Problem-specific knowledge based artificial bee colony algorithm for the rectangle layout optimization problem in satellite design

Yichun Xu[1,2],Shuzhen Wan[2],Fangmin Dong[2]
[1]*Hubei Province Engineering Technology Research Center for Construction Quality Testing Equipments*
[2]*College of Computer and Information Technology*
China Three Gorges University, Yichang 443002, Hubei, China
xuyichun@ctgu.edu.cn, wanshuzhen@163.com, fmdong@ctgu.edu.cn

***Abstract*—The layout optimization problem is brought from the design of the recoverable satellite, where a set of objects (equipments or devices) are required to be installed on a circular load board. The aim of the problem is to find a layout of the objects with no interference, less unbalance, and less space occupied. Artificial bee colony (ABC) algorithms show good performance in many engineering problems. In this article, based on the analysis of the solution distribution, a problem-specific knowledge based ABC is proposed, which is configured with special initialization and parameter settings. On an open benchmark with ten instances, the proposed ABC is compared with two widely used algorithms. Its performance outperforms the genetic algorithm on all the instances, and outperforms the quasi-human algorithm on nine instances.**

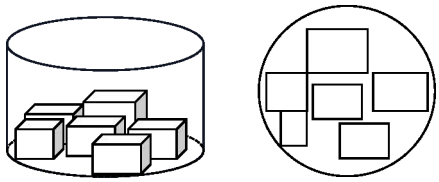
***Keywords*—swarm intelligence; artificial bee colony algorithm; layout optimization problem; weighted rectangle packing**

I. Introduction

In the design of the recoverable satellite, some objects (devices or equipments) are required to be installed on a circular load board (Fig. 1). Three kinds of constrains or objectives should be concerned: 1. There should be no interference between objects. 2. The layout of the objects should be compact so that they occupy less space. 3. The unbalance of the system should be small enough, so the system is easy to control. In this article, we study the two-dimensional problem that the shapes of the objects are modeled as rectangles, which is called the rectangle layout optimization problem (RLOP).

RLOP was first proposed in [1], where the authors studied the isomorphism of the layouts by graph theory and group theory, and then proposed a global optimization framework. From then on, some meta-heuristics were proposed for this problem, such as the genetic algorithms (GA)[2], the particle swarm optimization (PSO) [3, 4], the simulated annealing algorithm (SA) [5], and the ant colony optimization (ACO) [6]. Another class of algorithms are based on the quasi-physics and quasi-human strategies [7], that an elastic potential energy function is defined to measure the overlaps between objects, and then the overlaps are reduced by the elastic force step by step. Recently, [8] proposed a three-dimensional model, that the items have height and there are multiple layers to install the items. Except for assigning a layer for each item, their layout optimization algorithm is basically an application of the algorithm for two-dimensional models. In general, the existing algorithms are with good results on the small-scale RLOP. With the increase of the rectangle numbers, they become time consuming and the solutions decline in quality.

DOI reference number: 10.18293/SEKE2022-014



(a) 3D model (b) 2D model

Fig. 1: Installing equipments on the load board of a recoverable satellite

This article aims at designing a new algorithm for RLOP with the help of some "guiders". The main idea is based on the observation that in a good layout, the bigger and the heavier objects often locate at the center of load board. This observation leads us to design a new greedy strategy. We then combine the greedy strategy into an artificial bee colony (ABC) framework. The artificial bee colony (ABC) algorithm is a kind of nature-inspired optimizer in swarm intelligence proposed by Karaboga [9]. Because ABC algorithm needs fewer parameters and the performance is often good, it is very popular in most engineering fields [10].

On an open benchmark with ten instances, the performance of the proposed ABC outperforms the widely used genetic algorithm [2] and the quasi-human algorithm [7].

II. Mathematical model

In two-dimensional case, we need to pack a set of rectangles with masses into a containing circle as Fig. 1(b). For the constraints, the rectangles cannot overlap each other, and the center of mass of the system should be near the center of the containing circle to keep the equilibrium. The aim of the

problem is to minimize the envelopment circle which covers all the rectangles. The problem can also be stated as follows.

Define n rectangles by a list $R = (l_1, w_1, m_1), (l_2, w_2, m_2), \ldots, (l_n, w_n, m_n)$, where l_i, w_i, and m_i are the length, the width, and mass of rectangle i. Assume the center of mass and shape is located at the same point in each rectangle. In a two-dimensional Cartesian coordinate system, we set the Cartesian origin to the center of the containing circle. The list $X = (x_1, y_1, \theta_1), (x_2, y_2, \theta_2), \ldots, (x_n, y_n, \theta_n)$ denotes a layout, where x_i, y_i is the center of the rectangle i, and θ_i denotes its orientation. The aim of the problem is to find a layout X to satisfy the following constraints:

1) $\theta_i \in \{0, 1\}$, where θ_i=0 or 1 mean that the edge with length l_i is parallel or perpendicular to the x axis, then the items are placed orthogonal to each other.
2) There is no overlap between any two rectangles, that is, for all $i \neq j$, at least one of the following conditions should be satisfied:

$$x_i + l'_i/2 \leq x_j - l'_j/2 \quad (1)$$
$$x_i - l'_i/2 \geq x_j + l'_j/2 \quad (2)$$
$$y_i + w'_i/2 \leq y_j - w'_j/2 \quad (3)$$
$$y_i - w'_i/2 \geq y_j + w'_j/2 \quad (4)$$

where l'_i and w'_i are related to the length and width after considering the orientation θ_i, which satisfy

$$l'_i = l_i(1 - \theta_i) + w_i\theta_i \quad (5)$$
$$w'_i = w_i(1 - \theta_i) + w_i\theta_i \quad (6)$$

3) The center of mass of all rectangles should be located at the center of the circle for equilibrium, that is, given a small positive permissible value of δ

$$(x_w, y_w) = \left(\frac{\sum_{i=1}^{i=n} m_i x_i}{\sum_{i=1}^{i=n} m_i}, \frac{\sum_{i=1}^{i=n} m_i y_i}{\sum_{i=1}^{i=n} m_i}\right) \quad (7)$$
$$\sqrt{x_w^2 + y_w^2} \leq \delta \quad (8)$$

such that the radius r of the envelopment circle is minimized, where

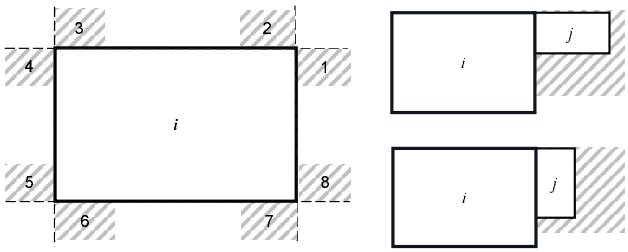
$$r = \max_{1 \leq i \leq n} \left(\sqrt{(|x_i| + l'_i/2)^2 + (|y_i| + w'_i/2)^2} \right). \quad (9)$$

III. ABC ALGORITHM

Before introducing the design of ABC for RLOP, we first give out a constructive heuristic to compose a layout, which is an important building block.

A. Constructive kernel heuristic (CKH)

The ABC algorithm is based on a constructive heuristic first appeared in [2]. At first, all the rectangles are waited in a queue, and the first rectangle is packed in the center of the circle. When packing a rectangle, for the goal of minimizing envelopment radius, we require it close to an already packed rectangle. As in Fig. 2(a), an already packed rectangle i provides 8 regions along its edges and vertices. A rectangle j to be packed should choose a region with an orientation



(a) regions (b) placement

Fig. 2: The regions provided by a rectangle(a) and the placement of a rectangle in a region with different orientation(b)

(Fig. 2(b)), so there are totally 16 schemes to place j beside i. After deletion of the schemes with overlap, there are still many feasible schemes besides all the packed rectangles, and the algorithm will select one by a greedy strategy–the region leads to the minimal temporary envelopment circle will be chosen.

B. Inversion count and distribution of the solutions

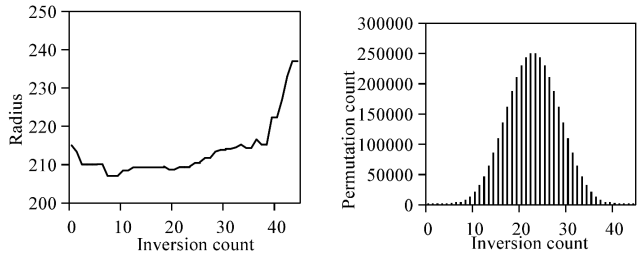
The application of CKH in this article is different from [2]. In the CKH, the output layout X is dependent on the permutation p. Because there are $n!$ permutations in total, the blind search such as [2] has very low efficiency. According to the computational practice, we find that the bigger and heavier objects should be placed first, which means a greedy permutation $\boldsymbol{g}$ in the descending order of $l_i w_i m_i$ can lead to a good layout. This finding relates the concept of inversion count to the goodness of a permutation.

For a permutation p, an inversion exists between the items p_i and p_j, if $l_{p_i} w_{p_i} m_{p_i} < l_{p_j} w_{p_j} m_{p_j}$ and $i < j$. According to the number theory, the inversion count of a permutation ranges from 0 to $\frac{n(n-1)}{2}$, and the greedy permutation g has a inversion count of 0.

By the computational experiences, a permutation with small inversion count often results in a better layout. On a randomly chosen RLOP instance with 10 rectangles (larger instance is in similar situation), we enumerate all the permutations and get the corresponding layouts and their envelopment radii by CKH. The minimal radii from the permutations with same inversion count are plotted in Fig. 3(a). The number of permutations with the same inversion count are illustrated in the histogram Fig. 3(b). We found that the permutations with inversion count less than 10 are obvious have better results, Moreover, the number of such types of permutations is relatively small that it is easy to search them.

C. ABC algorithm based on inversion count

ABC algorithm is a meta-heuristic based on the foraging behavior of honey bees. There are three kinds of bees in a colony. The employed bees work on a food source and share the information with the onlooker bees by dancing. The onlooker bees select a food source after watching the dance and try to improve it. When the food source is exhausted, the



(a) minimal radius (b) histogram

Fig. 3: The distribution of the solutions

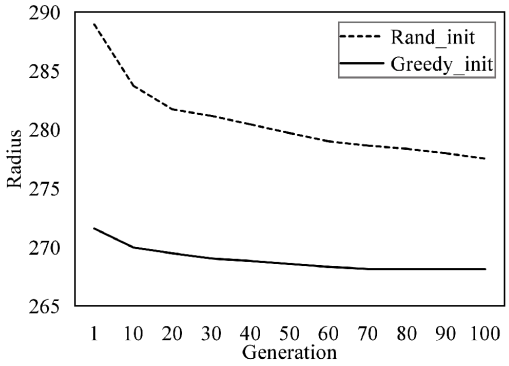


Fig. 4: Results on different initialization

related employed bee becomes a scout bee, who tries to find a new resource. The ABC algorithm for the RLOP is described in algorithm 1.

Algorithm 1 ABC algorithm

```
Initialize the first generation s_1, s_2, ..., s_n by the greedy strategy
for generation = 2 to G do
    for i=1 to n do {Employed phase}
        s_i:=best of (s_i, INSERT(s_i))
    end for
    for i=1 to n do {Onlooker phase}
        Randomly select a s_k with probability of P(s_k)
        s_k:=best of (s_k, INSERT(s_k))
    end for
    for i=1 to n do {Scout phase}
        if s_i is not improved for T generations then
            Restore s_i from the first generation
        end if
    end for
end for
return the best solution
```

1) Individual and fitness function: The individual solution s_i is a permutation p, which can be evaluated by the fitness function as (10), where $r(s_i)$ is the envelopment radius result of CKH.

$$f(s_i) = \frac{1}{r(s_i)} \tag{10}$$

2) first generation: The first generation defines the start points of the search. Based on the analysis in section III-B, we should focus on the permutations with small inversion count. We initial the fist generation with the greedy permutation g and other $n-1$ permutations generated by swapping the adjacent elements of g. The inversion count of the first generation is less than or equal to 1.

3) Mutation operator : In the employed phase, we choose the INSERT mutation operator like the genetic algorithm [11]. In a permutation p, after a block of $p_i, p_{i+1}, \ldots, p_{i+k-1}$ is chosen, the INSERT operator moves p_{i+k-1} before p_i. The mutation operator can change the inversion count of the permutation. Moreover, the larger the block size k, the more the inversion count is changed. So the block size k should have an upper bound.

4) Selection probability: The selection probability for an individual in the onlooker phase is proportional to its fitness value, which is defined as (11)

$$P(s_i) = \frac{f(s_i)}{\sum_{i=1}^{n} f(s_i)} \tag{11}$$

IV. Computational results

The numerical experiments were on a Dell OptiPlex 7080 Tower, with a 3.10 GHz Intel i5-10500 CPU,16 GB RAM, Win10 OS. And the programs were compiled by Dev C++ 5.9.2. By the experiences, the block size of INSERT is set to 5, and the initialization interval T in the scout phase is set to 10.

A. Experiment 1: Comparison of greedy initialization and random initialization

This experiment is to compare the greedy initialization and the popular random initialization in ABC. 30 instances with 20 rectangles for each were randomly generated that w_i, l_i are in range of [1,200] and m_i is near $w_i l_i$ for each rectangle i. In the experiment, the average radii of the 30 instances were recorded in each generation. The convergence curves are provided in Fig. 4. The results show that the greedy initialization is more advantage than the random initialization. Even after 100 generations of search, the average radii with random initialization is still worse than the start point of the greedy initialization. The ABC algorithm with random initialization wastes too much search energy in the subspace of permutation with larger inversion count.

B. Experiment 2: Numerical computation on an open benchmark with large-scale instances

In this experiment, we ran the ABC algorithm on a benchmark provided in [7], that there are 10 test instances R1, R2, ..., R10, and the numbers of rectangles are 10, 20, ..., 100 respectively. Two algorithms in the literatures were selected as the baselines, which are the quasi-human algorithm (IBF) in [7] and the genetic algorithm (GA) in [2].

TABLE I: Results on 10 instances

inst	rtarget	GA		IBF		ABC	
		Fail	time(s)	Fail	time(s)	Fail	time(s)
R1	36.90	5	/	0	113.79	5	/
R2	65.30	5	/	4	746.96	0	378.16
R3	56.05	5	/	5	/	0	530.71
R4	75.30	5	/	5	/	0	50.48
R5	91.55	5	/	5	/	1	317.02
R6	101.79	5	/	5	/	0	88.18
R7	102.89	5	/	5	/	0	128.74
R8	109.09	5	/	5	/	0	407.82
R9	115.32	5	/	5	/	0	1628.23
R10	124.10	5	/	5	/	1	881.53

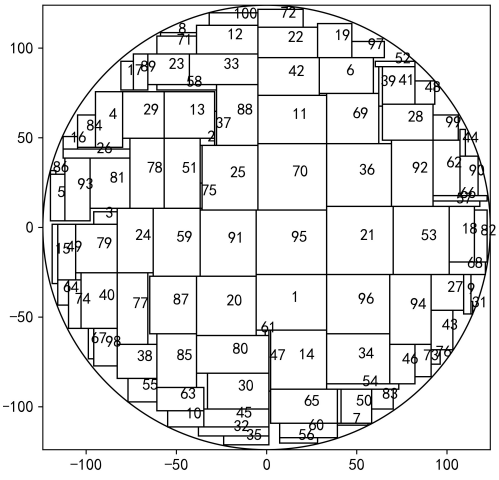
IBF uses the container radius as an input and its objective is to find a layout smaller than the given radius. To make a fair comparison with IBF, we defined a target container radius (**rtarget**) for each instance, and then used the time of finding a valid layout as the metric of performance. We set the stopping criterion of all the three algorithm as obtaining a layout with envelopment radius less than the target, or execution time exceeding an hour. The other parameters of the baselines were set as the literatures. In the ABC, the number of individual n was set to the number of rectangles in each instance. The three algorithms were executed on each instance for 5 times. If an execution did not output a layout in an hour, we marked it as one '**failure**'. Excluding the failed executions, we provide the average time of the rest executions on each instance. The results are listed in Table I and we give the layout diagram of the instance R10 by ABC in Fig. 5.

From the detailed data in Table I, GA fails in all the tests and shows the worst performance among the three algorithms. The reason why GA loses is that it tries to search the whole solution space and wastes much energy on the space with poor solutions, while the proposed ABC makes the search around the greedy solution, many good results are in this subspace.

IBF only passes the first 2 smaller instances, that it gets best results in R1, and gets a layout for R2 after 4 failures. It fails all the tests on the last 8 instances. But on the other side, ABC gets the best results in the rest 9 instances except R1, and passes 43 tests among the total 45 tests. The only two failures in tests of instance 5 and 10 should be because we set harder targets. ABC's failures in R1 is because of the shortcoming of CKH. It places the objects at certain positions, which restricts the pattern of the solution, so it may miss the optimal layout of smaller instances.

V. Conclusion

A problem-specific knowledge based artificial bee colony (ABC) algorithm for the layout optimization problem in the satellite design is presented in this article. After the investigate of the distribution of the solutions, the ABC algorithm is designed to search the most likely subspace containing high quality solutions, so that it can easily find a good solution in short time. On an open benchmark with 10 instances, ABC algorithm is compared with two widely used algorithms. It outperforms the genetic algorithm on all the instances, and outperforms the quasi-human algorithm on nine of them. The proposed ABC algorithm may have great practical value to find the rational layout of the objects in the aerospace industry.



(a) R10

Fig. 5: Layout diagram of ABC on R10

References

[1] E. Feng, X. Wang, X. Wang, and H. Teng, "A global optimization agorithm for layout problems with behavior constraints," *Applied Mathematics, A Journal of Chinese Universities*, vol. 14, no. 1, pp. 98–104, 1999.

[2] Y. Xu, F. Dong, Y. Liu, and R. Xiao, "Genetic algorithm for rectangle layout optimization with equilibrium constraints," *Pattern Recognition and Artificial Intelligence*, vol. 23, no. 6, pp. 794–801, 2010.

[3] Y.-C. Xu, R.-B. Xiao, and M. Amos, "Particle swarm algorithm for weighted rectangle placement," in *the 3rd Int'l Conf. on Natural Computation*, pp. 728–732, 2007.

[4] Z. Huang and R. Xiao, "Hybrid algorithm for the rectangular packing problem with constraints of equilibrium," *Journal of Huazhong University of Science and Technology (Natural Science Edition)*, vol. 9, no. 3, pp. 96–99, 2011.

[5] Y.-C. Xu, R.-B. Xiao, and M. Amos, "Simulated annealing for weighted polygon packing." https://arxiv.org/abs/0809.5005, 2008.

[6] M. Ji and R. Xiao, "Ant colony optimization and heuristic algorithms for rectangle layout optimization problem with equilibrium constraints," *Journal of Computer Applications*, vol. 30, no. 11, pp. 2898–2901, 2010.

[7] J. Liu, J. Li, Z. Lv, and Y. Xue, "A quasi-human strategy-based improved basin filling algorithm for the orthogonal rectangular packing problem with mass balance constraint," *Computers and Industrial Engineering*, vol. 107, pp. 196–210, 2017.

[8] C.-Q. Zhong, Z.-Z. Xu, and H.-F. Teng, "Multi-module satellite component assignment and layout optimization," *Applied Soft Computing*, vol. 75, pp. 148–161, 2019.

[9] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Tech. Rep. tr062005, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[10] B. Akay, D. Karaboga, B. Gorkemli, and E. Kaya, "A survey on the artificial bee colony algorithm variants for binary, integer and mixed integer programming problems," *Applied Soft Computing*, vol. 106, no. 3, p. 107351, 2021.

[11] M. Serpell and J. Smith, "Self-adaptation of mutation operator and probability for permutation representations in genetic algorithms," *Evolutionary Computation*, 2010.