Solving the problem of packing equal and unequal circles in a circular container

Grosso A., Jamali A.R.M.J.U., Locatelli M., Schoen F.

March, 2008

Abstract

In this paper we propose a Monotonic Basin Hopping approach and its population-based variant Population Basin Hopping to solve the problem of packing equal and unequal circles within a circular container with minimum radius. Extensive computational experiments have been performed both to analyze the problem at hand, and to choose in an appropriate way the parameter values for the proposed methods. Different improvements with respect to the best results reported in the literature have been detected.

Keywords: Circle Packing, Monotonic Basin Hopping, Multistart

1 Introduction

In its most general formulation the packing problem can be defined as follows. Given a container which depends on a size parameter r and denoted by $C(r) \subset \mathbb{R}^d$, and given n geometrical objects $D_1, \ldots, D_n$, whose position in the d-dimensional space depends on t position parameters $\alpha_{i1}, \ldots, \alpha_{it}$, i.e., $D_i = D_i(\alpha_{i1}, \ldots, \alpha_{it}) \subset \mathbb{R}^d$, $i = 1, \ldots, n$, we would like to choose the parameters in such a way that all the objects are packed into the container without overlapping (the objects can at most "touch" each other) and the size of the container is minimized. More formally, the problem is the following

$$\begin{array}{lll} \min & r & \\ & D_i(\alpha_{i1}, \ldots, \alpha_{it}) \subseteq C(r) & i = 1, \ldots, n \\ & D_i^0(\alpha_{i1}, \ldots, \alpha_{it}) \cap D_j^0(\alpha_{j1}, \ldots, \alpha_{jt}) = \emptyset & i \neq j \end{array}$$

where D_i^0 denotes the interior of D_i.

A widely studied case is the one in $\mathbb{R}^2$ where the objects are *equal* or *unequal* circles of given radius and the container has some regular shape like a square, a circle or an equilateral triangle. In such cases we have: $t = 2$; the position parameters α_{i1} and α_{i2} correspond to the coordinates of the center of circle i; the size parameter r has different interpretations according to the shape of the

container (e.g., the side length for the square and the equilateral triangle, or the radius for the circle).
Much literature exists about the problem of packing equal circles in a square. This includes computer-aided optimality proofs [8, 18, 20], branch-and-bound approaches [16], and different heuristic approaches like, e.g., [2, 4, 5, 7, 21]. A survey about this problem can be found in [26] and a recent book has been dedicated to the subject [27].
In [24] the problem of placing circles with different sizes into a rectangular container with fixed width and minimum height is considered.
Different contributions exist in the literature also about the problem of placing equal or unequal circles in a circular container. Heuristic approaches for this problem have been proposed, e.g., in [1, 3, 11, 12, 13, 19, 23, 25, 29, 30].
Benchmark results for the problem of packing equal circles in a container whose shape is a square, a circle or an equilateral triangle are reported and continuously updated in E. Specht's web site [22]. Test instances for the problem of packing unequal circles in a circle can be found, e.g., in [11], but, in the authors' opinion, even more challenging instances have been proposed in the Circle Packing Contest (see `http://www.recmath.org/contest/CirclePacking/index.php` for details on the contest), where, given a positive integer n, competitors were asked to place n circles of radii respectively equal to $1, 2, \ldots, n$ inside a circular container with minimum radius.
Finally, we also refer to the paper [6] where a detailed survey about methods and applications of packing problems can be found.

In [2] we investigated the problem of packing equal circles in the unit square and proposed a quite successful method for such problem. The problem is in fact equivalent to that of minimizing the edge length of a square container into which we want to pack n equal circles with unit radius. In this paper we investigate a related problem: packing circles with unit radius into a circular container with minimum radius. Except for the shape of the container (there the unit square, here a circle) the two problems are quite similar and we do expect that the extension of the method employed in [2] also gives very good results for this problem (this is indeed confirmed by the computational results, see Section 4). But the aim of this paper is not merely to apply a method, proved to be successful for one problem, to a closely related one. The aim of the paper is also (and, actually, mainly) to perform a more detailed computational investigation both of the problem at hand and of the proposed method, in order to better understand how to choose its most relevant parameters.

We will also briefly investigate the problem of packing unequal circles in a circle. In spite of the similarity of this problem with the problem of packing equal circles, we will show that the obvious extension of the method proposed for the case of equal circles to the case of unequal ones will not be successful. The peculiarities of the problem with unequal circles (in particular, its combinatorial nature due to the different radii of the circles) has to be taken into account in order to define a successful method also for this case.

The paper is organized as follows. In Section 2 we will introduce so called Monotonic Basin Hopping approach. In Section 3 we will introduce a population-based approach strictly related to Monotonic Basin Hopping, called Population Basin Hopping. In Section 4 we will present different computational experiments aimed both at analyzing the problem at hand and at selecting the most appropriate values for the parameters on which the proposed methods depend. Finally, in Section 5 we will investigate the case of unequal circles.

2 Monotonic Basin Hopping

It is well known that the problem of packing equal or unequal circles in a circle can be reformulated as a mathematical programming one. Indeed, it can be stated as follows

$$\begin{aligned}
\min \quad & r && (1)\\
& \alpha_{i1}^2 + \alpha_{i2}^2 \le (r - r_i)^2 \qquad i = 1, \dots, n && (2)\\
& (\alpha_{i1} - \alpha_{j1})^2 + (\alpha_{i2} - \alpha_{j2})^2 \ge (r_i + r_j)^2 \quad i \ne j && (3)\\
& r \ge \max_{i=1,\dots,n} \; r_i && (4)
\end{aligned}$$

where r_i denotes the radius of circle i (in the case of equal circles we have $r_i = 1$ for all i). Constraints (2) guarantee that each circle is within the container, while constraints (3) guarantee that circles do not overlap (constraint (4) is an obvious lower bound for the radius of the container).

In what follows we will restrict our attention to the case of equal circles. We will get back to the case of unequal circles in Section 5. The problem of packing equal circles, i.e., problem (1)-(4) with $r_i = 1$ for all i, turns out to be a global optimization one. The number of local minimizers tends to increase quite quickly with the number n of circles (see the discussion in Section 4.1). When dealing with global optimization problems, an obvious approach is the Multistart one. In such an approach we simply start different local searches from randomly generated initial points and return the best local minimizer. However, the rapid increase in the number of local minimizers suggests that Multistart can not be an efficient method for this problem. The method we are going to propose is quite close to Multistart (they are both based on multiple local searches and they only differ in the mechanism for the generation of the initial points) but at the same time also dramatically more efficient than Multistart. In the field of global optimization such method has been (to the authors' knowledge) first applied to molecular conformation problems (see [14, 28]) under the name of Monotonic Basin Hopping (MBH), but in fact it can also be viewed as a special case of the Iterated Local Search (ILS) approach (see, e.g., [17]), which is usually employed in the field of combinatorial optimization problems. Since our problem belongs to the global optimization field, we will refer to this method with MBH. Its description is rather simple. The main ingredients of the method are: a *local search procedure* LS, a *perturbation move* P, and a *stopping rule* SR.

Monotonic Basin Hopping

```
    Let $X$ be a local minimum (randomly generated)
    While $SR$ not satisfied
        Let $Y := LS(P(X))$
        If $f(Y) < f(X)$ Then
            let $X := Y$
        EndIf
    EndWhile
    Return $X$
```

In the next subsections we will detail our choices of LS, P and SR for the problem at hand, but before that we remark once again how close is MBH with respect to the Multistart approach. In fact, we can even consider Multistart as a special case of MBH where the perturbation move $P(X)$ simply generates a random point, independent from the current one X.

2.1 Local search procedure

As shown in (1)-(4), our problem can be viewed as a non-convex one with objective and constraint functions continuously differentiable infinitely many times. Therefore, any local search method for this kind of problems can be employed. However, our past experience (see [1, 2]) suggests that SNOPT [9] is particularly well suited for these problems.

2.2 Perturbation move

One of the keys of the success of the existing MBH or ILS methods is often the perturbation move. A good rule is to choose it in such a way that the structure of the current local minimizer is not completely disrupted. The basic idea is that the method should move between different but ”close” local minimizers. In the case of equal circles a very simple but, as we will see, quite effective, perturbation move, is based on a uniform random perturbation of each coordinate of each circle center within some interval $[-\Delta, \Delta]$. The single parameter Δ, on which the perturbation depends, is of great importance. If Δ is too small, the starting point will be very likely in the basin of attraction of the current local minimizer (we are not disrupting at all the structure of the current local minimizer); on the other hand, if Δ is too large, the method becomes basically equivalent to a Multistart method (which disrupts the structure too much). In Section 4.2 we will further discuss the choice of Δ and perform experiments in order to select an appropriate value for it.

2.3 Stopping rule

Ideally we would like to stop a method as soon as no more progress can be expected. For the Multistart method, for which, under mild assumptions, it can be proved that it is able to detect the global minimizer with probability one if we allow for an infinite number of local searches, this would mean stopping when the global minimizer has been detected. Instead, a single run of MBH does not necessarily lead to a global minimizer and might get stuck into a local minimizer from which it is unable to escape. In such case what we can do is simply to restart MBH from a new random starting point (a sort of Multistart where local searches are substituted by MBH runs). In practice, if no special information is available, we are unable to stop when we are really sure that no more progress will be possible. The best we can do is to stop when no improvement has been observed for a sufficiently large number of iterations (of course, this is just a heuristic rule with no guarantee that improvements are not possible any more). The number of iterations without improvements after which we stop MBH is denoted by the parameter `MaxNoImp`. The choice of this parameter is particularly important: we should not stop too early (which could mean that we are not patient enough to reach the global minimizer) or too late (which would mean a waste of computational effort). The choice of this parameter will be computationally investigated in Section 4.3.

3 Population Basin Hopping

Each run of MBH follows a single path through the space of local minimizers. An alternative to MBH is Population Basin Hopping (PBH) [10], inspired by the Conformational Space Annealing algorithm (see, e.g., [15]), in which the single path search is substituted by a multiple path search. During this search, members of the population collaborate with each other in order to guarantee *diversification* of the search and to avoid the *greediness* which might characterize a single path search. The new ingredient in PBH with respect to MBH is a *dissimilarity* measure d, which quantifies the similarity between solutions. In particular, given two generic solutions X, Y, d should satisfy

$$d(X,X) = 0 \quad \text{and} \quad d(X,Y) \geq 0.$$

New parameters are N (the size of the population) and $dcut$ (a threshold dissimilarity value). The overall algorithm is the following.

Population Basin Hopping

```
    Let $\mathcal{X}$ be a collection of $N$ local minimizers (randomly generated)
    While $SR$ not satisfied
        let $\mathcal{Y} := \{LS(P(X)) \ : \ X \in \mathcal{X}\}$
        Repeat for each $Y_k \in \mathcal{Y}$
            let $X_h \in \arg\min_{X \in \mathcal{X}} \ d(X, Y_k)$
            If $d(Y_k, X_h) > dcut$ Then
                let $X_h \in \arg\max_{X \in \mathcal{X}}\ f(X)$
            EndIf
            If $f(Y_k) < f(X_h)$ Then
                let $X_h := Y_k$
            EndIf
        EndRepeat
    EndWhile
    Return $\mathcal{X}$
```

Basically, at each iteration: a set $\mathcal{Y}$ of new candidates is generated through the application of the perturbation move to each member of the population; each new candidate Y_k, $k = 1, \ldots, N$, competes either with the member X_h of the current population $\mathcal{X}$ most similar to it with respect to the dissimilarity measure d (if $d(X_h, Y_k) \leq dcut$), or with the worst member of the population (if $d(X_h, Y_k) > dcut$, i.e., Y_k is dissimilar enough with respect to all members of the current population); if it wins (i.e., if it has a better function value), it replaces X_h in the population for the next iteration. Note that MBH is, in fact, a special case of PBH where $N = 1$. There is a trade off between two conflicting objectives in choosing N. We have already outlined above the (possible) advantages of PBH: increasing N increases diversification and decreases greediness. On the other hand, increasing N also increases the computational effort per iteration. We will discuss appropriate choices for N in Section 4.4.

In what follows we discuss the dissimilarity measure and the *dcut* value which will be employed throughout the paper. Let $X = \{(\alpha_{i1}, \alpha_{i2})\}_{i=1,\ldots,n}$ and $Y = \{(\beta_{i1}, \beta_{i2})\}_{i=1,\ldots,n}$ be two distinct local minimizers. Let $\rho_h(X)$ be the distance of circle h from the barycenter of the centers of all circles in the local minimizer X, and define $\rho_h(Y)$ in a similar way; let δ_X be the vector whose components are the distances $\rho_h(X)$ ordered in a nondecreasing way, and define δ_Y in a similar way; finally, we define the following dissimilarity measure:

$$d(X, Y) = \sum_{k=1}^{n} |\delta_X[k] - \delta_Y[k]|. \tag{5}$$

The value *dcut* will be fixed throughout the paper to half the average dissimilarity within the initial randomly generated population.

Finally, the stopping rule *SR* is basically the same employed for MBH: we stop if the best member of the population does not change for a fixed number `MaxNoImp` of iterations.

4 Computational experiments and analysis

We performed different computational experiments both to analyze the properties of the problem under investigation and to select the parameter values for MBH and PBH in an appropriate way. All the tests have been performed on a Pentium IV 2.4 GHz 1GB Ram.

4.1 Number of local minimizers

Our first set of experiments aims at estimating how the number of local minimizers increases with the number n of circles. In order to recognize *distinct* local minimizers we consider their objective function values (i.e., the radius of the container). We adopt a conservative criterion by declaring two local minimizers different if they have a large enough difference in their objective function values. Taking into account the precision of the local solver, the threshold value above which two local minimizers are considered as distinct ones on the basis of their objective function values has been fixed to 10^{-8}. Note that, according to this criterion, we may consider as equal also different minimizers. In spite of this, the increase in the number of distinct local minimizers turns out to be very quick. Indeed, in Figure 1 we report the total number of distinct local minimizers which have been detected over 50,000 local searches starting from randomly generated (over a sufficiently large box) initial points for n up to 40. Though the increase is not a regular one, the trend is quite clear, showing a rapid increase with the number of circles. This gives a clear indication that Multistart is most likely not an appropriate method to tackle this problem, which will be confirmed by the results reported in the next subsection.

4.2 Choice of the parameter Δ and Multistart

We have already commented about the importance of parameter Δ. We remark that a good choice of Δ depends on two conflicting objectives. We define *successful* a run of MBH leading to a global minimizer. On one hand, we would like that a successful run of MBH converges to the global minimizer as fast as possible; on the other hand, we would like to maximize the probability that a run of MBH is successful. Note that the first objective is maximized when Δ is very small: in such case a successful run is just one where the initial point is already the global minimizer. On the other hand, this is exactly the situation where the second objective is minimized. The second objective is maximized when Δ is very large (any run converges to the global minimizer in this case), but in this case the first objective is minimized (the convergence is very slow, basically the same as the one of Multistart).

We tested four different values $\Delta \in \{0.6, 0.8, 1.0, 1.2\}$. At the same time we also tested the Multistart algorithm. For a fair comparison, we allowed a number of local searches in Multistart equal to twice the number of local searches required by MBH with $\Delta = 0.8$. In all tests we fixed the parameter `MaxNoImp` to 100 (such choice is justified by the results reported in Section 4.3). The results

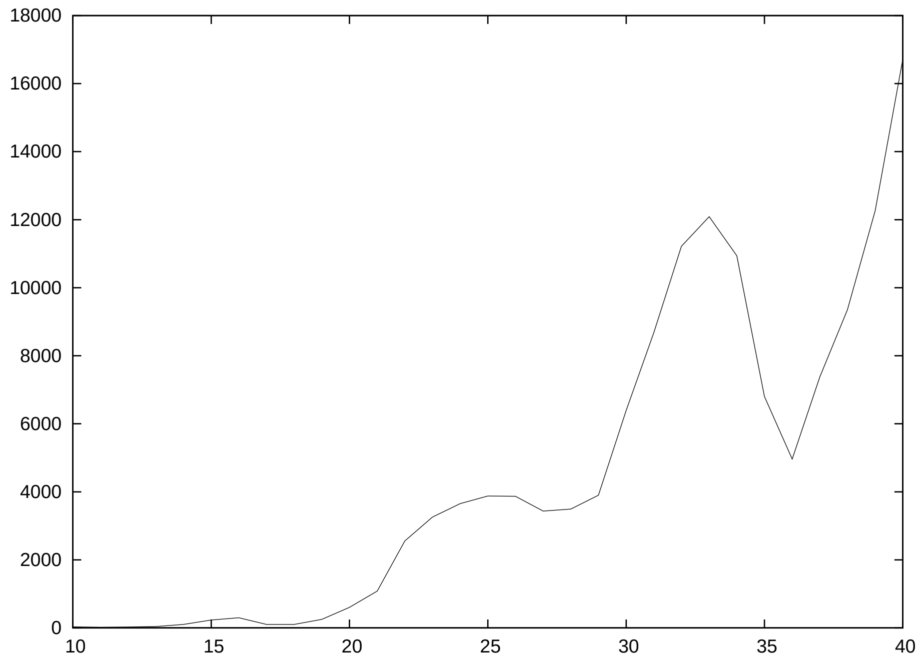

Figure 1: Empirically determined number of local minima.

are reported in Table 1. Column `BestKnown` reports the best known result, according to [22], for a given n value. Column `OurResult` reports the best result we obtained with the different Δ values. We notice that together with a couple of results, namely $n = 79, 80$, which are worse with respect to those reported in [22], we also have seven cases, namely $n = 66, 67, 70, 71, 75, 77, 78$, where our results improve those in [22] (within the table improvements are reported in **boldface** and failures in *emphasized* text). Such improvements, together with some others, are also displayed in Tables 2-3. Column `NrSuccesses` reports the number of successes over 5 MBH runs for each Δ value. For the case $\Delta = 1.2$ we also tried with `MaxNoImp` $= 200$, a choice justified by the fact that, as we increase Δ, more time is needed to explore a larger neighborhood. The corresponding number of successes is reported within parentheses. Finally, Column `Multistart` reports the results obtained with the Multistart approach.

We notice that there is not an optimal choice of Δ for all n values. All the same the results give a quite clear indication. Indeed, we can remark that while the value $\Delta = 1.2$ leads to 14 failures (decreasing to 8 with `MaxNoImp` $= 200$) and the values $\Delta = 0.6, 1.0$ both lead to 8 failures, the value $\Delta = 0.8$ only leads to 3 failures. Therefore, we can consider $\Delta = 0.8$ the most robust choice among the tested ones.

As the discussion in Section 4.1 already suggested, the results obtained with Multistart are usually quite poor, and are clearly inferior with respect to those obtained with MBH. In spite of the larger number of local searches allowed for Multistart, such algorithm is able to reach the best known solution only in few cases, and only in a single case (namely $n = 61$) the best known solution is reached quite regularly[1]. It is also important to remark that the cost per local search in MBH is much lower (almost four times lower) with respect to the cost per local search in Multistart. This can be explained with the fact that in MBH local searches are not started from completely random points as in Multistart, but in the neighborhood of a previously detected local minimizer, so that the local search procedure is able to converge in few iterations.

			NrSuccesses				
n	`BestKnown`	`OurResult`	0.6	0.8	1.0	1.2	`Multistart`
60	8.646219845458	8.646219845458	4	5	5	5(5)	8.64621984545792
61	8.661297575540	8.661297575540	4	5	5	5(5)	8.66129757554036
62	8.829765408972	8.829765408972	0	5	0	0(1)	8.82976540897210
63	8.892351537551	8.892351537551	0	3	2	2(2)	8.89235153755062
64	8.961971108486	8.961971108486	1	4	0	1(1)	8.96395305832546
65	9.017397323209	9.017397323209	3	1	4	1(3)	9.01739732320874
66	9.096665836768	**9.096279426924**	2	3	1	0(0)	9.09678021497725
67	9.169119588389	**9.168971881784**	1	0	0	0(0)	9.17621809355103
68	9.229773746751	9.229773746751	2	2	1	0(0)	9.23535110170223
69	9.269761266641	9.269761266641	2	5	4	0(1)	9.28787933388477
70	9.346055334486	**9.345653194048**	0	3	1	1(1)	9.34587739998685
71	9.416206538907	**9.415796896871**	0	4	1	1(1)	9.41689334139778
72	9.473890856713	9.473890856713	1	3	4	0(1)	9.47518329418440
73	9.540509504650	9.540509504650	1	2	0	0(0)	9.55517439692162
74	9.589239461626	**9.589232764339**	2	1	2	0(1)	9.61087441941583
75	9.672029634515	**9.672029631947**	0	2	0	0(1)	9.67643142736819
76	9.729596802162	9.729596802162	1	1	3	0(0)	9.72959680216165
77	9.798987497420	**9.798911924507**	1	3	1	0(0)	9.79926512503908
78	9.857712212603	**9.857709899885**	0	2	0	0(1)	9.85884796436351
79	9.905063467661	*9.924486046000*	0	0	0	0(0)	9.92100276409317
80	9.968151813153	*9.969802931195*	0	0	0	0(0)	9.97399528196106

Table 1: Overall results for 10 MBH runs with different Δ values and results of the Multistart approach.

4.3 Choice of the `MaxNoImp` parameter

As already pointed out, `MaxNoImp` is an important parameter for MBH. Too low a value would cause to stop the algorithm before convergence is reached, while too large a value would cause a waste of computational effort. Also in

[1]The case $n = 61$ belongs to the regular sequence $n = 3k(k-1) + 1$, $k = 1, \ldots$, for which the (presumably) optimal solution has a circle centered at the origin and, around it, successive layers, each made up by $6j$ circles, $j = 0, 1, \ldots, k-1$

this case we can hardly expect that there exists an optimal choice for all n values. So our aim is to look for a *robust* choice of this parameter value. We tested the following set of values: `MaxNoImp`$\in \{50, 100, 200, 300, 400, 500\}$. In view of the results in Section 4.2 we fixed $\Delta = 0.8$. The first set of results over 5 runs of MBH for each value $n = 30 \dots 100$ is reported in Table 2. The first column `BestKnown` reports the best known solution according to [22] for a given n value. Column `OurResult` reports the best result we obtained over the 5 MBH runs (in **boldface** we report the results which improve those reported in [22], while in *emphasized* text we report the failures). In Column `NrSuccesses` we report for each `MaxNoImp` value the number of times the best solution reported in [22] has been reached (or improved)[2]. Finally, in Column `CPU time` we report the overall computation time (in seconds) for the 5 runs, referred to the value `MaxNoImp`=500. We remark that in 19 cases we could obtain an improvement with respect to [22]. All the same, we also have some failures. In particular, we have 13 failures with `MaxNoImp`=50 but these immediately drop down to 9 with `MaxNoImp`=100 and progressively decrease to 5 with `MaxNoImp`=500. As a further test we decided to enlarge the number of MBH runs from 5 to 50 for the 9 cases where a failure occurred with `MaxNoImp`=100. The results are reported in Table 3. We notice that $n = 31$ turns out to be an extremely hard case for MBH: only with `MaxNoImp`=500 a single success could be obtained. This case will be further discussed in Section 4.4 where we will consider a different (and more successful over this instance) approach. In all the other cases we always have at least one success (with the only exception of the failure for $n = 83$ with `MaxNoImp`=50) and in one case, namely $n = 92$, we have a further improvement with respect to [22]. As a general comment about the results we obtained, we notice that even a very aggressive strategy like choosing `MaxNoImp`=50 is often successful (of course, the number of successes is lower in this case, but this is counterbalanced by the lower computational effort). All the same with this choice failures occur more often (even over 100 runs). For this reason we believe that less aggressive choices like `MaxNoImp`=100 or `MaxNoImp`=200 represent the best compromise between the ability of reaching a good solution and the computational effort required.

4.4 Computational experiments with PBH

We also performed some experiments with PBH, in particular with population size $N = \{1, 2, 4, 5, 8, 10\}$, $\Delta = 0.8$ and `MaxNoImp` = 100 (recall that $N = 1$ is equivalent to MBH) for large n values ranging between 80 and 100. The largest tested N values, say $N \in \{5, 8, 10\}$, usually guarantee the highest percentage of successes, showing that for large N values PBH turns out to be a quite robust approach. On the other hand, we should not forget that increasing N also means increasing the computational cost per iteration. If the results are not compared with respect to the percentage of successes, but rather with

[2]Note that for the common n and parameter values, these results are sometimes slightly different with respect to those reported in Table 1. This is due to the stochastic component of the MBH approach.

			NrSuccesses						
n	`BestKnown`	`OurResult`	50	100	200	300	400	500	`CPU time`
30	6.197741070879	6.197741070879	5	5	5	5	5	5	110.49
31	6.291502622129	*6.352805480965*	0	0	0	0	0	0	130.14
32	6.429462970950	6.429462970950	5	5	5	5	5	5	151.14
33	6.486703123560	6.486703123560	5	5	5	5	5	5	164.69
34	6.610957090001	6.610957090001	5	5	5	5	5	5	174.41
35	6.697171091790	6.697171091790	5	5	5	5	5	5	179.81
36	6.746753793424	6.746753793424	5	5	5	5	5	5	247.81
37	6.758770483144	6.758770483144	5	5	5	5	5	5	240.7
38	6.961886965228	6.961886965228	5	5	5	5	5	5	213.43
39	7.057884162624	7.057884162624	5	5	5	5	5	5	231.1
40	7.123846435943	7.123846435943	5	5	5	5	5	5	283.29
41	7.260012328677	7.260012328677	4	4	4	5	5	5	264.59
42	7.346796406943	7.346796406943	4	5	5	5	5	5	294.11
43	7.419944856341	7.419944856341	5	5	5	5	5	5	449.45
44	7.498036682995	7.498036682995	4	4	5	5	5	5	313.27
45	7.572912326368	7.572912326368	0	2	3	3	3	3	482.36
46	7.650179914694	7.650179914694	5	5	5	5	5	5	414.0
47	7.724170052598	7.724170052598	3	5	5	5	5	5	428.22
48	7.791271430559	7.791271430559	4	4	5	5	5	5	467.42
49	7.886870958803	7.886870958803	1	1	1	1	2	2	775.54
50	7.947515274784	7.947515274784	2	2	4	4	4	4	655.69
51	8.027506952419	8.027506952419	5	5	5	5	5	5	542.55
52	8.084717190690	8.084717190690	5	5	5	5	5	6	699.28
53	8.179582826841	8.179582826841	1	2	3	3	3	3	807.84
54	8.203982383469	8.203982383469	2	3	3	3	3	3	701.1
55	8.211102550928	8.211102550928	3	3	4	4	4	4	1178.43
56	8.383529922579	8.383529922579	5	5	5	5	5	5	732.19
57	8.447184653410	8.447184653410	5	5	5	5	5	5	952.84
58	8.524553770140	8.524553770140	3	4	4	5	5	5	1078.31
59	8.592499959370	8.592499959370	5	5	5	5	5	5	1495.39
60	8.646219845458	8.646219845458	0	5	5	5	5	5	1168.12
61	8.661297575540	8.661297575540	5	5	5	5	5	5	1031.43
62	8.829765408972	8.829765408972	2	3	4	4	4	4	1400.12
63	8.892351537551	8.892351537551	3	4	5	5	5	5	1363.18
64	8.961971108486	8.961971108486	0	1	1	1	1	1	1266.97
65	9.017397323209	9.017397323209	3	3	3	3	3	3	1708.53
66	9.096665836768	**9.096279426924**	3	3	3	3	4	4	1813.23
67	9.169119588389	**9.168971881784**	1	1	2	2	2	2	2563.13
68	9.229773746751	*9.234077342045*	0	0	0	0	0	0	1390.52
69	9.269761266641	9.269761266641	5	5	5	5	5	5	1831.42
70	9.346055334486	**9.345653194048**	1	2	3	3	3	3	2391.79
71	9.416206538907	**9.415796896871**	4	5	5	5	5	5	2487.81
72	9.473890856713	9.473890856713	3	3	3	3	3	3	2246.99
73	9.540509504650	**9.540346152138**	1	1	1	1	1	2	2806.35
74	9.589239461626	**9.589232764339**	1	2	2	2	2	2	2543.33
75	9.672029634515	**9.672029631947**	1	2	2	2	2	2	3034.33
76	9.729596802162	9.729596802162	1	1	1	1	2	3	3927.92
77	9.798987497420	**9.798911924507**	1	1	3	4	4	4	3694.47
78	9.857712212603	**9.857709899885**	0	0	0	0	0	2	4852.32
79	9.905063467661	*9.909306621540*	0	0	0	0	0	0	3590.06
80	9.968151813153	*9.969802931195*	0	0	0	0	0	0	4032.13
81	10.010864241201	10.010864241201	2	2	2	3	3	3	5293.59
82	10.050824223451	10.050824223451	1	3	4	4	4	4	5432.51
83	10.116864426926	**10.116857875102**	0	0	1	2	2	2	7914.08
84	10.149530867236	10.149530867236	4	5	5	5	5	5	4780.79
85	10.163111465877	10.163111465877	3	4	4	4	5	5	7532.68
86	10.298701310984	**10.298701053110**	3	4	5	5	5	5	5128.9
87	10.363209161980	**10.363208505078**	2	5	5	5	5	5	4927.78
88	10.432342147160	**10.432337692732**	4	4	4	4	4	4	5578.01
89	10.500627671551	**10.500491814574**	2	2	2	3	3	3	4874.01
90	10.546069177954	10.546069177954	3	3	3	3	3	3	5059.66
91	10.566772233506	10.566772233506	2	3	3	3	3	3	6113.41
92	10.684689759023	*10.687984877108*	0	0	0	0	0	0	10041.7
93	10.733386127679	**10.733352600260**	0	1	2	3	3	3	7251.63
94	10.778032163883	**10.778032160252**	1	2	2	2	2	2	7831.68
95	10.840205021597	10.840205021597	0	0	0	0	1	1	13635.1
96	10.883669894312	10.883669894312	1	1	1	1	1	1	9701.68
97	10.938791648300	**10.938590110073**	1	1	1	2	2	2	9259.48
98	10.979383128207	10.979383128207	0	0	0	2	5	5	19099.9
99	11.037197388568	**11.035161062993**	4	5	5	5	5	5	7533.95
100	11.082527292540	**11.082149724310**	1	2	4	5	5	5	15311.6

Table 2: Overall results for 5 MBH runs with different `MaxNoImp` values

			NrSuccesses						
n	`BestKnown`	`OurResult`	50	100	200	300	400	500	`CPU time`
31	6.291502622129	6.291502622129	0	0	0	0	0	1	1911.75
68	9.229773746751	9.229773746751	10	16	18	19	21	21	25695.9
78	9.857712212603	**9.857709899885**	4	6	9	16	18	21	50324.2
79	9.905063467661	9.905063467661	1	1	2	2	2	2	44082.2
80	9.968151813153	9.968151813153	3	3	3	3	4	4	66829.3
83	10.116864426926	**10.116857875102**	0	3	9	13	19	21	99316.2
92	10.684689759023	**10.684645847916**	1	3	3	4	5	5	88112.0
95	10.840205021597	10.840205021597	4	9	15	17	18	19	118471.0
98	10.979383128207	10.979383128207	12	22	28	35	41	41	123050.0

Table 3: Overall results for 50 MBH runs with different `MaxNoImp` values over some hard instances

respect to the number of local searches per success, then low N values (even $N = 1$, i.e., MBH) are often the best option. Basically, it seems that for these problems single or few path searches are already quite efficient and that the benefits coming from the greater diversification guaranteed by PBH with larger N values are overridden by the larger computational cost per iteration.

We also compared MBH (50 runs with `MaxNoImp` up to 500) and PBH (10 runs with $N = 10$) over the hardest instances for MBH (those reported in Table 3). The results usually confirm the previously discussed ones: PBH is quite robust (always at least 3 successes over 10 runs) but comparable with MBH in terms of number of local searches per success. There is, however, an exception, namely the case $n = 31$, which is worthwhile to discuss. In this case PBH strongly outperforms MBH. While MBH really had to struggle with this instance and was able to detect the best known solution only in 1 out of 50 runs with `MaxNoImp`=500 (and never for lower values of `MaxNoImp`), PBH was able to reach the best known solution in 9 out of 10 runs. This result suggests that PBH with a relatively large N value is probably not, on average, the most efficient approach, but turns out to be a quite robust one, which guarantees to return a solution within a reasonable time also on those instances which are particularly hard for MBH.

As a final remark we should emphasize that here we just tested a single dissimilarity measure. It is certainly possible that future researches will reveal new measures, able to make PBH more efficient.

5 The case of unequal circles

In order to deal with the case of unequal circles we could simply extend the approaches employed for the case of equal circles with a slight variant in the perturbation move: the coordinates of each circle i are displaced by a uniform random perturbation within the interval $[-\Delta r_i, \Delta r_i]$, where r_i denotes the radius of circle i. But, as we will see through some experiments, this simple extension is not the best way to tackle the problem. Indeed, the case of unequal circles has some peculiarities which have to be taken into account. The combinatorial side of this problem, represented by the different radii of the circles, can be exploited in some ways. In particular, we can define another possible perturbation move, based on leaving unchanged the center of two circles but swapping their radii (of course, this would not cause any change when the two circles are equal). It turns out that in case of unequal circles the latter move is often much more effective than the former one (see the experiments reported in Tables 5-6).

Another good strategy which exploits the different radii of the circles is the following:

- remove the "small" circles;
- solve the problem with the remaining (largest) circles;
- sequentially insert the missing circles (following a non increasing order of the radii) in the "holes" of the current solution (possibly by enlarging the radius of the circular container).

Of course, we need to define what a "small" circle is. We defined a circle i as small if its radius is at least four times smaller than the largest one, i.e., circle i is small if

$$r_i \leq \frac{1}{4} \max_{j=1,\ldots,n} r_j. \tag{6}$$

This strategy strongly simplifies some of the tests by a considerable reduction of the search space during the first phase where some circles are removed.

A set of 18 test instances for the case of unequal circles is reported, e.g., in [11].[3] In some of these test problems the set of circles together with their radii is a subset of those for other test problems, while the optimal radius is the same. For instance, in test n.4 there are nine circles, three with radius equal to 10 and six with radius equal to 3.533, while in test n. 5 there are twelve circles, including those of test n.4 plus three circles with radius equal to 2.3. The optimal radius for both tests is 21.547. This means that when solving the larger test n.5, we are actually solving also the smaller test n.4 (it is enough to remove from the solution for test n.5 the three circles with radius 2.3). Taking into account these simplifications, we can reduce the test set to the nine ones reported in Table 4. The strategy of removing small circles strongly simplifies some of these tests. In particular, the first three tests are considerably simplified. Indeed, test n.1 reduces to six circles in the first phase; tests n.2 and 3 reduce to nine circles in the first phase, and in all these three cases the insertion of the missing circles can be easily carried on without having to enlarge the radius of the

[3]Actually, in that paper 24 test problems are reported, but four of them are with equal circles (namely, tests n.2, 21-23), and two of them are equivalent to other problems within the test set (namely, test n.7 is equivalent to test n.19, and test n.13 is equivalent to test n.20.

Test n.	n	Radii
1	28	$r_{1-3} = 10,\ r_{4-6} = 4.826,\ r_{7-12} = 2.371,\ r_{13} = 1.547$ $r_{14-19} = 1.345,\ r_{20-22} = 1.161,\ r_{23-28} = 0.9$
2	25	$r_{1-3} = 10,\ r_{4-9} = 3.533,\ r_{10-12} = 2.3,\ r_{13-18} = 1.8$ $r_{19} = 1.547,\ r_{20-25} = 1.08$
3	17	$r_{1-4} = 100,\ r_{5-9} = 41.415,\ r_{10-17} = 20$
4	10	$r_1 = 50,\ r_2 = 40,\ r_{3-5} = 30,\ r_6 = 21$ $r_7 = 20,\ r_8 = 15,\ r_9 = 12,\ r_{10} = 10$
5	11	$r_{1-2} = 25,\ r_{3-4} = 20,\ r_5 = 15,\ r_6 = 14$ $r_7 = 12,\ r_8 = 11,\ r_9 = 10.5,\ r_{10} = 10,\ r_{11} = 8.4$
6	14	$r_1 = 40,\ r_2 = 38,\ r_3 = 37,\ r_4 = 36,\ r_5 = 35$ $r_6 = 31,\ r_7 = 27,\ r_8 = 23,\ r_9 = 19,\ r_{10} = 17,\ r_{11} = 16$ $r_{12} = 15,\ r_{13} = 14,\ r_{14} = 11$
7	17	$r_1 = 25,\ r_2 = 20,\ r_{3-4} = 15,\ r_{5-7} = 10,\ r_{8-17} = 5$
8	15	$r_1 = 1,\ r_{i+1} = r_i + 1,\ i = 1, \ldots, 14$
9	162	$r_{1-3} = 1.8,\ r_4 = 1.75,\ r_{5-16} = 1.3,\ r_{17-25} = 1.05$ $r_{26-40} = 0.9,\ r_{41-71} = 0.8,\ r_{72} = 0.75,\ r_{73-83} = 0.7$ $r_{84-137} = 0.65,\ r_{138-162} = 0.55$

Table 4: Test set with unequal circles

circular container (i.e., all the missing circles can be inserted in the "holes" of the container). MBH over the reduced set of circles, followed by insertion of the missing circles, easily solves these three instances. In test n.4 the reduction of the search space is less strong (only two circles are initially removed) but all the same also this test turns out to be easily solved by MBH. For this reason we will not further discuss the first four instances and will concentrate our attention to the last five ones. In Table 5 we report the results obtained with 50 runs of MBH, 10 runs of PBH with $N = 5$, 5 runs of PBH with $N = 10$ over tests n.5,7,9, and 500 runs of MBH, 100 runs of PBH with $N = 5$, 50 runs of PBH with $N = 10$, over tests n.6, 8, for which we observed a larger variability of the final results. We remark that the dissimilarity measure is defined in a slightly different way with respect to (5): given a local minimizer X, in vector δ_X we first place the distances with respect to the barycenter of the circles with largest radius, ordered in a nondecreasing way, then the distances with respect to the barycenter of the circles with second largest radius, ordered again in a nondecreasing way, and so on for all the different radii. Column `Best Known` reports the putative optimum for the instance as reported in [11]. We notice

Test n.	`Best Known`	MBH	PBH with $N = 5$	PBH with $N = 10$
5	60.89	60.7099 (5/50)	60.7099 (3/10)	60.7099 (1/5)
6	114.98	113.5587 (1/500)	113.5587 (1/100)	113.5587 (1/50)
7	49.6837	49.1873 (31/50)	49.2296 (10/10)	49.2296 (5/5)
8	39.37	38.8379 (11/500)	38.8379 (3/100)	38.8379 (8/50)
9	11.6809	11.5528 (1/50)	11.5413 (1/10)	11.5519 (1/5)

Table 5: Best results and (within parenthesis) the number of times they have been obtained with MBH, PBH with $N = 5$, and PBH with $N = 10$.

that MBH and PBH are able to improve the best known results for all these instances. Over test n.7 we get better results with MBH, but this case deserves some discussion. Indeed, according to rule (6) the last ten circles with radius equal to 5 are removed during the first phase. The different runs of MBH over the reduced space return two distinct solutions with the seven remaining circles, one with radius 48.6111 and the other with radius 48.922, so that the first one is clearly better than the second one. But when moving to the second phase (insertion of the missing circles), the situation is reversed: the first solution leads to a solution with radius 49.2296, while the second one leads to a better solution with radius 49.1873. Basically, the second solution has a worse radius but larger holes where the missing circles can be placed. Since in PBH we performed the second phase only from the best member of the final population (corresponding in all cases to the first solution), we were never able to reach the best solution with radius 49.1873. Instead, such solution was often reached (8 out of 10 runs with $N = 5$, and 5 out of 5 runs with $N = 10$) once we allowed to perform the second phase from all members of the final population. What we can conclude from this experiment is that it is often a good strategy to perform the insertion of missing circles not only from the best solution returned by the first phase, but also from some suboptimal solutions obtained during the first phase, because the latter may lead to better solutions after insertion of the missing circles.

In test n.5 we obtained a large variety of final solutions in the different runs. The best result was obtained only once with all three methods. However, it seems that PBH is more robust with respect to MBH. Indeed, the final result was below 114 in 14 out of 500 runs of MBH, 27 out of 100 runs of PBH with $N = 5$, and 20 out of 50 runs of PBH with $N = 10$. Something similar, although less evident, also holds for test n.8, where a result below 39 has been reached in 75 out 500 runs with MBH, 46 out of 100 runs of PBH with $N = 5$, and 37 out of 50 runs of PBH with $N = 10$. Also in test n.8 we observed a large variability of the final solutions. This test turns out to be particularly challenging. Such instance is one (actually of moderate size) among those proposed in the Circle Packing Contest (see `http://www.recmath.org/contest/CirclePacking/index.php`), and its difficulty seems to confirm that such instances are more challenging than the other test instances with unequal circles reported in the literature. More generally, our impression is that the hardest instances for the case of unequal circles are those with many circles with slightly different radii. For a discussion about how to deal with the instances of the contest we refer to [1].

In test n.9 PBH, both with $N = 5$ and with $N = 10$, returns better results with respect to MBH. But this instance will be discussed in more detail below.

In Table 6 we report the results we obtained with the original perturbation move, i.e., the coordinates of each circle i with radius r_i are displaced by a uniform random perturbation within the interval $[-\Delta r_i, \Delta r_i]$. It can be clearly seen that in all cases, except test n.9, such perturbation move, which does not take into account the combinatorial nature of the problem, delivers results inferior to those obtained with the perturbation based on swapping the centers of circles with different radii. Note that:

- we restricted to just 50 runs of MBH, 10 of PBH with $N = 5$, and 5 of PBH with $N = 10$, also for tests n.6 and 8, but the indications given by these fewer runs with respect to those with the other perturbation move are already quite clear;

- the results obtained with PBH are usually better than those obtained with MBH;
- for test n.7 we report the results when circles are added to all the members of the final population.

Test n.9 deserves a separate comment. For this case it seems that the random perturbation is better than the one based on swaps. If we look at this instance, we notice that it contains a large number of circles with the same radius (e.g., 54 circles, one third of the total number of circles, have radius equal to 0.65). It is possible that such circles occupy a portion of the container which can not be optimized by swapping moves (recall that such moves only involve circles with different radii), while it can be optimized efficiently by random perturbations. Seen in another way, we have two distinct aspect in a problem with unequal circles: a *continuous* one, represented by the fact that circle centers have to be chosen in $\mathbb{R}^2$, and a *combinatorial* one, due to the different radii of the circles. In the case of circles with all equal radius the combinatorial component simply does not exist, while in case there are a lot of circles with different radii, the combinatorial component is more relevant than the continuous one. In case of test n.9, with few different radii and many circles with the same radius, it seems that taking into account the continuous aspect (through the use of the random perturbation) is more important than taking into account the combinatorial aspect (through the use of swapping moves). Something which could be explored in the future is a mixed strategy, where both swap moves and random ones are employed.

Test n.	Best Known	MBH	PBH with $N = 5$	PBH with $N = 10$
5	60.89	62.2629 (1/50)	61.8213 (1/10)	61.3821 (1/5)
6	114.98	115.9993 (1/50)	115.8722 (1/10)	115.7018 (1/5)
7	49.6837	49.2296 (1/50)	49.1873 (11/10)	49.1873 (1/5)
8	39.37	39.5503 (1/50)	39.4056 (1/10)	39.4980 (1/5)
9	11.6809	11.5269 (1/50)	11.5242 (1/10)	11.5118 (1/5)

Table 6: Best results and (within parenthesis) the number of times they have been obtained with MBH, PBH with $N = 5$, PBH with $N = 10$, using the random perturbation.

References

[1] Addis B., Locatelli M., Schoen F. Efficiently packing unequal disks in a circle, *Operations Research Letters*, 36 (1), 37-42 (2008)

[2] Addis B., Locatelli M., Schoen F., Disk packing in a square: a new global optimization approach, to appear in *INFORMS Journal on Computing* (2008)

[3] Birgin E.G., Sobral F.N.C., Minimizing the object dimensions in circle and sphere packing problems, *Computers & Operations Research*, 35, 2357-2375 (2008)

[4] Boll D.V., Donovan J., Graham R.L., Lubachevsky B.D., Improving dense packings of equal disks in a square, *The Electronic J. of Comb.*, 7, R46 (2000)

[5] Casado L.G., Garcia I., Szabó P.G., Csendes T., Equal circles packing in square II: new results for up to 100 circles using the TAMSASS-PECS algorithm, in *New trends in equilibrium systems: Mátraháza Optimization Days*, Kluwer Academic Publishers (1998)

[6] Castillo I., Kampas F.J., Pinter J.D., Solving circle packing problems by global optimization: Numerical results and industrial applications, to appear in *European Journal of Operational Research*

[7] Graham R.L., Lubachevsky B.D., Repeated patterns of dense packings of equal disks in a square, *The Electronic J. of Comb.*, 3, 1-16 (1996)

[8] de Groot C., Peikert R., Würtz D., Monagan M., Packing circles in a square: a review and new results, *System Modelling and Optimization, Proc. 15th IFIP Conf. Zürich*, 45-54 (1991)

[9] Gill P.E., Murray W., Saunders M. A., SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization, *SIAM J. Optim.*, 12, 979-1006, (2002)

[10] Grosso A., Locatelli M., Schoen F., A population based ap- proach for hard global optimization problems based on dissimilarity mea- sures, *Mathematical Programming*, 110(2), 373-404 (2007)

[11] Hifi M., M'Hallah R., Adaptive and restarting techniques-based algorithms for circular packing problems, *Computational Optimization and Applications*, 39, 17-35 (2008)

[12] Huang H., Huang W., Zhang Q., Xu D., An improved algorithm for the packing of unequal circles within a larger containing circle, *European Journal of Operational Research*, 141, 440-453 (2002)

[13] Huang W., Li Y., Jurkowiak B., Li C. M., Xu R.C., A two-level search strategy for packing unequal circles into a circle container, in *Proceedings of the International Conference on Principles and Practice of Constraint Programming*, Springer-Verlag, 868-872 (2003)

[14] Leary R.H., Global optimization on funneling landscapes, *J. Global Optim.*, 18, 367-383, (2000)

[15] Lee J., Scheraga H.A., Rackovsky S., New optimization method for conformational energy calculations on polypeptides: conformational space annealing, *J Comput. Chem.*, 18(9), 1222-1232 (1997)

[16] Locatelli M., Raber U., Packing equal circles in a square: a deterministic global optimization approach, *Discrete Applied Mathematics*, 122,139-166 (2002)

[17] Lourenco H.R., Martin O., Stützle T., Iterated Local Search. In: F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, 321-353, Kluwer Academic Publishers, Norwell, MA (2002)

[18] Markot M.C., Csendes T., A new verified optimization technique for the "packing circles in a unit square" problems, *SIAM Journal on Optimization*, 16, 193-219 (2005)

[19] Mladenovic N., Plastria F., Uroevi D., Reformulation descent applied to circle packing problems, *Computers & Operations Research*, 32, 2419-2434 (2005)

[20] Nurmela K.J., Oestergard P.R.J., More Optimal Packings of Equal Circles in a Square, *Discrete Comput. Geom.*, 22, 439-457 (1999)

[21] Nurmela K.J., Oestergard P.R.J., Packing up to 50 equal circles in a square, *Discrete Comput. Geom.*, 18, 111-120 (1997)

[22] Packomania web site maintained by E.Specht, `www.packomania.com`

[23] J. D. Pintér J.D., Kampas F.J., Nonlinear optimization in Mathematica with MathOptimizer Professional, *Mathematica in Education and Research*, 10, 1-18 (2005)

[24] Stoyan Y., Yaskow G., Mathematical model and solution method of optimization problem of placement of rectangles and circles taking into account special constraints, *International Transactions in Operational Research*, 5, 45-57 (1998)

[25] Stoyan Y.G., Yaskov G.N., A mathematical model and a solution method for the problem of placing various-sized circles into a strip, *European Journal of Operational Research*, 156, 590-600 (2004)

[26] Szabó P.G., Markót M.C., Csendes T., Global optimization in geometry - circle packing into the square, in: C. Audet, P. Hansen, G. Savard (Eds.), *Essays and Surveys in Global Optimization*, Kluwer, 233-266 (2005)

[27] Szabó P.G., Markót M.C., Csendes T., Specht E., Casado L.G., Garcia I., *New Approaches to Circle Packing in a Square*, Optimization and Its Applications, Springer (2007)

[28] Wales D.J., Doye J.P.K., Global optimization by basin-hopping and the lowest energy structures of Lennard-Jones clusters containing up to 110 atoms, *J. Phys. Chem. A*, 101, 5111-5116, (1997)

[29] Wang H., Huang W., Zhang Q., Xu D., An improved algorithm for the packing of unequal circles within a larger containing circle, *European Journal of Operational Research*, 141, 440-453 (2002)

[30] Zhang D., Deng A., An effective hybrid algorithm for the problem of packing circles into a larger containing circle, *Computers & Operations Research*, 32, 1941-1951 (2005)