A Novel Genetic Algorithm for the Layout Optimization Problem

Yi-Chun Xu, Ren-Bin Xiao and Martyn Amos

***Abstract*— In this paper we present a new algorithm for the Layout Optimization Problem: this concerns the placement of circular, weighted objects inside a circular container, the two objectives being to minimize imbalance of mass and to minimize the radius of the container. This problem carries real practical significance in industrial applications (such as the design of satellites), as well as being of significant theoretical interest. We present a genetic algorithm solution and compare it with two existing nature-inspired methods, one of which is the best published algorithm for this problem. Experimental results show that our approach out-performs these existing methods in terms of both solution quality and execution time.**

I. Introduction

The *Layout Optimization Problem* (LOP) concerns the physical placement of instruments or pieces of equipment in, for example, a spacecraft or satellite. Because these objects have mass, the system is subject to additional constraints (beyond simple Cartesian packing) that affect our solution. The two main constraints that we handle in this paper are (1) the space occupied by a given collection of objects, and (2) the non-equilibrium (i.e. imbalance) of the system. The rest of the paper is organized as follows: we first present a detailed description of the problem, and describe previous related research. We then describe our own algorithm, and give the results of numerical experiments. These confirm that our method out-performs the previous best known algorithm for this problem.

II. Description of the Problem

The LOP was proposed by Teng *et al.* [1] in 1994, and has significant implications for the cost and performance of devices such as satellites and spacecraft. It concerns the *two dimensional* physical placement of a collection of *objects* (instruments or other pieces of equipment) within a spacecraft/satellite "cabinet", or *container* (Figure 1(a)). In what follows, we assume that each object is circular, with a *radius*, and a *mass*. The container region is also assumed to be circular (Figure 1(b)).

In short, we are required to pack, without overlaps, a collection of weighted circles within a single containing circle, whilst minimizing the radius of the container and ensuring that the overall distribution of mass is balanced. The LOP is demonstrably NP-hard [2][3]; it was first proposed by Teng *et al.* [1] and further developed by Feng *et al.* [4], who proposed a solution based on graph and group theory.

Y.-C. Xu is with the Institute of Intelligent Vision and Image Information, School of Electrical Engineering and Information Technology, China Three Gorges University, China (e-mail: xuyichun@tom.com); R.-B. Xiao is with the School of Electrical Engineering and Information Technology, China Three Gorges University, China (e-mail: rbxiao@163.com); M. Amos is with the Department of Computing and Mathematics, Manchester Metropolitan University, United Kingdom (e-mail: martyn@martynamos.com).

A. Previous Work

Several algorithms for the LOP have been proposed, some based on the Genetic Algorithm (GA) [5], [6], [7], and others on Particle Swarm Optimization (PSO) [8], [9]. However, these are all *iterative* methods. The positions of the circles, which we call the *initial layout*, are first randomly generated, and then a search method is used to improve the quality of the layout. However, the overall search space becomes very large when the number of circles increases only moderately, and, with little heuristic information available to them, it is hard for those methods to find high quality layouts.

In terms of general circle packing, in [10] the authors fit circles into a rectangular frame, where the circles are placed sequentially and the position of each individually calculated. The authors of [11] use a similar idea to pack circles into a containing *circle*. However, because packed objects lack mass within their problem definition, these algorithms cannot be directly applied to the LOP.

B. Overview

We first describe a novel *order-based* positioning technique (OPT) for the LOP. Given a *sequence* of circles, which defines their placement in a specific *order*, the OPT generates an initial layout in $O(n^4)$ time. The problem then becomes one of finding the best *ordering* of circles, rather than finding the optimal *direct placement* of each circle, as in previous work. Since there exist, for n circles, $n! = n \times (n-1) \times \cdots \times 2 \times 1$ sequences, we propose a *genetic algorithm* to search this permutation space. We now describe in detail the problem and its solution.

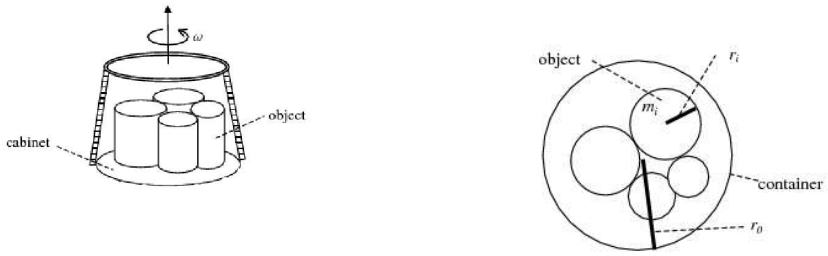
III. Definitions

Let $L = \{1, 2, \ldots, n\}$ denote a set of n circles (i.e., objects) with radii $r_1, r_2, \cdots, r_n$ and mass $m_1, m_2, \cdots, m_n$. Given a Cartesian coordinate system with some central origin, o (the center of the satellite containing circle, in our case), let (x_i, y_i) denote the position of circle i.

A. Previous Models

The models previously given in [5], [6], [7], [8], [9] may be described as follows:

1-4244-1340-0/07/$25.00



(a) (b)

Fig. 1. (a) Collection of objects within a containing cabinet. (b) Two-dimensional plan view.

$$\min f_1(L) = \max_{1\le i\le n}\left(r_i + \sqrt{x_i^2 + y_i^2}\right), \quad (1)$$

$$\min f_2(L) = \sqrt{(\sum_{i=1}^{n} m_i\omega^2 x_i)^2 + (\sum_{i=1}^{n} m_i\omega^2 y_i)^2}. \quad (2)$$

subject to:

$$\forall i, j = 1\ldots n,$$

$$\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \le r_i + r_j, \quad (3)$$

$$f_1(L) < r_0, \quad (4)$$

$$f_2(L) < \delta. \quad (5)$$

Equation (1) attempts to minimize f_1, the satellite radius containing all of the circles, L. Equation (2) attempts to minimize f_2, the non-equilibrium of mass for a collection of circles around the central point, L, where ω is the rotational speed of the satellite. As the value of ω does not affect the optimality of f_2, we set $\omega = 1$ in what follows.

Three constraints are applied: Equation (3) ensures that none of the circles can overlap any other; Equation (4) ensures that the satellite containing radius has an upper bound; Equation (5) ensures that the overall imbalance of mass is within some bound.

It is clear that the LOP is a *multi-objective* optimization problem. In previous work, two objectives are often combined into a *single* objective by a pair of weights λ_1 and λ_2 [5], [6], [7], [8], [9]:

$$\min f(L) = \lambda_1 f_1(L) + \lambda_2 f_2(L). \quad (6)$$

B. Our Model

In the "real world" domain of application for this problem, the mass imbalance that the system can tolerate (i.e., δ in (5)) is very small [5], [6], [7], [8], [9]. Therefore, in our model, we require the imbalance of the system be zero. Note that our requirement is stricter than that of (5). It seems intuitively obvious that this requirement may result in a suboptimal solution. However, the situation is not actually as bad as might be imagined. We know that the center of mass of the system is

$$o' = (\frac{\sum_{i=1}^{n} m_i x_i}{\sum_{i=1}^{n} m_i}, \frac{\sum_{i=1}^{n} m_i y_i}{\sum_{i=1}^{n} m_i}). \quad (7)$$

If we require the imbalance to be zero, we can shift every circle and let o' became the center of the satellite. If the envelopment radius of a layout is R, then the new radius *after* the shift $R' \le R + |oo'|$. From (2) and (5), the distance $|oo'| < \frac{\delta}{\sum_{i=1}^{n} m_i}$. Because in general δ is very small and $\sum_{i=1}^{n} m_i$ is very large, the distance $|oo'|$ can be omitted. This supports the assertion that we can require the imbalance of the system to be zero without significantly affecting the optimality of the final solution.

If the center of mass of the system is defined as the center of the satellite, the overall "envelopment" (i.e., containing radius) of the system becomes:

$$\textit{envelop}(L) = \max_{1\le i\le n} \left(r_i + \sqrt{(x_i - \frac{\sum_{j=1}^{n} m_j x_j}{\sum_{j=1}^{n} m_j})^2 + (y_i - \frac{\sum_{i=j}^{n} m_j y_j}{\sum_{j=1}^{n} m_j})^2}\right). \quad (8)$$

and our constrained optimization problem becomes:

$$\min \textit{envelop}(L), \text{ subject to:}$$

$$\forall i, j = 1\ldots n,$$

$$\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \le r_i + r_j, \quad (9)$$

$$\textit{envelop}(L) < r_0. \quad (10)$$

IV. The Order-Based Positioning Technique (OPT)

In previous work on the LOP, a random layout is generated and then iteratively improved until a satisfactory solution is obtained. Because the possible search space is so large, an algorithm's performance (in terms of quality) is strongly tied to the initial layout. In addition, the running times of such algorithms tend to be prohibitively large. Here, we demonstrate an alternative positioning technique which yields compact layouts in an efficient manner.

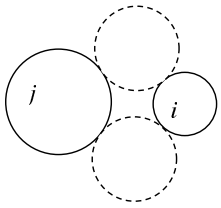


Fig. 2. Positions defined by pair of circles (i, j)

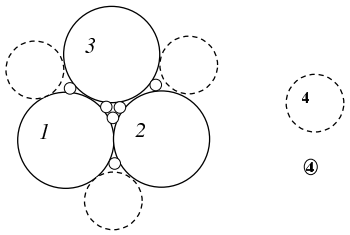


Fig. 3. Size of circle to be placed affects number of possible positions.

Let $(i_1, i_2, \ldots, i_n)$ be a permutation of $(1, 2, \ldots, n)$. We position our circles in L one by one in the order defined by this permutation. Suppose we have already placed the circles $(i_1, i_2, \ldots, i_{k-1})$: we now describe the strategy for placing circle i_k.

When positioning circle i_k (where $k > 2$, see Remark 1 below), we require that the circle should touch at least two previously positioned circles. This is a reasonable requirement, since immediate adjacency will generally yield a more compact layout than one defined by spatially separate circles. This can be explained well by a simple LOP example with two existing circles (Figure 2). Since two existing circles define two possible positions for a third i_k (these positions being denoted by the dashed lines), then $k-1$ existing circles can define $2 \times C_{k-1}^2$ positions. For example, when positioning circle i_4, we have three pairs, $(i_1, i_2), (i_1, i_3)$, and (i_2, i_3), giving six available positions for i_4.

The significant question we must address is how to select the best *position* for circle i_k from the $2 \times C_{k-1}^2$ available positions. We know that, for each additional circle, the envelopment radius as a whole is generally enlarged. In order to minimize the rate of growth of this radius, we must choose a position for circle i_k (from a possibly very large set of candidates) which yields the minimal envelopment radius. One approach is to apply a *greedy* policy.

The evaluation of the partial layout is performed in similar fashion to (8), but this time only k circles are used:

$$\textit{envelop}(\{i_1, i_2, \ldots, i_k\}) = \max_{i \in (i_1, i_2, \ldots, i_k)} \left(r_i + \sqrt{(x_i - \frac{\sum_{j=i_1, i_2, \ldots, i_k} m_j x_j}{\sum_{j=i_1, i_2, \ldots, i_k} m_j})^2 + (y_i - \frac{\sum_{j=i_1, i_2, \ldots, i_k} m_j y_j}{\sum_{j=i_1, i_2, \ldots, i_k} m_j})^2} \right). \quad (11)$$

Remark 1: Circles i_1 and i_2 must first be positioned. In what follows, we simply place them adjacent to one another, defining their positions as $(-r_{i_1}, 0)$ and $(r_{i_2}, 0)$.

Remark 2: Although we state that there are $2 \times C_{k-1}^2$ positions available for circle i_k, some of these positions may introduce an overlap. These positions are therefore excluded when we choose the position for i_k. This remark can be explained with reference to Figure 3, where we are attempting to place circle 4. If, in this particular case, circle 4 happens to be small (the lower of the two shown to the right of Figure 3), then there are 6 positions available (shown by solid circles). However, if circle 4 happens to be large, then only 3 positions are valid (shown by the dashed circles).

Remark 3: Because we assess $2 \times C_{k-1}^2$ positions when placing circle i_k, each time checking for overlaps $k-1$ times, then the total number of steps is about $2 \times \sum_{k=1}^{n} (C_{k-1}^2 \cdot (k-1))$, and the overall time complexity of the OPT is $O(n^4)$.

We now describe the OPT in pseudo-code:

Input: a permutation $(i_1, i_2, \ldots, i_n)$
Output: a layout defined by the permutation.
OPT
{
  Set $(x_{i_1}, y_{i_1}) = (-r_{i_1}, 0), (x_{i_2}, y_{i_2}) = (r_{i_2}, 0)$
  Repeat for each circle i_k, $2 < k \leq n$
  {
    Set $BestEnvelop = \infty$
    Repeat for each pair of circles (i_m, i_n),
    where $m < k, n < k, m \neq n$
    {
      Calculate the two positions (x^1, y^1) and (x^2, y^2) for i_k
      by (i_m, i_n), which satisfy the following:
        $(x - x_{i_m})^2 + (y - y_{i_m})^2 = (r_{i_k} + r_{i_m})^2$
        $(x - x_{i_n})^2 + (y - y_{i_n})^2 = (r_{i_k} + r_{i_n})^2$
      Test (x^1, y^1) and (x^2, y^2). If any of them do not cause
      overlaps, calculate the envelopment radius using (11),
      then update $BestEnvelop$ and store (x_{i_k}, y_{i_k}) in case
      it reduces the envelopment radius
    }
  }
  Output the layout $(x_{i_1}, y_{i_1}, \cdots, x_{i_n}, y_{i_n})$
}

We have described an efficient algorithm for generating reasonable initial layouts, given a sequence of circles to be placed. We now show how to search the (large) space of permutations to find a circle sequence that gives rise to an *optimal* layout.

V. Our Genetic Algorithm

GAs have been successfully applied to problems similar to the LOP (such as bin-packing [12], [13]), as well as, more generally, other NP-complete problems [14]. We now describe the operation of our own GA.

A. GA for the Layout Optimization Problem

As described in the previous Section, a permutation of $(1, 2, \ldots, n)$ can yield a layout by specifying the order in which circles are placed. As there exist $n!$ possible permutations for n circles, the GA is an appropriate technique to use in order to search such a large space. We should emphasize, for the purposes of clarity, that unlike in [6], where the GA was used to evolve the *positions* of each circle, we use a GA to evolve the *placement order* of each circle. We now describe specific aspects of our generational GA.

1) Coding Scheme: We use a permutation of $(1, 2, \ldots, n)$, $(i_1, i_2, \ldots, i_n)$, to code a individual. M individuals are randomly generated to form the initial population.

2) Fitness Function: We define fitness as the envelopment radius. To evaluate each individual, we first use OPT to obtain the layout specified by the individual, and then use (8) to get the fitness.

3) Crossover Operator: We apply a crossover operation that retains the validity of permutations (that is, each element in an individual's gene sequence is unique), as described in [13]. For convenience, we assume M is an even number, so we have $M/2$ pairs of parents. For each pair of parents, we apply the crossover operation to generate two children. In this paper, we apply crossover as follows: l_1 and l_2 are a pair of parent solutions. Suppose p is a random integer, where $1 \leq p \leq n$. The first child is generated by taking circles $1 \ldots p$ from l_1 and appending to this subsequence any missing circles *in the order in which they appear in* $l2$. The second child is obtained in the same way, only with with first subsequence taken from l_2, and the remainder being made up from l_1. For example, given $l_1 = (1, 2, 3, 4, 5)$ and $l_2 = (5, 4, 3, 2, 1)$, with $p = 3$, the two offspring sequences would be $(1, 2, 3, 5, 4)$ and $(5, 4, 3, 1, 2)$.

4) Mutation Operator: If an individual is selected for mutation, we randomly select two circles from its sequence and exchange their positions.

VI. Experimental Results

We compare our algorithm with an alternative evolutionary approach [6], and with an algorithm based on Particle Swarm Optimization [9], which is the best published algorithm for this problem to date. These other methods both search for the optimal layout by directly evolving the positions of every circle (as well as considering imbalance). We use the following benchmark suite of 10 problem instances to test the three algorithms:

1) Size= 10,
   Radii[ ]={20, 22, 17, 17, 7, 21, 11, 5, 23, 8},
   Mass[ ]={35, 61, 49, 89, 68, 80, 93, 82, 70, 20};
2) Size= 15,
   Radii[ ]={8,14,8,15,11,17,21,16,6,18,24,13,20,10,15},
   Mass[ ]={75,29,36,58,75,32,98,52,76,85,59,18,85,36,12};
3) Size= 20,
   Radii[ ]={20,24,8,11,13,7,7,15,24,18,15,17,17,14,16,18,5,21,21,13},
   Mass[ ]={86,72,81,54,29,94,92,41,57,77,40,67,31,47,39,61,73,83,11,20};
4) Size= 25,
   Radii[ ]={24,16,19,7,14,24,15,6,16,16,23,10,9,10,18,22,7,9,7,13,14,8,18,6,8},
   Mass[]={16,80,52,21,42,86,67,96,61,79,57,62,32,38,20,75,80,11,53,32,41,68,85,53,71};
5) Size= 30, Radii[ ]={14,15,11,19,9,6,23,9,23,13,24,12,24,24,10,8,9,8,6,11,6,16,24,12,9,19,13,24,21,18},
   Mass[ ]={24,52,37,17,12,19,51,67,23,46,14,96,55,84,21,92,69,65,72,36,73,83, 83,97,73,81,30,46,49,51};
6) Size= 35,
   Radii[ ]={10,20,13,19,19,10,14,14,24,11,20,15,7,18,22,10,13,12,21,14,9,10, 9,7,8,18,8,8,23,14,13,21,23,16,10},
   Mass[]={44,46,14,32,70,31,95,24,75,99,99,79,10,79,69,64,12,47,41,62,17,85, 43,70,43,63,44,57,62,20,17,80,47,68,19};
7) Size= 40,
   Radii[ ]={6,12,20,6,14,19,9,20,10,13,12,14,23,17,16,19,15,10,12,18,21,6,20,17,13, 20,17,6,21,15,12,9,14,20,23,16,23,9,23,18},
   Mass[ ]={74,48,16,35,19,58,87,90,17,29,32,63,46,76,26,88,71,49,89,14,68,94,41, 53,36,67,14,88,99,46,66,14,21,44,73,72,72,37,82,12};
8) Size= 45,
   Radii[ ]={13,8,11,21,9,20,24,20,17,21,7,13,24,7,6,8,18,15,12,18,17,21, 8,23,22,15,10,17,24,8,14,6,16,14,6,10,19,21,20,6,16,14,6,19,11},
   Mass[ ]={91,95,96,47,63,37,56,96,84,70,36,41,48,12,86,43,70,71,56,89,52,49,53, 82,42,35,11,82,88,58,74,16,91,57,26,39,48,68,72,69,27,44,25,99,96};
9) Size=50,
   Radii[ ]={9,17,5,15,24,23,12,9,5,13,7,18,19,21,7,18,18,24,12,23,22,13,5, 6,17,21,7,18,14,17,10,15,18,8,8,16,7,18,24,6,20,10,21,11,22,24,12,7,14,11},
   Mass[ ]={19,85,60,19,88,18,28,55,66,47,49,69,93,94,35,43,93,34,27,61,20,52,51,41, 98,85,82,89,54,43,54,94,80,99,41,41,63,28,19,53,11,78,65,10,98,43,78,24,84,16};
10) Size= 55,
    Radii[ ]={17,23,17,13,18,21,23,22,7,9,8,13,20,11,10,19,10,14,12,22,19,10,17,11,21, 8,15,16,19,21,17,19,8,6,13,13,14,19,18,23,20,24,24,13,13,19,7,6,10,8,8,10,24,19,24},
    Mass[ ]={97,62,28,36,97,58,13,21,40,97,79,90,62,47,64,23,23,95,99,44,71,79,52,59, 47,60,41,47,90,95,81,98,70,47,90,13,93,50,21,80,17,52,96,73,88,16,91,97,40,52,50,90,19,69,14};

We set the GA parameters as follows: generations = 100, population size = 20, mutation rate = 0.125. The system parameters of the other two algorithms are set according to the descriptions in [6], [9].

Our results are shown in Table I, with the best layouts generated by our algorithm shown in Figure 4. Each algorithm was run 10 times. Results are shown for best overall envelopment radius, R, obtained, average value of R, average unbalance, u, and average run time, t. All experiments were carried out on an Intel Celeron 1.5GHZ/256M PC.

Our findings clearly show that the GA provided by [6] performs worst out of the three algorithms on all 10 instances. The PSO algorithm [6] gets better results, and our GA performs best of all. Our algorithm out-performs the other two according to all four metrics: in terms of R we achieve an average performance improvement of over 30% over the other two algorithms. We consistently achieve zero imbalance, unlike the others, and our implementation generally runs 5 to 10 times faster in terms of machine time.

TABLE I

NUMERICAL RESULTS FOR 10 RUNS OF EACH ALGORITHM ON EACH OF 10 LOP INSTANCES.

	GA: Tang *et al.*[6]				PSO: Zhou *et al.* [9]				Our Algorithm			
LOP	Best R	$\overline{R}$	$\overline{u}$	$\overline{t}$	Best R	$\overline{R}$	$\overline{u}$	$\overline{t}$	Best R	$\overline{R}$	$\overline{u}$	$\overline{t}$
1	82.03	90.41	1.58	39621	61.32	64.08	0.0002	18401	60.41	60.96	0	466
2	135.43	154.62	1.93	68841	76.58	78.52	0.0002	31816	67.49	68.53	0	1536
3	167.78	172.76	5.93	97477	89.15	89.58	0.0002	47496	82.63	84.23	0	3593
4	161.15	177.51	4.19	142020	106.31	109.40	0.0002	63201	83.74	84.93	0	6393
5	180.73	200.46	2.16	192330	136.88	142.77	0.0004	87985	99.78	101.04	0	11061
6	193.23	206.74	5.47	254591	148.39	154.78	0.0004	112144	103.56	104.38	0	17274
7	214.86	223.14	3.78	322568	165.79	172.59	0.0004	138030	116.17	116.77	0	25421
8	215.85	228.20	1.69	400186	172.69	180.16	0.0004	202446	119.02	120.89	0	35499
9	214.93	228.10	2.97	536608	189.89	194.02	0.0005	192479	124.91	126.86	0	47716
10	225.13	233.58	1.36	595425	200.82	208.67	0.0003	236835	137.52	139.39	0	64363

VII. CONCLUSIONS

In this paper we provide a novel order-based positioning technique (OPT) for the layout optimization problem in satellites. We combine this method with a genetic algorithm to search the space of possible object orderings. Numerical results show that this algorithm out-performs the best published solution method for this problem, in terms of both solution quality and execution time. Future work will focus on the packing of irregular (as opposed to round) shapes within satellite bodies.

ACKNOWLEDGMENTS

We would like to thank the authors of [6] and [9] for sharing their code. This work was supported by the National Natural Science Foundation of China (No. 60474077).

REFERENCES

[1] H. Teng, S. Sun, W. Ge, and W. Zhong, "Layout optimization for dishes installed on a rotating table," *Science in China (A)*, vol. 37, no. 10, pp. 1271–1280, 1994.

[2] R. Fowler, M. Paterson, and S. Tanimoto, "Optimal packing and covering in the plane are NP-complete," *Information Processing Letters*, vol. 12, no. 3, pp. 133–137, 1981.

[3] H. Wang, W. Huang, Q. Zhang, and D. Xu, "An improved algorithm for the packing of unequal circles within a larger containing circle," *European Journal of Operational Research*, vol. 141, pp. 440–453, 2002.

[4] E. Feng, X. Wang, X. Wang, and H. Teng, "An algorithm of global optimization for solving layout problems," *European Journal of Operational Research*, vol. 114, pp. 430–436, 1999.

[5] Z. Qian, H. Teng, and Z. Sun, "Human-computer interactive genetic algorithm and its application to constrained layout optimization," *Chinese Journal of Computers*, vol. 24, no. 5, pp. 553–559, 2001.

[6] F. Tang and H. Teng, "A modified genetic algorithm and its application to layout optimization," *Journal of Software*, vol. 10, pp. 1096–1102, 1999.

[7] Y. Yu, J. Cha, and X. Tang, "Learning based GA and application in packing," *Chinese Journal of Computers*, vol. 24, no. 12, pp. 1242–1249, 2001.

[8] N. Li, F. Liu, and D. Sun, "A study on the particle swarm optimization with mutation operator constrained layout optimization," *Chinese Journal of Computers*, vol. 27, no. 7, pp. 897–903, 2004.

[9] C. Zhou, L. Gao, and H. Gao, "Particle swarm optimization based algorithm for constrained layout optimization," *Control and Decision*, vol. 20, no. 1, pp. 36–40, 2005.

[10] J. George, J. George, and B. Lamar, "Packing different-sized circles into a rectangular container," *European Journal of Operational Research*, vol. 84, pp. 693–712, 1995.

[11] W. Huang, Y. Li, C. Li, and R. Xu, "New heuristics for packing unequal circles into a circular container," *Computers & Operations Research*, vol. 33, pp. 2125–2142, 2006.

[12] B. Kroger, "Guillotineable bin packing: A genetic approach," *European Journal of Operational Research*, vol. 84, pp. 645–661, 1995.

[13] D. Liu and H. Teng, "An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles," *European Journal of Operational Research*, vol. 112, pp. 413–420, 1999.

[14] K. A. De Jong and W. M. Spears, "Using genetic algorithms to solve NP-complete problems," in *Proceedings of the Third International Conference on Genetic Algorithms*, J. D. Schaffer, Ed. Morgan Kaufmann, 1989, pp. 124–132.

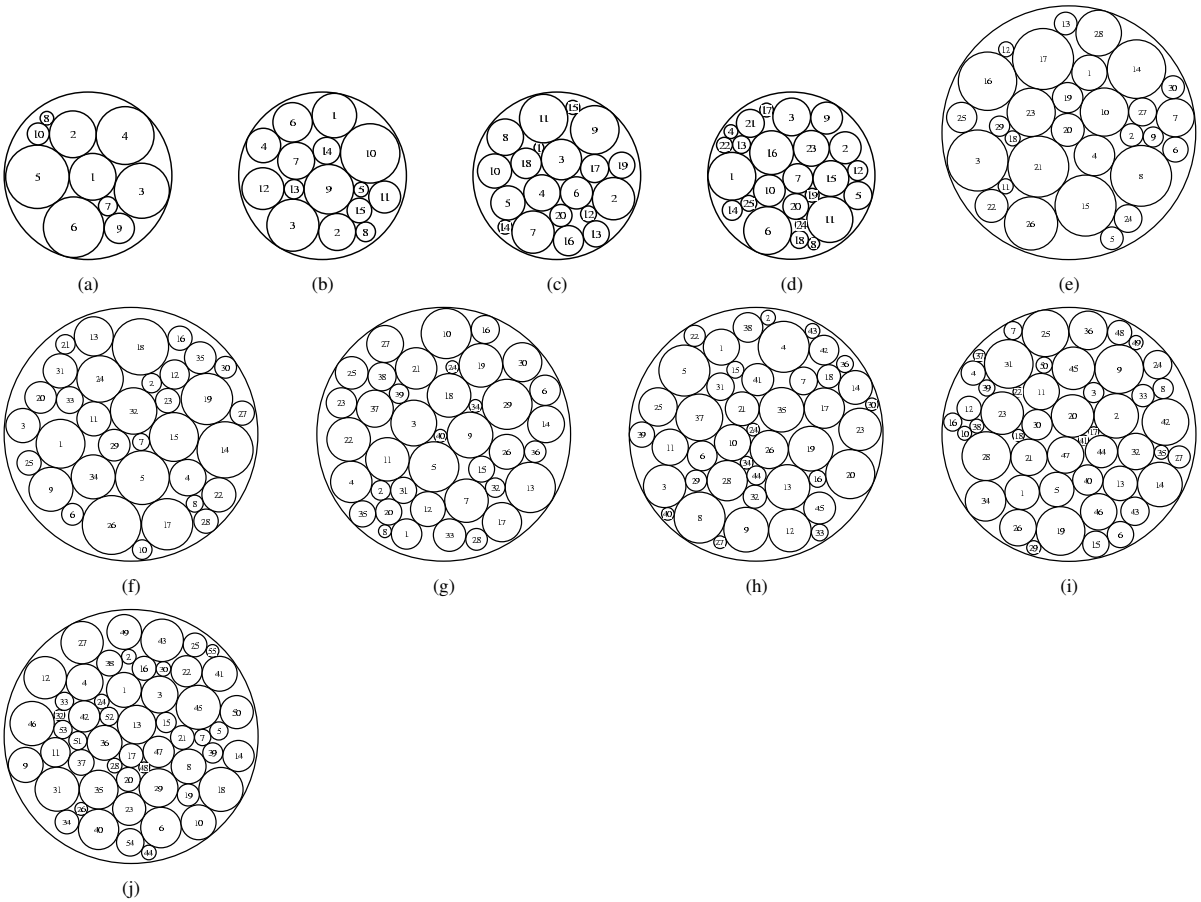


Fig. 4. Best solution found for each of the 10 instances. (a) R=60.41 (b) R=67.49 (c) R=82.63 (d) R=83.74 (e) R=99.78 (f) R=103.56 (g) R=116.17 (h) R=119.02 (i) R=124.91 (j) R=137.52.