

Optimal Packing of 28 Equal Circles in a Unit Square – the First Reliable Solution *

Mihály Csaba Markót

*Institute of Informatics, University of Szeged, H-6720 Szeged, Árpád tér 2.,
Hungary.*

Abstract. The paper deals with the problem class of finding the densest packings of non-overlapping equal circles within a unit square. We introduce a new interval branch-and-bound algorithm designed specifically for this optimization problem. After a brief description of the applied algorithmic tools, the capabilities of the algorithm are shown by solving the previously unsolved problem of packing 28 circles. The result confirms the optimality of an earlier found approximate solution and shows that it is unique in a certain sense.

Keywords: interval arithmetic, branch-and-bound methods, circle packing

AMS Subject Classification: 52C15, 52C26, 65G30, 90C30

1. Introduction

The original circle packing problem is the following: *place a given number of equal circles without overlapping into a unit square maximizing the diameter of the circles as the objective function.* It is quite easy to see that the problem of *placing a given number of points into the unit square where the objective function to be maximized is the minimal distance between the pairs of points* is equivalent to the circle packing one (see e.g. [17]). Since the latter problem is easier to handle, in the sequel we deal with the point packing problem:

$$\max_{x,y} f_n(x,y), \quad \text{s.t.} \quad 0 \leq x_i, y_i \leq 1, \quad i = 1, 2, \dots, n, \quad (1)$$

where $f_n(x,y) = \min_{1 \leq i < j \leq n} \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$, the unit square is $[0, 1]^2$, and the i th point is located at (x_i, y_i) . The number of the points, i.e. the integer $n (\geq 2)$ is considered as the parameter of the problem class.

In the last four decades several approaches were developed to solve circle packing problem instances. Until now, only the optimal packing of 2, . . . , 9, 14, 16, 25 and 36 circles are proved in a theoretical way. On the other hand, computer-aided optimality proofs exist for $n \leq 20$

* This work was supported by the Grants OTKA T 34350, OMFB D-30/2000 and OMFB E-24/2001.



[2, 3, 14] and for $21 \leq n \leq 27$ [13]. Recently, it was reported [7] that the function value of the currently known best packings are correct within the tolerance value of $1e-5$ for $n = 10, \dots, 35, 37, 38$ (without determining the location of all the optimizers). These methods are using the usual floating point arithmetic and they are bounding the rounding error only during the geometric steps of the algorithm. In contrast to that, the present paper introduces a fully interval-arithmetic based procedure providing the optimal values and all the possible optimizers with high accuracy.

The complete code and the running scripts of the algorithm are available at <http://www.inf.u-szeged.hu/~markot/packcirc.htm>.

2. Interval analysis and the branch-and-bound frame

The set of compact real *intervals* are denoted by \mathbb{I} , where for all $A \in \mathbb{I}$ $A = [\underline{A}, \overline{A}] = \{a \in \mathbb{R} \mid \underline{A} \leq a \leq \overline{A}\}$. Here $\underline{A}, \overline{A} \in \mathbb{R}$ mean the *lower* and *upper bound* of A , respectively. The *width* of an interval is defined by $w(A) := \overline{A} - \underline{A}$. A vector of n intervals is called an *n-dimensional interval* (or a *box*): $X \in \mathbb{I}^n$. For a given set of points $D \subseteq \mathbb{R}^n$, $\mathbb{I}(D)$ denotes the set of all n -dimensional boxes X for which $X \subseteq D$.

We call $F : \mathbb{I}(D) \rightarrow \mathbb{I}$ an *inclusion function* of $f : D \subseteq \mathbb{R}^n \rightarrow \mathbb{R}$, if $f(X) = \{f(x) \mid x \in X\} \subseteq F(X)$ holds for all $X \in \mathbb{I}(D)$, where $f(X)$ is the range of f over X . For more details on interval analysis, see [11].

We sketch the interval branch-and-bound algorithm [9, 15] computing all the global maximizers of the general global optimization problem

$$\max_{z \in Z_0} f(z), \quad (2)$$

where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is a continuous objective function and $Z_0 \in \mathbb{I}^n$ is the search interval.

Algorithm 1:

- 1: $Z := Z_0$; WorkList := $\{(Z, \overline{F}(Z))\}$; $\tilde{f} := \underline{F}(Z)$;
- 2: **while** (WorkList is not empty) **do**
- 3: $(Z, \overline{F}(Z)) := \text{Head}(\text{WorkList})$;
- 4: Delete(Head(WorkList));
- 5: Bisection(Z, U^1, U^2);
- 6: **for** $k := 1$ **to** 2 **do**
- 7: Try to improve \tilde{f} ;
- 8: Use accelerating devices for U^k ;
- 9: **if** (U^k is deleted as a whole) **then continue** with the next k ;

```

10:   if ( $w(F(U^k)) < \varepsilon$ ) then Insert(ResultList, ( $U^k, \overline{F}(U^k)$ ));
11:   else Insert(WorkList, ( $U^k, \overline{F}(U^k)$ ));
12: Maximum := ( $f, \max\{\overline{F}(Z) \mid (Z, \overline{F}(Z)) \in \text{ResultList}\}$ );
13: return ResultList, Maximum;

```

In the following we specify some details of the algorithm:

An interval inclusion function $F(Z)$ for the point packing problems. In [8] an appropriate inclusion function was discussed:

THEOREM 1. [8]: *Let $(X, Y) \subseteq [0, 1]^{2n}$, and let*

$$D_{ij} = \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2}, \quad \forall i, j \in \{1, 2, \dots, n\}.$$

Then

$$F_n(X, Y) := [\min_{1 \leq i < j \leq n} D_{ij}, \min_{1 \leq i < j \leq n} \overline{D}_{ij}] \in \mathbb{I}$$

is an inclusion function of $f_n(x, y)$ over the $2n$ -dimensional box (X, Y) .

Bisection step. The classical subdivision method was used, i.e. we bisected the leading box Z perpendicular to its widest component.

Interval selection. The Moore-Skelboe interval selection rule was applied, i.e. at each iteration cycle the element $(Z, \overline{F}(Z)) \in \text{WorkList}$ having the maximal $\overline{F}(Z)$ value was selected for subdivision. If two or more elements had the same $\overline{F}(Z)$, we chose the oldest one of them. The selection strategy was realized through the appropriate sorting rule of the WorkList.

Accelerating devices: In Step 8 several tests were performed to delete those parts of the interval U^k which cannot contain global maximizer points. In order to test whether the investigated regions contain a global maximizer, we assumed that a guaranteed lower bound \tilde{f} of the global maximum value exists. In the present algorithm the initial \tilde{f} value was determined by computing a guaranteed lower bound of the objective function value on the best-known packing (see Section 4.2. for details).

1. *Cut-off test for U^k :* A box $F(U^k)$ can be eliminated if $\overline{F}(U^k) < \tilde{f}$.
2. *'Method of active areas' for U^k :* Considering U^k in the form of $(X, Y) \subseteq [0, 1]^{2n}$, one can visualize $(X_i, Y_i) \subseteq [0, 1]^2$ as a rectangle in the unit square containing the i th point to be placed. After initializing the remaining regions A_i as $A_i := (X_i, Y_i)$, $i = 1, \dots, n$, from each A_i we can iteratively delete those points which have a distance smaller than \tilde{f} to *all* points of another region A_j , $j \neq i$.

We represented the remaining areas still of interest as (not necessarily convex) polygons described by vertex sets, where the vertices were represented by machine numbers. The algorithmic details of this accelerating device can be found in [10].

3. A global elimination procedure

One can easily see that some difficulties would arise when one started Algorithm 1 on the whole initial box $[0, 1]^{2n}$. The result would consist of at least $n!$ solutions which give the same packing pattern, but these solutions differ from each other only in the indexing of the result components. Another problem is that each solution can be transformed (e.g. by reflections or rotations) into geometrically equivalent packings. A promising solution of the above difficulties was introduced for non-interval based computer-aided proofs in [2, 7, 13]. The method is called *tiling*:

Assuming that a lower bound \tilde{f} for the maximum value of a problem instance is given, split the unit square into closed and connected regions (tiles) in such a way that the distance between any two points in each tile is less than \tilde{f} . Then in a packing with an objective function value greater than or equal to \tilde{f} each tile can contain obviously at most one point of the packing. The optimal packings can be found by executing the search procedure on all possible tile combinations. We divided the square horizontally and vertically into uniform rectangles.

In the earlier studies the initial tile combinations were eliminated sequentially, but the increasing number of those combinations made problem instances for $n \geq 28$ unsolvable within an acceptable time limit: for $n = 27$ a 6×6 tiling can be applied resulting in $\binom{36}{27} \approx 9.4 \cdot 10^7$ initial subproblems. The optimality proof for this case [13] was completed in about one month of CPU time (performing only simple floating point computations). For $n = 28$ at least a 7×6 tiling is needed for the requirements, thus, $\binom{42}{28} \approx 5.3 \cdot 10^{10}$ combinations must be checked. With the old method, this computation would take approximately 47 years.

The main new idea of the proposed procedure is the following: if one can discover patterns of tiles which cannot contain components of an optimal solution, then several tile combinations (as higher dimensional subproblems) containing any of these patterns can be discarded. In [10] a simple tile elimination method was proposed on the basis of a central theorem and its conclusions. The essence of the mentioned theorem can be described as below:

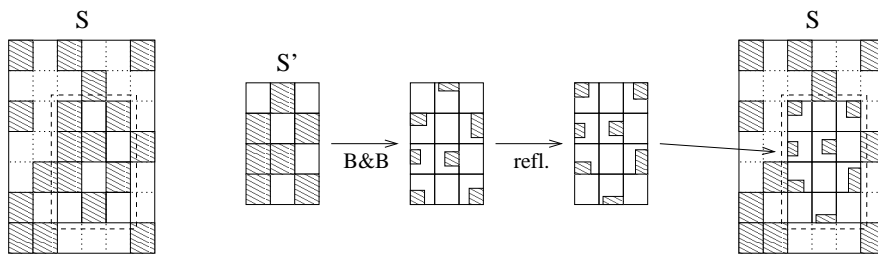


Figure 1. The basic idea of processing tile combinations.

Let φ be a distance-keeping and invertible geometrical transformation (e.g. a reflection or a rotation), and assume that one is able to reduce some regions of an initial tile set S' . When processing a subproblem on a tile set S with $\varphi(S') \subseteq S$, i.e. on a set which contains the transformed initial tile set of the smaller dimensional problem, it is enough to consider *the images of those of the remaining regions of S' as the particular components of the latter problem.* (Figure 1 illustrates the application of the above idea: the actual remaining regions of the tile combinations S and S' are given by the shaded areas. The transformation φ is a reflection to the horizontal centreline of the rectangular region enclosing S' .)

As a consequence of the main theorem of [10], if $S' \subseteq S$ and if it is proved that S' cannot contain point packings having at least \tilde{f} function value, then all the subproblems with tile set S can be eliminated at once (when using the same \tilde{f} value).

4. Optimality for $n = 28$

4.1. HARDWARE AND SOFTWARE ENVIRONMENT

The procedure was carried out on a PC having an Intel Pentium IV 1400 MHz processor and 1 Gbyte RAM, running under the Linux operating system. The global optimization framework was based on the Toolbox for C-XSC libraries [4], while we applied the interval arithmetic routines of PROFIL/BIAS [5]. The multiple precision extension of PROFIL [6] was also built in for some algorithmic steps.

4.2. THE GLOBAL PROCEDURE

The approximate solution and the corresponding function value of the currently known best packing [1] was found in [16]. Those results are rounded to 12 decimal digits. We computed the guaranteed enclosure of the objective function value on the published solution by a function

evaluation with multiple precision intermediate interval data, resulting in the lower bound $\tilde{f} = 0.2305354936419114$. (This differs from the earlier published best solution [16] in the 12th decimal digit due to the above mentioned roundings.) The \tilde{f} value allowed us to split the search space regularly into 7 rows and 6 columns resulting in 42 cells as initial tiles.

We slightly modified the original problem (1) in the following way: at first, f_{28}^2 was maximized instead of f_{28} , and moreover, the search space was enlarged from $[0, 1] \times [0, 1]$ to $[0, 42] \times [0, 42]$. The reason of the latter modification was that we had to split the square into 7×6 pieces with computer representable boundaries since the tile elimination method requires that the individual tiles have exactly the same size. Consequently, \tilde{f} was also transformed to a new value of $\tilde{f} = 93.7506267938615174$.

We followed the basic idea of finding the feasible patterns and remaining areas first on some small tile subsets of the square and then processing bigger and bigger subsets while using the results of the previous steps. In each phase we determined sets of tile combinations consisting either of the remaining areas of the previous phases (or its transformations) or of full tiles. These sets are denoted by

$$D_{s..f}^m, \text{ where } s \leq f, s, f \in \{1, \dots, 6\}, m \in \{0, 1, \dots, 28\}$$

symbolizing that a number of m remaining or full tile regions are considered and the regions are chosen from columns $s, s + 1, \dots, f$. The elements of the sets $D_{s..f}^m$ were processed sequentially by the optimization algorithm. With the exception of the final phase, we stopped the optimization algorithm after a few iterations: in Phases 1, 2, and 3 this number was 3, while in Phase 4 (where the goal was to refine the remaining regions) it was 3000. As a result, some combinations were fully or partly eliminated. The resulting sets are denoted by $\bar{D}_{s..f}^m$. These gave the input components of the subsequent phases. Additionally, we introduce the notation

$${}_{s_1..f_1}^{m_1} D_{s..f}^m, 1 \leq s \leq s_1 \leq f_1 \leq f \leq 6, 0 \leq m_1 \leq m \leq 28$$

denoting that of the subset of $D_{s..f}^m$ in which each element contains exactly m_1 tile regions from columns s_1, \dots, f_1 .

The details of generating and processing tile combinations during the successive phases can be found in [10]. The computational details are highlighted in Table I grouped by the different phases. The table is organized as follows: the two columns on the left show the name and the size (number of elements) of the input sets of the global optimization process. The middle group of columns contains the CPU time (CPUt, in

Table I. Computational details of solving the packing problem of 28 points.

	D	$ D $	CPUt	NFE	NMA	\bar{D}	$ \bar{D} $
Phase 1	$D_{1..3}^{14}$	116 280	788	54 849	60 253	$\bar{D}_{1..3}^{14}$	17 803
	$D_{1..3}^{15}$	54 264	147	4 393	16 652	$\bar{D}_{1..3}^{15}$	1 078
	$D_{1..3}^{13}$	203 490	2 576	232 254	174 998	$\bar{D}_{1..3}^{13}$	77 840
	$D_{1..3}^{16}$	20 349	22	27	5 253	$\bar{D}_{1..3}^{16}$	0
Phase 2	${}_{1..3}^{13} D_{1..4}^{17}$	588 457	64 222	4 638 350	3 002 370	${}_{1..3}^{13} \bar{D}_{1..4}^{17}$	390 113
	${}_{1..3}^{13} D_{1..4}^{18}$	632 243	43 448	2 838 323	2 211 658	${}_{1..3}^{13} \bar{D}_{1..4}^{18}$	226 075
	${}_{1..3}^{13} D_{1..4}^{19}$	43 725	1 231	55 709	81 246	${}_{1..3}^{13} \bar{D}_{1..4}^{19}$	2 973
	${}_{1..3}^{13} D_{1..4}^{20}$	0	-	-	-	${}_{1..3}^{13} \bar{D}_{1..4}^{20}$	0
	${}_{1..3}^{14} D_{1..4}^{17}$	438 269	38 481	2 731 676	1 893 296	${}_{1..3}^{14} \bar{D}_{1..4}^{17}$	231 559
	${}_{1..3}^{14} D_{1..4}^{18}$	354 113	20 278	1 309 583	1 089 172	${}_{1..3}^{14} \bar{D}_{1..4}^{18}$	103 339
	${}_{1..3}^{14} D_{1..4}^{19}$	88 472	3 348	142 521	176 906	${}_{1..3}^{14} \bar{D}_{1..4}^{19}$	9 423
	${}_{1..3}^{14} D_{1..4}^{20}$	2 532	14	10	2 540	${}_{1..3}^{14} \bar{D}_{1..4}^{20}$	0
	${}_{1..3}^{14} D_{1..4}^{21}$	0	-	-	-	${}_{1..3}^{14} \bar{D}_{1..4}^{21}$	0
	${}_{1..3}^{15} D_{1..4}^{17}$	21 760	1 358	95 197	74 121	${}_{1..3}^{15} \bar{D}_{1..4}^{17}$	7 812
	${}_{1..3}^{15} D_{1..4}^{18}$	26 828	1 063	66 255	65 001	${}_{1..3}^{15} \bar{D}_{1..4}^{18}$	4 956
	${}_{1..3}^{15} D_{1..4}^{19}$	14 108	337	13 348	22 503	${}_{1..3}^{15} \bar{D}_{1..4}^{19}$	850
${}_{1..3}^{15} D_{1..4}^{20}$	2 862	23	530	3 209	${}_{1..3}^{15} \bar{D}_{1..4}^{20}$	31	
${}_{1..3}^{15} D_{1..4}^{21}$	0	-	-	-	${}_{1..3}^{15} \bar{D}_{1..4}^{21}$	0	
${}_{1..3}^{15} D_{1..4}^{22}$	0	-	-	-	${}_{1..3}^{15} \bar{D}_{1..4}^{22}$	0	
Phase 3	${}_{1..3}^{13} D_{1..6}^{28}$	238 954	1 954	6 930	244 549	${}_{1..3}^{13} \bar{D}_{1..6}^{28}$	57
	${}_{1..3}^{14} D_{1..6}^{28}$	973 812	8 183	38 766	1 004 390	${}_{1..3}^{14} \bar{D}_{1..6}^{28}$	507
	${}_{1..3}^{13} D_{1..6}^{28}$					${}_{1..3}^{13} \bar{D}_{1..6}^{28}$	
Phase 4	$D_{1..6}^{28}$	564	1 345	68 254	41 956	$\bar{D}_{1..6}^{28}$	6
Phase 5	$D_{1..6}^{28}$	1	15 650	472 264	236 132	$\bar{D}_{1..6}^{28}$	1
	Σ	3 821 083	204 468	12 769 239	10 406 205		1 074 423

seconds), the number of function evaluations (NFE) and the number of executions of the ‘method of active areas’ (NMA, see Section 2.) when processing the elements of the corresponding D sets. The columns on the right show the name and the size of the output sets of the optimization process.

Phase 1: First we evaluated $\bar{D}_{1..3}^m$ for some m . The initial sets $D_{1..3}^m$ were built from full tile combinations, since we had no previous information about the possible configurations. We started the process by evaluating $\bar{D}_{1..3}^{14}$ and proceeded with $\bar{D}_{1..3}^{15}$ and $\bar{D}_{1..3}^{13}$. In the next step, $\bar{D}_{1..3}^{16}$ proved to be empty, which implied $\bar{D}_{1..3}^m = \emptyset$ for all $m \geq 16$ by the consequence of the central theorem of [10]. At the same time we considered the sets $\bar{D}_{1..3}^m$, $m \leq 12$ as combinations consisting of full tiles.

Phase 2: At this moment we had $\bar{D}_{1..3}^m$ for $0 \leq m \leq 21$. We evaluated ${}_{1..3}^{m'} \bar{D}_{1..4}^{m''}$ for all the required m', m'' values in two steps: first, we com-

puted ${}_{1..3}^{m'}D_{1..4}^{m''}$ (see [10]) and then we ran the optimization algorithm on these sets to obtain ${}_{1..3}^{m'}\bar{D}_{1..4}^{m''}$. Thus, in Phase 2 we added new regions chosen from column 4 in all the possible ways to the elements of $\bar{D}_{1..3}^m$.

Phase 3: It is easy to see that we could have proceeded with adding a new column to the result patterns of Phase 2. Instead, we have found that already at this point we had enough local information to merge possible patterns on the first four and last four columns by matching the remaining regions within columns 3 and 4. (The possible patterns on the last 4 columns (i.e. ${}_{4..6}^{m'}\bar{D}_{3..6}^{m''}$) were determined by simple geometric transformations from the sets ${}_{1..3}^{m'}\bar{D}_{1..4}^{m''}$.) In details, we determined ${}_{1..3}^{13}D_{1..6}^{28}$ by joining the elements of ${}_{1..3}^{13}\bar{D}_{1..4}^p$ and ${}_{4..6}^{15}\bar{D}_{3..6}^q$ and we determined ${}_{1..3}^{14}D_{1..6}^{28}$ by joining the elements of ${}_{1..3}^{14}\bar{D}_{1..4}^p$ and ${}_{4..6}^{14}\bar{D}_{3..6}^q$. (Notice that we did not have to evaluate ${}_{1..3}^{15}D_{1..6}^{28}$ since it consists of the symmetric cases of ${}_{1..3}^{13}D_{1..6}^{28}$. For symmetry reasons it was also enough to merge ${}_{1..3}^{14}\bar{D}_{1..4}^p$ with ${}_{4..6}^{14}\bar{D}_{3..6}^q$ only when $q \geq p$.) In this way we have obtained a relatively small set of combinations consisting of 28 regions.

Phase 4: The result sets ${}_{1..3}^{13}\bar{D}_{1..6}^{28}$ and ${}_{1..3}^{14}\bar{D}_{1..6}^{28}$ were joined to one set $D_{1..6}^{28}$ to be refined. Since this set could contain a lot of equivalent solutions, we still did not use the usual stopping criterion of the optimization algorithm, instead, we increased the maximal number of allowed iteration steps from 3 to 3000. As a result, we have managed to reduce the number of remaining combinations to 6.

Phase 5: Now it was possible to check all the 6 regions one-by-one. (However, for other n values much bigger number of combinations may remain, thus, we are planning to automatize this step in the future.) It was easy to see that four of the six combinations were leading to symmetric packings, since the initial tile combinations of them were symmetric. The fifth and sixth combination did not have this property, however, it was proved that if we rotate them by 90 degrees around the midpoint of the square, their transformed remaining regions fit into the initial tile combination of one of the first four remaining combinations. (The reason of the need of this investigation was that the 7×6 splitting is not invariant to rotations by ± 90 degrees.) Consequently, it was enough to consider one of the six combinations in the further investigations. This also means that the optimal packing of 28 points (circles) is unique in a certain sense: disregarding symmetric cases, all the optimizers are located in a small result box.

The only remaining combination was then refined by the optimization algorithm using the stopping criterion parameter of $\epsilon = 1e-4$. After transforming the result back to the original point packing problem, we have obtained the tightest currently known enclosure of the global maximum value, which is $F^* = [0.2305354936419110, 0.2305354937010030]$

Table II. The enclosure of the global maximizer point for packing of 28 points.

i	X_i	Y_i
1.	[<u>0.0000000000000000</u> , <u>0.0000000001340473</u>]	[<u>0.0000000000000000</u> , <u>0.0000000001720872</u>]
2.	[<u>0.0000000000000000</u> , <u>0.0000000009018277</u>]	[<u>0.2833357010171238</u> , <u>0.2833357021748722</u>]
3.	[<u>0.0000000000000000</u> , <u>0.0000000009469569</u>]	[<u>0.5138711946590346</u> , <u>0.5138711958167832</u>]
4.	[<u>0.0000000000000000</u> , <u>0.0000000007711232</u>]	[<u>0.7444066883009454</u> , <u>0.7444066894586941</u>]
5.	[<u>0.0000000000000000</u> , <u>0.000000000789306</u>]	[<u>0.9999999998814951</u> , <u>1.0000000000000000</u>]
6.	[<u>0.1818703763590640</u> , <u>0.1818703768084189</u>]	[<u>0.1416678505105635</u> , <u>0.1416678511731824</u>]
7.	[<u>0.1996495936336676</u> , <u>0.1996495987129519</u>]	[<u>0.3986034449660930</u> , <u>0.3986034489290318</u>]
8.	[<u>0.1996495936336676</u> , <u>0.1996495965685251</u>]	[<u>0.6291389386080037</u> , <u>0.6291389425706844</u>]
9.	[<u>0.1918713855226908</u> , <u>0.1918713859093646</u>]	[<u>0.8722033440944282</u> , <u>0.8722033447152648</u>]
10.	[<u>0.3637407527181273</u> , <u>0.3637408323617748</u>]	[<u>0.0000000000000000</u> , <u>0.0000000587212721</u>]
11.	[<u>0.3815199703326766</u> , <u>0.3815199723665520</u>]	[<u>0.2569355943756818</u> , <u>0.2569355979841080</u>]
12.	[<u>0.3992991877579223</u> , <u>0.4023815123711082</u>]	[<u>0.5089496079216517</u> , <u>0.5138711956474279</u>]
13.	[<u>0.3915209793722809</u> , <u>0.3915209798429083</u>]	[<u>0.7569355970687199</u> , <u>0.7569355978752804</u>]
14.	[<u>0.3837427710453809</u> , <u>0.3837427717920864</u>]	[<u>0.9999999993994661</u> , <u>1.0000000000000000</u>]
15.	[<u>0.5633903466917400</u> , <u>0.5633903920122968</u>]	[<u>0.1152677366531080</u> , <u>0.1152678049926778</u>]
16.	[<u>0.5839312813110917</u> , <u>0.5839312817689947</u>]	[<u>0.3672817571203574</u> , <u>0.3672817578530293</u>]
17.	[<u>0.5911705735259049</u> , <u>0.5911705737877094</u>]	[<u>0.6416678506253949</u> , <u>0.6416678510151504</u>]
18.	[<u>0.5833923648949710</u> , <u>0.5833923656955049</u>]	[<u>0.8847322523833678</u> , <u>0.8847322537245491</u>]
19.	[<u>0.7630399406653529</u> , <u>0.7630399518871784</u>]	[<u>0.0000000000000000</u> , <u>0.0000000013981615</u>]
20.	[<u>0.7694645054372355</u> , <u>0.7694645063580891</u>]	[<u>0.2304459563203037</u> , <u>0.2304459575945446</u>]
21.	[<u>0.7727203430882406</u> , <u>0.7727203431481791</u>]	[<u>0.4995893688385575</u> , <u>0.4995893689690468</u>]
22.	[<u>0.7830419591264131</u> , <u>0.7830419596109964</u>]	[<u>0.7694645058854756</u> , <u>0.7694645063580891</u>]
23.	[<u>0.7830419587445609</u> , <u>0.7830419596109966</u>]	[<u>0.9999999995273864</u> , <u>1.0000000000000000</u>]
24.	[<u>0.9935754343072639</u> , <u>0.9935754455065703</u>]	[<u>0.0000000000000000</u> , <u>0.000000003122235</u>]
25.	[<u>0.999999990791464</u> , <u>1.0000000000000000</u>]	[<u>0.2304459563203037</u> , <u>0.2304459565975328</u>]
26.	[<u>0.999999999530722</u> , <u>1.0000000000000000</u>]	[<u>0.4609814499622147</u> , <u>0.4609814502384716</u>]
27.	[<u>0.9999999995154167</u> , <u>1.0000000000000000</u>]	[<u>0.6915169436041255</u> , <u>0.6915169449529064</u>]
28.	[<u>0.999999991335642</u> , <u>1.0000000000000000</u>]	[<u>0.9220524372460362</u> , <u>0.9220524396576592</u>]

with $w(F^*) \approx 6 \cdot 10^{-11}$. The enclosure of the global maximizer is reported in Table II. The enclosure of the optimal packing of 28 points contains the previously known best packing [1, 16]. Note, that this packing has a point which place is not fixed (i.e. it represents an infinite number of packings with the same objective function value). Our result confirms this fact: the width of the 12th component indicates that the exact optimizer may really has this structure.

5. Summary and conclusions

We implemented a fully reliable, interval-based method for solving circle packing problems. We extended the group of problems (cf. [12],

Section 1.5) solved by mathematically rigorous computer-aided techniques by proving the global optimality of a previously known approximate packing configuration of 28 circles. Moreover, it was proved that apart from symmetric cases all the exact optimizers are located in a small result box. The total computational time was about 57 hours for the whole process, which is much smaller than the expected time requirement of the earlier methods.

References

1. R.L. Graham and B.D. Lubachevsky, Repeated patterns of dense packings of equal disks in a square, *Electronic Journal of Combinatorics* 3:R16 (1996).
2. C. de Groot M. Monagan, R. Peikert, and D. Würtz, Packing circles in a square: review and new results, in P. Kall (ed.): *System Modeling and Optimization – Proc. 15th IFIP Conf. Zürich, 1991*, Lecture Notes in Control and Information Services 180 (1992), 45-54.
3. C. de Groot, R. Peikert, and D. Würtz, The optimal packing of ten equal circles in a square, *IPS Research Report 90-12*, ETH, Zürich, 1990.
4. R. Hammer, M. Hocks, U. Kulisch, and D. Ratz, *Numerical Toolbox for Verified Computing I*, Springer-Verlag, Berlin, 1993.
5. O. Knüppel, PROFIL – Programmer’s Runtime Optimized Fast Interval Library, Bericht 93.4, Technische Universität Hamburg-Harburg, 1993.
6. O. Knüppel, A Multiple Precision Arithmetic for PROFIL, Bericht 93.6, Technische Universität Hamburg-Harburg, 1993.
7. M. Locatelli and U. Raber, Packing Equal Circles in a Square: a Deterministic Global Optimization Approach, *Discrete Applied Mathematics* 122 (2002), 139-166.
8. M.Cs. Markót, An Interval Method to Validate Optimal Solutions of the “Packing Circles in a Unit Square” Problems, *CEJOR* 8 (2000), 63-78.
9. M.Cs. Markót, T. Csendes, and A.E. Csallner, Multisection in Interval Branch-and-Bound Methods for Global Optimization II. Numerical Tests, *Journal of Global Optimization* 16 (2000), 219-228.
10. M.Cs. Markót and T. Csendes, A new verified optimization technique for the “packing circles in a unit square” problems. Submitted for publication. Available from <http://www.inf.u-szeged.hu/~markot/packcirc.htm>.
11. R.E. Moore, *Interval Analysis*, Prentice-Hall, Englewood Cliffs, 1966.
12. A. Neumaier, *Introduction to Numerical Analysis*, Cambridge Univ. Press, Cambridge, 2001.
13. K.J. Nurmela and P.R.J. Östergård, More Optimal Packings of Equal Circles in a Square, *Discrete and Computational Geometry* 22 (1999), 439-457.
14. R. Peikert, Dichteste Packungen von gleichen Kreisen in einem Quadrat, *Elemente der Mathematik* 49 (1994), 17-25.
15. H. Ratschek and J. Rokne, *New Computer Methods for Global Optimization*, Ellis Horwood, Chichester, 1988.
16. <http://www.packomania.com>, updated by Eckard Specht.
17. P.G. Szabó, Some New Structures for the “Equal Circles Packing in a Square” Problem, *CEJOR* 8 (2000), 79-91.