, circular movements are not very useful anymore at the container boundary, so a full polar coordinate formulation is not so suited. However, in average, RD still outperforms both single formulation methods on average: compare the 3.04% deviation of RD with the 6.33% and 13.73% of M_C and M_P, respectively.

4.2. RD within metaheuristics

By using 50 trials we in fact applied a Multi-start RD for solving PPCC and PPCS. However, it is well-known that random multi-start local search methods suffer from the so-called *center-limit-catastrophe* [13], and we found out that MRD is no exception: solutions obtained by simple multiple random starts remain mostly of average quality (see e.g., [14–16]).

Several meta-heuristic methods such as tabu search, memetic search, or variable neighborhood search, suggest other more efficient ways to use local search procedures repeatedly (see the recent survey of meta-heuristic methods in [17]). RD might thus also be used within VNS [18] or some other meta-heuristic as a local search routine. This avenue was not pursued here.

4.3. Formulation space and reformulation search

Another natural extension of RD, is to use reformulations at the upper level within a meta-heuristic scheme as follows.

Table 5
Problem of packing circles in unit square

n	Best known	% dev. of best			% dev. of average			Av. run. time (s)		
		RD	M_C	M_P	RD	M_C	M_P	RD	M_C	M_P
10	6.74757140	0.00	0.00	0.00	1.88	2.96	3.75	0.01	0.01	0.02
15	7.86370315	0.54	0.54	0.00	1.69	3.12	5.38	0.03	0.04	0.05
20	8.97808315	0.00	1.56	0.00	3.89	5.84	9.11	0.05	0.11	0.10
25	10.00000000	0.00	0.00	0.00	3.30	3.75	14.09	0.10	0.24	0.28
30	10.90856809	0.63	0.63	0.58	3.22	4.37	11.79	0.26	0.61	0.57
35	11.86370360	0.32	0.32	0.59	2.11	2.43	8.98	0.38	1.04	1.11
40	12.62837533	0.09	0.09	0.19	1.75	5.79	12.32	1.10	1.94	1.97
45	13.38198309	0.16	0.16	0.11	1.49	3.20	11.58	1.24	2.26	2.54
50	14.01009567	0.28	1.04	0.28	1.82	2.90	12.58	1.87	4.00	3.64
55	14.69391977	0.61	0.61	0.37	1.81	5.49	11.56	3.27	6.15	5.22
60	15.37742112	0.38	0.38	0.53	3.02	7.38	15.52	5.17	8.11	7.13
65	15.82179344	0.93	0.93	1.09	3.94	5.88	20.69	7.50	12.26	10.04
70	16.50255154	0.36	0.92	0.80	3.81	9.02	18.91	13.43	13.12	11.92
75	17.09561268	0.67	0.73	0.55	1.51	7.96	18.72	17.01	18.04	15.37
80	17.43050631	1.45	1.50	0.77	6.74	14.03	20.13	24.95	23.65	23.37
85	17.96028299	1.39	1.23	1.05	5.03	8.83	18.27	33.11	30.44	26.05
90	18.60466847	0.77	1.25	1.13	1.91	11.12	18.68	43.62	35.85	27.81
95	19.07658639	0.80	0.94	0.49	5.79	12.99	13.73	51.02	43.49	35.48
100	20.00000000	0.00	0.00	0.00	3.08	3.26	15.04	80.80	61.15	47.68
Average		0.49	0.68	0.45	3.04	6.33	13.73	15.00	13.82	11.60

Global and combinatorial optimization methods perform a search through the solution space $\mathscr{S}$ for a fixed formulation. Local search methods are based on neighborhoods $\mathscr{N}(s) \subseteq \mathscr{S}$ of a solution $s \in \mathscr{S}$.

RD suggests the introduction of a *formulation space* $\mathscr{F}$ as well: it is a set of different formulations of the same problem. In order to perform a search through the formulation space $\mathscr{F}$, one first equips $\mathscr{F}$ with some distance function, which defines neighborhoods $\{\mathscr{N}_\ell(\phi) \mid \ell = 1, \ldots, \ell_{\max}\}$ of a formulation $\phi \in \mathscr{F}$. Also a formulation ϕ_1 is considered as better at x than a formulation ϕ_2 if the solution obtained starting from x by using ϕ_1 is better than that obtained by using ϕ_2. All known successful ideas for search through $\mathscr{S}$ may then also be applied in *Reformulation Search* methods, by searching in $\mathscr{F}$ for a better formulation at the current solution, and using it to carry on the search through $\mathscr{S}$.

For PPCC, for example, one might consider the set $\mathscr{F}$ of all mixed formulations, in which some circle centers are given in Cartesian coordinates while the others are given in polar coordinates. Distance between two formulations is then the number of centers whose coordinates are expressed in different systems in each formulation. The neighborhoods $\mathscr{N}_\ell(\phi)$ ($\ell = 1, \ldots, \ell_{\max} \leqslant n$) of formulation ϕ are then defined as those formulations differing from ϕ at no more than ℓ circle centers.

5. Conclusions

In this paper we have introduced a simple near-optimal solution method for solving a nonlinear global optimization problem. It explores the fact that a point which is stationary w.r.t. one formulation is not necessarily so with another. Therefore, our method, called Reformulation Descent (RD), alternates between several formulations using a fast NLP code that stops in a stationary point.

The proposed RD heuristic has been tested by solving two circle packing problems, where two formulations, in Cartesian and polar coordinates, respectively, are used. Minos was used for finding a stationary point. Computer results for RD compare favorably with single formulation methods. When compared to a Truncated Newton method results were similar in terms of solution quality, but RD was about 150 times faster on average.

It remains for future work to try out using RD to solve other packing problems, as well as NLP non-convex test problems from the literature. Furthermore, the idea of Reformulation Descent, may be applied in any other situation, as soon as a same problem may be formulated in several quite different ways. Opportunities are now sought in combinatorial optimization. It also remains for future work to investigate the use of RD within some metaheuristic schemes, as suggested in Section 4.

Acknowledgements

The first and third authors are partly supported by the Serbian Ministry of Sciences, Project # 1583. Part of this work was done while the second author was visiting professor at FUNDP, Namur, Belgium, whose support is gratefully acknowledged. We also thank the referee whose comments improved the presentation of the paper.

Appendix A.

Using Spenbar. Let us consider the constrained NLP problem

$$\min \quad f(x) \tag{A.1}$$

$$\text{s.t.} \quad g_i(x) \geqslant 0, \quad i = 1, \ldots, m, \tag{A.2}$$

$$\underline{b}_j \leqslant x_j \leqslant \bar{b}_j, \quad j = 1, \ldots, n, \tag{A.3}$$

where $x \in \mathbb{R}^n$, and the functions f and g_i $(i = 1, \ldots, n)$ are continuous on $\mathbb{R}^n$. In the general form nonlinear equality constraints are also considered in the model, but we omit these since they are not present in PPCC and PPCS. Note also that, unlike in Minos, no distinction is made between linear and nonlinear variables or constraints. Spenbar (see the latest version in [10]) implements the penalty-barrier method, whose main idea is to construct a sequence of unconstrained minimization subproblems as follows [19]:

$$\min \quad F(x, \mu, \lambda) = f(x) - \mu \sum_{i=1}^{m} \lambda_i \log\left(1 + \frac{g_i(x)}{\mu}\right) \tag{A.4}$$

$$\text{s.t.} \quad \underline{b}_j \leqslant x_j \leqslant b_j, \quad j = 1, \ldots, n, \tag{A.5}$$

where the λ_i $(i = 1, \ldots, m)$ are nonnegative estimates of the Lagrange multipliers associated with the inequality constraints, and $\mu > 0$ is the barrier parameter. The solution (x_k, λ_k, μ_k) of the subproblem in iteration k is considered to be an initial point of the next iteration. Each subproblem is solved by means of a truncated Newton method with simple bounds, as implemented in subroutine TN by S. Nash.

References

[1] Gardner M. Fractal music, hypercards and more …, (Mathematical recreations from Scientific American). New York: W.H. Freeman and Co.; 1992. p. 156, Fig. 68.

[2] Kravitz S. Packing cylinders into cylindrical containers. Mathematics Magazine 1967;40:65–70.

[3] Pirl U. Der Mindestabstand von n in der Einheitskreisscheibe gelegenen Punkten. Mathematishe Nachrichten 1969;40:111–24.

[4] Drezner Z, Erkut E. On the continuous p-dispersion problem. Journal of the Operational Research Society 1995;46:516–20.

[5] Graham RL, Lubaschevski BD, Nurmela KJÖ, stergård PR. Dense packings of congruent circles in a circle. Discrete Mathematics 1998;181:139–54.

[6] Minos, Stanford Business Software Inc., website http://www.sbsi-sol-optimize.com/products_minos5_5.htm

[7] Murtagh BA, Saunders MA. MINOS user's guide, Report SOL 77-9, Department of Operations Research, Stanford University, CA, 1997.

[8] Murtagh BA, Saunders MA. Large scale linearly constrained optimization. Mathematical Programming 1978;14: 41–72.

[9] Gill PE, Murray W, Wright MH. Practical optimization. London: Academic Press, Brace Jovanovich; 1981.

[10] Andrei N. Penalty-barrier algorithms for nonlinear optimization: preliminary computational results. Studies in Informatics and Control 1998;7:15–36.

[11] Specht E. The best known packings of equal circles in the unit circle (up to N = 500), website (last update used dated 28 oct 2002) http://hydra.nat.uni-magdeburg.de/packing/cci/cci.html

[12] Mladenović N, Plastria F, Urošević D. Reformulation descent for packing circles into the unit circle. Cahiers du GERAD, G-2003-68, Montreal, October 2003. http://www.gerad.ca/fichiers/cahiers/G-2003-68.pdf

[13] Baum EB. Toward practical 'neural' computation for combinatorial optimization problems. In: Denker J, editor. Neural networks for computing. Boston: American Institute of Physics; 1986.

[14] Boese KD, Kahng AB, Muddu S. A new adaptive multi-start technique for combinatorial global optimization. Operations Research Letters 1994;16:101–13.

[15] Hansen P, Mladenović N. Variable neighborhood search for the p-median. Location Sciences 1997;5:207–26.

[16] Hansen P, Mladenović N. Variable neighborhood search. In: Glover F, Kochenberger G, editors. Handbook of Metaheuristics. Dordrecht: Kluwer Academic Publisher; 2003. p. 145–84.

[17] Glover F, Kochenberger G, editor. Handbook of metaheuristics. Dordrecht: Kluwer Academic Publisher; 2003.

[18] Mladenović N, Hansen P. Variable neighborhood search. Computers and Operations Research 1997;24:1097–100.

[19] Polyak R. Modified barrier functions (theory and methods). Mathematical Programming 1992;54:177–222.