Pesquisa Operacional (2024) 44: e284365 p.1-30
doi: 10.1590/0101-7438.2023.043.00284365
© 2024 Brazilian Operations Research Society
Printed version ISSN 0101-7438 / Online version ISSN 1678-5142
www.scielo.br/pope
ARTICLES

A BIASED RANDOM-KEY GENETIC ALGORITHM FOR THE 2D STRIP PACKING PROBLEM WITH ORDER AND STABILITY CONSTRAINTS

Santosh Kumar Mandal[1*], Thiago Alves de Queiroz[2] and Flávio Keidi Miyazawa[3]

Received March 11, 2024 / Accepted July 17, 2024

ABSTRACT. This paper deals with the two-dimensional strip packing problem (2D-SPP) with the order/or multi-drop and vertical stability constraints. The existing exact algorithm that solves this problem is not able to provide optimal solutions on large instances in a reasonable amount of time. Hence, with a view to quickly obtain a physically stable packing of minimum height while satisfying the order constraint, the Biased Random-Key Genetic Algorithm (BRKGA) is combined with Bottom-Left-Fill (BLF) and Open Space (OS) heuristics. Both versions of the algorithm (BRKGA + BLF and BRKGA + OS) retrieved optimal solutions on many benchmark instances, consuming lesser computational time than the exact algorithm. A comparative study was also performed between the BRKGA, Simulated Annealing (SA) and Particle Swarm Optimization (PSO) algorithms on newly generated large instances. Moreover, the effectiveness of the BRKGA has also been checked on the classical 2D-SPP and two-dimensional orthogonal packing problem (2D-OPP) datasets.

Keywords: multi-drop requirements, vertical stability, Biased Random-Key Genetic Algorithm.

1 INTRODUCTION

The strip packing problem, owing to its large industrial applications in different fields, is still drawing considerable attention of researchers around the world. In this paper, the two-dimensional strip packing problem (2D-SPP) is under consideration. In the 2D-SPP, a strip of finite width and virtually infinite height and a set of rectangular items are given. The goal is to

*Corresponding author

[1]Institute of Computing, University of Campinas (UNICAMP), Campinas, SP, Brazil – E-mail: santosh.nifft@gmail.com – https://orcid.org/0000-0002-2108-304X

[2]Institute of Mathematics and Technology, Federal University of Catalão, Catalão, Goiás, Brazil – E-mail: taq@ufcat.edu.br – https://orcid.org/0000-0003-2674-3366

[3]Institute of Computing, University of Campinas (UNICAMP), Campinas, SP, Brazil – E-mail: fkm@ic.unicamp.br – https://orcid.org/0000-0002-1067-6421

orthogonally pack all these rectangular items into the strip such that no two items overlap and the packing height is minimized. The 2D-SPP has different variants depending on the orientation and guillotine constraints. The problem addressed in this paper is subtype “OF” (Lodi et al., 1999) in which the items have a fixed orientation and no guillotine cutting is required.

A survey on the 2D-SPP can be found in the recent study conducted by Júnior et al. (2022). It is observed that most existing works on the 2D-SPP in fact do not consider real-life practical constraints. One such constraint is called the *vertical stability*. The vertical stability constraint ensures that after being packed, the items do not rotate or fall down due to the force of gravity. The other practical constraint that naturally arises in packing problems is the *order constraint*. It is also referred as the multi-drop or *Last In First Out* (LIFO) constraint. Assuming that items will be unloaded from the up-side of the strip, the order constraint states that if the order of item i is greater than the order of item j, then item i must not block the way out when unloading item j. A somewhat similar situation occurs in the capacitated vehicle routing problem with two-dimensional loading constraints (2L-CVRP) (Wei et al., 2018). In a vehicle route, while unloading/delivering items of one client, there should not exist items of other clients ahead on the route blocking the way out of the items of the current client (Silva et al., 2022).

This paper considers both the order and stability constraints while solving the 2D-SPP. Henceforth, the considered problem will be termed as two-dimensional strip packing problem with order and stability constraints (2SPOS). A typical example of the 2SPOS is the problem of loading freight trains, where loading the freight cars (or the containers used in double-stack cars) can be seen as a loading of pallets satisfying the order and static stability constraints (Queiroz & Miyazawa, 2014). The 2D-SPP is NP-hard (Hochbaum & Maass, 1985), and the consideration of the order and stability constraints makes the 2SPOS even more complex to be solved.

The existing exact algorithm for the 2SPOS is computationally expensive and could solve only small-sized instances. Meta-heuristics are fast and have already shown their strengths in obtaining (near) optimal solutions on large packing problems. These techniques are commonly combined with a placement algorithm (decoding algorithm) for solving cutting and packing problems (Soke & Bingul, 2006). Motivated by these facts, in this study the Biased Random Key Genetic Algorithm (BRKGA) with Bottom-Left-Fill (BLF) and Open Space (OS) placement heuristics is implemented for the 2SPOS. The reason for choosing the BRKGA is its success in solving a variety of hard optimization problems, including packing problems (Gonçalves & Resende, 2013; Junior et al., 2020). There exist many decoding algorithms in the literature for packing problems. They pack items following different rules, which generate packings of different heights. In the present case, one decoding algorithm may create a stable packing and the other may not. Therefore, in this work, two different decoding algorithms (BLF and OS) were utilized. Another widely used meta-heuristics for solving complex combinatorial optimization problems are the Simulated Annealing (SA) and Particle Swarm Optimization (PSO). There have also been many successful applications of SA and PSO for cutting and packing problems. For example, Burke et al. (2009) achieved very good solutions for the two-dimensional rectangular cutting stock problem using the SA algorithm. Omar & Ramakrisnan (2013) successfully applied an evolutionary PSO for solving the non-oriented two dimensional bin packing problem. In this work, therefore, SA and PSO have also been tested for the 2SPOS.

In order to evaluate the performance of the BRKGA, comparative studies have been performed on a variety of datasets. First of all, the performance of both versions of the BRKGA (BRKGA + BLF and BRKGA + OS) is checked against the exact algorithm of Queiroz & Miyazawa (2014) on the available 2SPOS dataset. Following this, the BRKGA + BLF/OS, SA + BLF/OS and PSO + BLF/OS algorithms are compared with each other on newly generated large 2SPOS instances. The BRKGA + BLF outperforms all others on the 2SPOS datasets; hence, lastly its effectiveness is also examined on the classical 2D-SPP and 2D-OPP benchmark datasets. The 2D-OPP (two-dimensional Orthogonal Packing Problem) consists in determining if a set of rectangular items can be packed in the strip of a given length size.

The 2D-OPP appears, for example, as a sub-problem of the 2D-SPP and 2L-CVRP. Therefore, having good algorithms for the 2D-SPP means having good algorithms for the 2D-OPP, which in turn, can allow solving the 2L-CVRP efficiently. In the 2L-CVRP, considering practical constraints is a must while approaching real logistic situations, e.g., considering the order (multi-drop requirements), load balancing, load stability, among others. Hence, in this work, 2SPOS is tackled with a view to develop a good and fast algorithm for it. The proposed BRKGA generates very good solutions for 2SPOS as well as for 2D-SPP and 2D-OPP in reasonable computational time, which justifies the present work.

The rest of the paper is organized as follows. The next section presents a literature review on the 2D-SPP. Section 3 introduces the 2SPOS in detail, including a discussion on the stability. In section 4, the workings of all the proposed algorithms are explained in detail. In section 5, performance of algorithms is assessed on the basis of obtained results on multiple datasets. Lastly, the conclusion and possible extensions of this work are presented in section 6.

2 LITERATURE REVIEW

A large number of researchers have contributed to solving the 2D-SPP. Due to high complexity of the problem, heuristics and meta-heuristics have mainly been preferred. Baker et al. (1980) proposed the well-known Bottom-Left (BL) heuristic, which places a rectangle at the lowest possible position and left-justifying it. The BL is fast, but unable to fill "holes" (empty spaces surrounded by previously placed rectangles). To overcome this drawback, Chazelle (1983) developed the Bottom-Left-Fill heuristic. Liu & Teng (1999) further proposed the Improved Bottom-Left, which always gives priority to the down movement. Zhang et al. (2006) developed a fast recursive heuristic to find the minimum height for the 2D-SPP. Burke et al. (2004) proposed the Best-Fit (BF) heuristic for the two-dimensional rectangular cutting stock problem. Their method selects the best rectangle for the placement according to the spaces to be filled into the strip. A bidirectional best-fit heuristic was later proposed by Aşık & Özcan (2009). Leung et al. (2011) also utilized the concept of best-fit and proposed a scoring rule based heuristic for the 2D-SPP. Verstichel et al. (2013) proposed Three-way Best-Fit (T-w BF) heuristic, adding three new criteria in the placement procedure. Wei et al. (2009) presented a Least-Waste-First strategy based heuristic for the 2D-SPP that determines the best pair of position and rectangle for the placement. Wei et al. (2011) proposed a skyline based heuristic, suggesting a set of priority rules. Wei et al. (2017) further improved the skyline heuristic by introducing the fitness number while selecting the best rectangle. He et al. (2013) earlier proposed a deterministic heuristic algorithm for the 2D-SPP in which orthogonal rotation of pieces was allowed.

Several other researchers hybridized meta-heuristics with heuristic placement rules to further enhance the solution quality. For example, Jakobs (1996) implemented Genetic Algorithm with the BL method. Liu & Teng (1999) also used Genetic Algorithm with the BL heuristic. Hopper & Turton (2001) combined Genetic Algorithm, Simulated Annealing, and Tabu Search with the BL/BLF and evaluated their performance for the 2D-SPP. Burke et al. (2009) obtained improved solutions by the hybridization of the BF heuristic together with Simulated Annealing and the BLF. Dereli & Daş (2007) proposed a new recursive placement procedure, which was combined with Simulated Annealing for the 2D-SPP. Leung et al. (2011) presented a two-stage Intelligent Search Algorithm with their improved BF heuristic, which is based on a scoring rule. Yang et al. (2013) proposed a Simple Randomized Algorithm, improving the scoring rule and introducing the least waste priority strategy in the placement heuristic. Wei et al. (2016) presented an efficient Intelligent search Algorithm involving three stages, each using a heuristic approach to construct solutions based on an improved scoring rule and the least-waste-first strategy. Wei et al. (2011) proposed the Iterative Doubling Binary Search algorithm with their skyline based heuristic. İsmail Babaoğlu (2017) presented an implementation of the fruit fly optimization algorithm to solve the 2D-SPP. The aim of his study was to find the best sequence of the rectangles with FOA, and then to place the rectangles by the BLF approach. Another popular meta-heuristic for the 2D-SPP is the effective corner increment-based algorithm of (Chen & Chen, 2018).

More recently, Rakotonirainy & van Vuuren (2020) proposed two improved meta-heuristics for the 2D-SPP. The first algorithm is a hybrid of Simulated Annealing and a heuristic construction algorithm, while the second algorithm involves application of Simulated Annealing directly in the space of completely defined packing layouts, without an encoding of solutions. Alvarez-Valdes et al. (2008) presented a greedy randomized adaptive search procedure (GRASP) for the 2D-SPP, investigating several strategies for the constructive and improvement phases. Burke et al. (2011) presented a much simpler but equally competitive iterative packing methodology based on squeaky wheel optimization for the 2D-SPP. Shalaby & Kashkoush (2013) proposed the particle swarm optimization algorithm for a 2D irregular strip packing problem. Júnior et al. (2017) proposed a parallel Biased Random-Key Genetic Algorithm with multiple populations for the irregular strip packing problem by applying a collision-free region concept as the positioning method. Bortfeldt (2006) suggested a Genetic algorithm for the 2D-SPP that works without any encoding of solutions. Rather, fully defined layouts are manipulated as such by means of specific genetic operators. Yuan & Liu (2009) approached the 2D-SPP using max-min ant system and an improved placement strategy. Gómez-Villouta et al. (2010) presented a reinforced Tabu Search algorithm for the 2D-SPP. Recently, Grandcolas & Pain-Barre (2022) presented a hybrid meta-heuristic approach called Progress and Verify Strategy for the 2D-SPP. It relies on two procedures: a local search algorithm that delivers satisfying placements of items on the horizontal axis, and an exact procedure that searches for the positions of items on the vertical axis.

Most of the research works mentioned above solve the classical 2D-SPP. However, there exists some other studies, which tackle the 2D-SPP considering additional constraints. Wei et al. (2019) solved the two-dimensional strip packing problem with unloading constraints (2DSPU) and proposed an open space based first-fit heuristic to generate a packing pattern for a given sequence of items. They also used a randomized local search to improve the solution by trying different sequences. Silveira et al. (2014) presented approximation algorithms for the 2DSPU. Earlier, Silveira et al. (2013) proposed two approximation algorithms and a GRASP for the 2DSPU and did extensive computational experiments to verify the performance of these algorithms. Queiroz & Miyazawa (2013) solved the 2D-SPP considering multi-drop and load bearing constraints. Moreover, Queiroz & Miyazawa (2014) investigated the 2SPOS proposing a branch-and-cut approach.

Some other works that deal with the stability constraint were proposed for the container loading problem (Ramos et al., 2016; Christensen & Rousøe, 2009; Bracht et al., 2016; Tarantilis et al., 2009; Martínez-Franco et al., 2020). To the best of our knowledge, the work of Queiroz & Miyazawa (2014) is only that considers the order and stability constraints together in the context of 2D-SPP. Their algorithm may require large computational time and could solve only very small-sized instances. Hence, in this work, an attempt has been made to develop a fast algorithm that could produce good solutions in reasonable computation time even on large instances of the 2SPOS. Since the BRKGA has already been successfully applied to solve the 2D and 3D bin packing problems, in this paper it is hybridized with the BLF and OS heuristics for the considered problem. The SA and PSO are also considered with these heuristics in order to evaluate the performance of BRKGA on instances of different sizes.

3 PROBLEM BACKGROUND

In the 2SPOS, there is a strip S of width W and unbounded height, and a set of N items, each item i with width w_i, height h_i, order number o_i, and mass m_i. The o_i represents the unload order of item i (items with a smaller o_i are unloaded first). It is assumed that the items are unloaded from the top of the strip using only vertical movements. The bottom-left corner of the strip has coordinates $(0,0)$ and its width (height) side is parallel to the $x\,(y)$-axis. The bottom-left corner of an item i packed in S is represented by the coordinates (x_i, y_i). The problem's objective is to pack all items into S in order to minimize the packing height while satisfying the following constraints:

- All items must be inside the strip.
- All items must be packed orthogonally.
- Items cannot be rotated.

- Items cannot overlap each other.
- The resulting packing needs to be vertically stable.
- Items must respect the order constraint: for any two items i and j, where $o_i \leq o_j$, $(x_j \geq x_i + w_i) \vee (x_i \geq x_j + w_j) \vee (y_i \geq y_j + h_j)$ must be true.

3.1 Stability in 2SPOS

This section details how to check whether an item in the packing is vertically stable or unstable. To do so, it is assumed that only the force of gravity with the intensity of $g = 9.81\ m/s^2$ is acting on items and no other external forces are present on the packing. Let A_i^+ (A_i^-) be the set of items immediately above (below) and in direct contact with item i. Now, consider the Figure 1, where:

1. F_i^w is the weight of item i, i.e, $F_i^w = m_i g$. It is acting at the geometric center of item i.
2. F_{ij} is the force acting on the top of i that is transferred from j. For $j \in A_i^+$, there is $F_{ij} = F_j^v$ if $A_j^- = \{i\}$, otherwise, $F_{ij} = \eta_{ij} F_j^v$. Force F_{ij} acts on i according to two cases: at the same vertical direction of the j center of mass if A_j^- only has i; or, at the middle of the contact surface between j and i if A_j^- has two or more items.
3. F_i^v is the resultant force in the vertical direction, which is obtained by summing up F_i^w and F_{ij}, for all $j \in A_i^+$.

To exemplify the calculation of η, let's consider items q_1, q_2 and q_3. Forces $F_{q_1 i}$, $F_{q_2 i}$ and $F_{q_3 i}$ that are on the top of them have been transferred from item i, as illustrated in Figure 1. The b and c are mid points of the horizontal gaps between q_1 & q_2 and q_2 & q_3, respectively. In the absence of gaps, these points will denote the points at which items touch each other. The a and d are end points of the bottom face of item i. So, for the leftmost item q_1, the value of $\eta_{q_1 i}$ is the area of item i between the left end point a and the mid point b divided by the complete area of item i. The calculation of $\eta_{q_1 i}$ is in the same way if the left edge of q_1 is at the same x-coordinate or right side of the left edge of item i. For the intermediate item q_2, $\eta_{q_2 i}$ is equal to the area of item

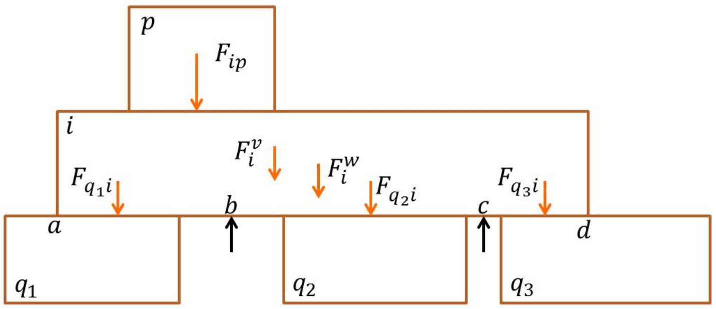

Figure 1 – Example of vertical forces acting on a packing.

i between the mid point b and the mid point c divided by the complete area of item i. Lastly, for the rightmost item q_3, η_{q_3i} equals the area of item i between the mid point c and the right end point d divided by the complete area of item i. The η_{q_3i} is calculated in the same way if the right edge of q_3 is at the same x-coordinate or left side of the right edge of item i. Once the value of η is known, the transferred forces can be calculated. For example, F_{q_1i} is equal to $\eta_{q_1i}F_i^v$. It is to be noted that this is a heuristic way to transfer the forces and it does not always guarantee that the transferred forces are correct.

Queiroz & Miyazawa (2014) determined the transferred forces in a different way when $|A_i^-| \geq 2$. They assumed that i is the structural element called beam and the adjacent items in A_i^- are columns that gives support to i. Due to the interaction between the beam and the columns, reaction forces appear. These reactions forces actually represent the forces that are transferred from i to it's supporting items. In the case of two supports (two items in A_i^-), static equilibrium equations of rigid bodies were used to calculate reaction forces. In the case of three or more supports, the three-moment equation method was applied. In the case of a single support, F_i^v is totally transferred as same as in this paper. Their method of vertical stability checking can be computationally expensive as the calculation of reaction forces is made on $O\left(n^3\right)$. In this paper, therefore, a heuristic approach is adopted to estimate the transferred forces quickly.

Furthermore, F_i^v acts at the center of mass. Its position along the the x-axis (Δ_i^v) is calculated by the formula shown in the equation (1).

$$\Delta_i^v = \frac{F_i^w \delta_i + \sum_{j \in A_i^+} F_{ij} \delta_j}{F_i^w + \sum_{j \in A_i^+} F_{ij}} \tag{1}$$

in which δ_i and δ_j are the coordinates on the x-axis where the respective forces F_i^w and F_{ij} are acting on i. Note that, when only the weight force is acting on an item, its center of mass coincides with the geometric center. In the presence of forces from other items, the center of mass shifts to Δ_i^v. Moreover, an item i is considered stable if the projection of Δ_i^v lies above any item $j \in A_i^-$ or between two items in A_i^- or on the floor of the strip. The packing is said to be vertically stable if all the items packed in it are stable.

The items in the packing are analyzed from top to bottom and left to right for those packed at the same y-coordinate. While doing so, the resultant vertical force (F_i^v) and its position on each item are saved. To obtain the transferred forces on any item i, the items j immediately above i and in direct contact with it are determined first. For each j, it is checked which items are present immediately below and in direct contact with j. If there is only i, then the resultant vertical force of item j is fully transferred to i. Otherwise, item i receives a fraction of the resultant vertical force of item j, which is determined by the η calculation method described above. Once transferred forces are known, Δ_i^v is calculated and checked whether it lies in one of the stable regions. A pseudocode describing all these steps for checking the vertical stability has also been provided in Algorithm 1 (Appendix II).

4 SOLUTION METHODOLOGY

The integration of meta-heuristics with heuristic placement rules has worked well on complex packing problems. In this paper, the biased random-key genetic algorithm, simulated annealing and particle Swarm Optimization meta-heuristics have been amalgamated with the bottom-left-fill and open space heuristic to solve the 2SPOS. These algorithms follow the standard framework proposed in the literature. In the next sections, the working process of these techniques and how they are applied to handle the 2SPOS have been discussed.

4.1 Biased random-key genetic algorithm

The Genetic Algorithm (GA), introduced by Holland (1975), is widely accepted as a powerful optimization technique for solving real-world hard combinatorial optimization problems quickly, reliably and accurately. The main building blocks of the GA are "chromosomes", which represent candidate solutions to the optimization problem. The traditional GA often produces infeasible chromosomes upon crossover in the case of sequencing problems and hence, extra computational effort is needed to repair them. In order to circumvent this, Bean (1994) proposed the random-key genetic algorithm (RKGA). In RKGA, chromosomes are represented as vectors of so called "keys" (randomly generated real numbers in the interval $[0,1]$). And, *parameterized uniform crossover* (Spears & DeJong, 1991) is used that always generates legal off-springs. Aiming to further improve the performance of the RKGA, Gonçalves & Resende (2011) developed the *biased random-key genetic algorithm* (BRKGA) by making a slight modification in the mating selection of the RKGA.

The framework of the BRKGA is illustrated in Figure 2. It begins with the initialization of a population comprising of p random-key vectors. Each component of a key vector is a uniformly randomly generated real number in the interval $[0,1]$. The fitness of each key vector (individual or chromosome) is evaluated, and the current population is partitioned into two groups of individuals: a group with p_e elite individuals (solutions with the best fitness values) and the remaining $p - p_e$ non-elite individuals. Then, a new population for the next generation is created. All elite individuals are copied into it without any modifications. The mutation process is ensured by adding p_m mutants, which are also key vectors generated in the same way that an individual of the initial population is generated. To keep the population size constant, the rest of the $p - (p_m + p_e)$ slots in the new population are filled with off-springs created by the crossover operation. In the BRKGA, the crossover is performed between randomly chosen an elite solution and a non-elite solution. While in the RKGA it is done between any two randomly chosen solutions from the entire population. When the new population is complete, fitness values are calculated for all the newly created random-key vectors and the population is partitioned into elite and non-elite individuals to continue the evolution process. A pseudocode of BRKGA is provided in Algorithm 2 (Appendix II). The implementation details of the BRKGA for the 2SPOS are discussed in sections 4.1.1 and 4.1.2.

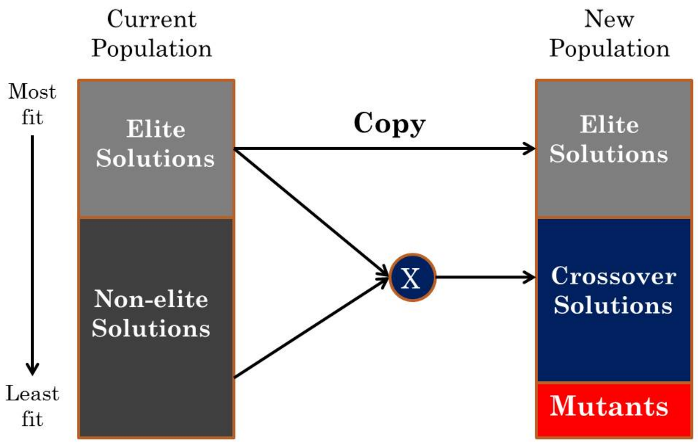

Figure 2 – An iteration of BRKGA.

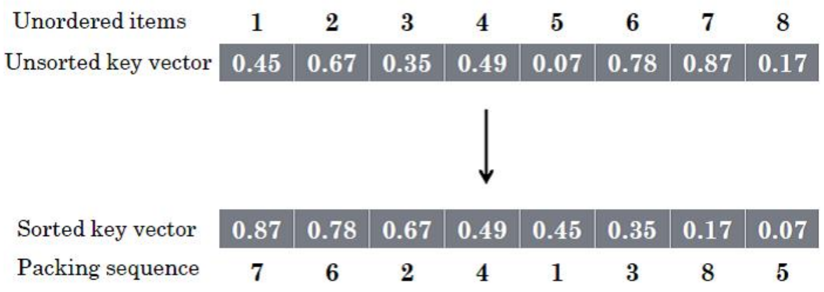

Figure 3 – Generation of the item packing sequence.

4.1.1 Population and decoding

The BRKGA is a population based global search meta-heuristic technique. Its performance is greatly enhanced when it can explore different parts of the search space with a diverse set of solutions. Thus, in order to promote diversity and favor an unbiased investigation of the whole solution space, the initial population is generated uniformly randomly. Each chromosome is encoded as a vector of random keys and having length equal to N (total number of rectangular items in an instance of the problem). The random keys (real numbers in $[0,1]$) are used to obtain a packing sequence of items. To explain this, consider Figure 3 showing 8 items and a key vector. Each item is associated with a key (item 1 with the first key, item 2 with the second key and so on). The key values are sorted in the descending order, and the positions of items in the sorted key vector define the sequence in which they will be inserted into the strip. For example, in Figure 3, the obtained packing sequence is $(7,6,2,4,1,3,8,5)$.

While applying the BRKGA on the 2SPOS instances, the packing sequence obtained by sorting keys is further modified by the Algorithm 3 (Appendix II). And, then the placement heuristic

(BLF/OS) is applied that packs items one by one following the new modified sequence. In the case of 2D-SPP/2D-OPP instances, Algorithm 3 is not used, and the packing sequence obtained after sorting keys is directly used by the placement heuristic.

Algorithm 3 was proposed after preliminary experiments on the 2SPOS instances. It was observed that the BRKGA+BLF/OS with packing sequences obtained by sorting keys was not able to find feasible solutions due to violation of the order constraint. Algorithm 3 rearranges items in a descending order of their order numbers. Due to different order of items having the same order number in the packing sequences generated by sorting keys, Algorithm 3 produces different strings. In the packing sequence created by Algorithm 3, higher order number items are packed before lower order number items. By this way of packing the BRKGA+BLF/OS could find packing patterns that do not violate the order constraint.

4.1.2 Crossover

The crossover is a process of mixing genes of parent solutions to produce offsprings. One of the motivations behind the development of the BRKGA (or RKGA) is to avoid infeasible chromosomes created by the traditional crossover operators (e.g., one point or two-point operators). So, following this notion, *parameterized uniform crossover* has been utilized in which there is no such feasibility issue. In order to describe this, let e, $\bar{e}$ and c be an elite parent, a non-elite parent, and an offspring, respectively. Their length is equal to N, and k represents a component/gene ($k = 1,2,3,\ldots,N$) of the key vectors. The probability of crossover is p_c. For each component k of the offspring, a random number between 0 and 1 is generated. If it is less than or equal to p_c then the offspring takes the corresponding gene value from the elite parent (i.e., $c(k) = e(k)$), otherwise, $c(k) = \bar{e}(k)$.

4.2 Simulated annealing

Simulated Annealing (SA) is a local search based meta-heuristic proposed by Metropolis et al. (1953). It imitates the physical process of annealing in which a solid is heated up to a predetermined temperature and then allowed to cool slowly to gain uniform hardness. The general framework of SA for the minimization problem has been shown in Algorithm 4 (Appendix II). It consists of two nested loops. The inner loop simulates the achievement of thermal equilibrium at a given temperature, so it is referred as the thermal equilibrium loop. The outer loop performs the cooling process, in which the temperature is decreased from its initial value towards zero until certain convergence criterion is achieved, so this loop is referred as the cooling loop or annealing loop. SA starts with the generation of an initial feasible solution (S_o). In each step of the inner loop, a neighbour solution (S') is generated. If the neighbour solution (S') is better than the current solution (S), then S' is accepted. Otherwise, the neighbour solution (S') is accepted as the current solution (S) with a probability of acceptance that depends on the energy (fitness value) as well as on the current temperature. Afterwards, the best found solution (S^*) is updated. After exiting from the inner loop, the current temperature is reduced by a geometric cooling schedule and then the inner loop is restarted.

In this paper, the initial solution is randomly generated as a permutation of item numbers. To search for neighbour candidate solutions, SA uses the *Swap* method in which two items are randomly selected and their positions are exchanged. On the 2SPOS instances, however, solutions are further modified by Algorithm 3 (Appendix II) before calling the BLF/OS heuristic for the same reason of getting infeasible solutions due to violation of the order constraint.

4.3 Particle Swarm Optimization

The Particle Swarm Optimization (PSO) algorithm mimics cognitive and collaborative behaviours of swarm of birds during the search of food. In PSO, each individual in the swarm is referred as a particle. To reach at the desired destination, particles navigate the hyperspace (problem's solution space). In the process of exploration, they memorize their own best positions and the best position discovered by the swarm. In order to move from one position to the other, particles update their velocities and positions according to equations (2) and (3).

$$V_l(t) = \Omega \times V_l(t-1) + C_1 \times rand(.) \times \left(X_l^{best} - X_l(t-1)\right) + C_2 \times rand(.) \times \left(X_s^{best} - X_l(t-1)\right) \quad (2)$$

$$X_l(t) = V_l(t) + X_l(t-1) \quad (3)$$

where: Ω is the inertia weight constant; $V_l(t)$ is the velocity of particle l at t^{th} iteration; C_1 and C_2, respectively, denote cognitive and collaborative ability of particles called accelerations coefficients; $rand(.)$ is a randomly generated value between 0 and 1; X_l represents the current position of particle l; X_l^{best} and X_s^{best} denote the best position of an individual particle l and the global best position found by swarm, respectively.

As shown in equations (2) and (3), each particle adjusts its position based on its own best-known solution (personal best) and the best solution discovered by the entire group (global best). This collaborative movement enables particles to converge toward optimal solutions over iterations. A detailed pseudocode of PSO has been shown in Algorithm 5 (Appendix II). Each iteration consists of three phases: updating velocities/positions, updating personal best positions, and updating the global best position. The position of a particle is encoded as a vector of length N (total number of rectangular items in an instance of the problem). Each component of a position vector is a real number in $[lb, ub]$. During the search process, if any component of a position vector becomes less than lb, it is reset at lb and if becomes greater than ub, it is reset equal to ub. The velocity of a particle is also a vector of length N. Each component of the initial velocity vectors is a uniformly randomly generated real number in $[0,1]$. To obtain a packing sequence, a position vector is sorted in the same way as a key vector in BRKGA. As in SA and BRKGA, Algorithm 3 is used on the 2SPOS instances when applying the PSO algorithm.

4.4 Bottom-left-fill and Open space

The bottom-left-fill (BLF) heuristic packs items one by one in accordance with a given sequence. It strictly adheres to the condition that no items packed can be moved further to the bottom or to the left. In BLF, a list of insertion positions is maintained. Among the feasible insertion positions from the list, the position with the minimum y-coordinate is selected, breaking ties with the minimum x-coordinate. An insertion position is considered feasible when upon insertion the (partial/complete) packing obtained satisfies all the constraints, including the stability one.

In the beginning, the position list (*pList*) contains only the origin, which is the coordinate of the bottom left corner of the strip. Whenever an item is inserted, its loading position is erased from the *pList* and at most four new insertion points are added into the *pList*. This is illustrated in Figure 4. An item i having width w and height h is placed at the position (x,y). The four new insertion positions created are as follows: (1) bottom right corner of item i $(x+w,y)$, (2) top left corner of item i $(x,y+h)$, (3) minimum non-occupied y-axis point of the right edge projection of item i $(x+w,y')$, and (4) leftmost non-occupied point of the top edge projection of item i $(x',y+h)$. To update the *pList* after insertion of item i, the position (x,y) is deleted and the newly created four positions are added into the *pList*. Moreover, duplicate position entries are removed from *pList*.

The open space (OS) heuristic also packs the rectangles one by one following an ordered sequence of rectangles. This method uses the open space to represent the candidate position that the rectangle can be placed at. A list of candidate positions is maintained during the packing process, and the first position in the list at which the (partial/complete) packing obtained after the insertion satisfies all the constraints is selected. It is suggested to see the research article of Wei et al. (2019) for a detailed description of the open space heuristic. In this paper, exactly the same version proposed in Wei et al. (2019) has been implemented.

It is to be noted that there is no change in the original BLF and OS heuristics under the order and stability constraints. The order and stability constraints of the (partial/complete) packing are checked while detecting the feasibility of an insertion position. Moreover, while applying the BLF/OS, if for any item a feasible insertion position is not found, then the solution is considered infeasible and an infinitely large packing height is assigned to it.

5 RESULTS AND DISCUSSION

This section presents a detailed description of exhaustive experiments accomplished in this research. The algorithms described in the previous section have been tested on a wide range of benchmark instances. The first test set includes 26 small-sized 2SPOS instances created by Queiroz & Miyazawa (2014). The strip width (W) and the number of items (N) in these instances vary in the range of 10-60 and 5-20, respectively. The order of each item was randomly chosen in the set $\{1,2,3,4\}$, while the mass of an item is given by its area. The second dataset consists of 15 newly generated 2SPOS instances, called MQM. These instances are partitioned into three groups: (1) MQM1 - MQM5 ($W = 80$ and $N = 125$), (2) MQM6 - MQM10 ($W = 60$

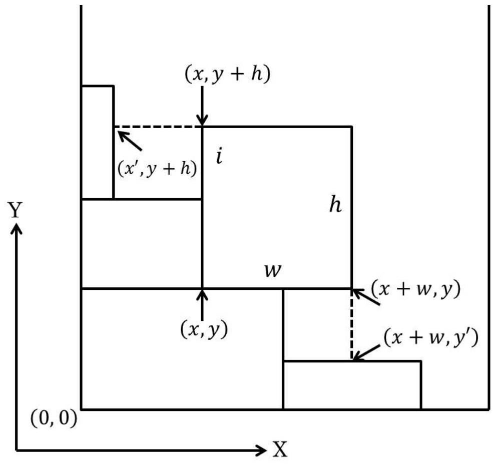

Figure 4 – Insertion of an item in BLF.

and $N = 100$) and (3) MQM11 - MQM15 ($W = 40$ and $N = 75$). The dimensions (w_i, h_i) of each item i were generated randomly in the closed interval $[0.10W, 0.40W]$. The order number an item receives is uniformly chosen in $[1, C_t]$, where $C_t = \lceil \frac{tN}{10} \rceil$ and $t = 2, 4, 6, 8, 10$, respectively, for $1^{st}, 2^{nd}, \ldots, 5^{th}$ instance in each group. The mass of an item is equivalent to its area. Three classical 2D-SPP datasets; namely, C, beng and NT(N) were also used. The C dataset (Hopper & Turton, 2001) consists of 21 instances with W and N ranging between 20-160 and 16-197, respectively. The minimum and maximum number of items in the beng instances (Bengtsson, 1982) are 20 and 200, while W is either 25 or 40. In the NT(N) dataset (Hopper, 2000), W is 200 for all 35 instances and N lies between 17-197. The last dataset, called CJCM, contains 42 2D-OPP instances generated by Clautiaux et al. (2007). This set of 42 instances is divided into 15 feasible (F) and 27 infeasible (NF) instances. For each instance, the container is a square 20×20 and the number of items $10 \leq N \leq 23$. The algorithms were coded in C++ on Visual Studio Code 1.74.2 and run on Linux server with Intel(R) Xeon(R) CPU E5-2603 v3 @ 1.60 GHz and 32 GB of RAM.

In order to achieve high-quality solution for each problem instance, detailed experiments have been performed for several possible governing control parameters of BRKGA, SA, and PSO. While doing experiments for the BRKGA, p_e and p_m have been varied in the range of 10-25% and 15-30%, respectively, in a step of 5. Also, three values of p_c (0.70, 0.75, 0.80) were tested. It was observed that the combination of $p_e : 0.10 \times p$ and $p_m : 0.15 \times p$ with p_c value of 0.70 produce better results. Furthermore, the population comprising 30-100 key vectors was checked and it was found that BRKGA was best with the population size (p) of 30 for 2SPOS/2D-SPP datasets and 100 for the 2D-OPP dataset. A higher population size than these required more computing time without a respective positive gain in the final solution. So, the (p, p_e, p_m, p_c) value of $(30, 3, 5, 0.70)$ and $(100, 10, 15, 0.70)$ were held constant for 2SPOS/2D-SPP instances and 2D-OPP instances, respectively. The BRKGA was allowed to run till 2300 cpu seconds on

2SPOS/2D-SPP instances and 900 cpu seconds on 2D-OPP instances. Likewise, through some initial experiments, the T_o, T_f and α values in SA were set at 1.0, 0.01 and 0.9, respectively. The control parameter *Len* was inspected by varying in the range of 30-100. The *Len* value of 100 was used finally as it generated much better solutions. As the pseudo code of SA shows, it was allowed to explore the search space until the current temperature falls below T_f. Similarly, parameters of PSO were set after executing some experiments. The population size (number of particles) was set at 30. Three values of Ω (0.3, 0.6, 0.9) were checked and PSO was found to be better with Ω value of 0.9. The acceleration coefficients C_1 and C_2 were set at 1.5, and $[lb, ub]$ was set as $[-5, 5]$. The PSO was also allowed to explore the search space till 2300 cpu seconds. The presented results by BRKGA/SA/PSO on each instance in Tables 1, 2, 3, A1, A2, A3 and A4 belong to the best of 5 runs.

The results obtained by the two versions of the BRKGA (BRKGA + BLF and BRKGA + OS) on the first 2SPOS test set have been provided in Table 1. The entry "-" represents that the time limit was reached (that is, greater than or equal to 2300 cpu seconds), and no feasible solution was found. The optimal solutions obtained by the branch & cut (BCut) algorithm of Queiroz & Miyazawa (2014) have also been shown. The columns labeled H and cpu time(s) give packing heights and computational times, respectively, obtained by the algorithms. As it can be seen, BRKGA + BLF and BRKGA + OS performed very well, retrieving optimal solutions on 11 and 9 instances, respectively. Moreover, BRKGA has found to be much faster than BCut. For example, on the instance SS03-60-8, BCut obtained the packing height of 16 while consuming a large computational time of 1004.00 seconds, whereas BRKGA + BLF and BRKGA + OS gave the same packing height in just 0.0150 and 0.0087 cpu seconds, respectively. Furthermore, the deviation/gap from the optimum packing height is extremely small on the instances BRKGA could not retrieve the optimal solution. For example, on the instances SS01-20-15 and SS03-20-15, packing heights produced by BRKGA + BLF/OS deviate from the optimum values by 2 and 1 unit(s), respectively. In fact, on 10 instances BRKGA + BLF produced the deviation of less than 5 units from the optimum packing height. While BRKGA + OS gave the deviation of less than 5 units on 12 instances. On the ngcut09 instance, the algorithm could not produce a feasible solution due to violation of the stability constraint. The placement heuristics (BLF/OS) could not find a feasible insertion position (in terms of stability) for an item (say i). Upon seeing such an item i in the packing, it was observed that infeasibility occurred because the projection of Δ_i^y was lying in the hole below (empty space surrounded by previously placed rectangles), which was not between two items in A_i^-. Overall, BRKGA + BLF seems to be better as it obtained lesser packing height on 3 instances in negligible computational times.

To better assess the effectiveness of both versions of BRKGA, a comparative analysis is being made between them and against the SA and PSO algorithms on the 15 large MQM instances. Tables 2 and 3 present results on the MQM dataset by BRKGA/SA and PSO, respectively. The column labeled $D(\%)$ gives a measure of the compactness of the packing. It is calculated by the

Table 1 – Results on 2SPOS dataset by Queiroz & Miyazawa (2014)

			BCut		BRKGA + BLF		BRKGA + OS	
Instance	N	W	H	cpu time(s)	H	cpu time(s)	H	cpu time(s)
ngcut01	5	10	11	0.00	**11**	0.0023	**11**	0.0021
ngcut02	7	10	15	1.00	**15**	0.0060	**15**	0.0060
ngcut03	10	10	16	19.00	**16**	0.6803	19	0.0149
ngcut04	5	15	8	0.00	**8**	0.0073	**8**	0.0047
ngcut05	7	15	15	2.00	**15**	0.0058	**15**	0.0078
ngcut06	10	15	15	364.00	**15**	0.0553	16	0.0121
ngcut07	5	20	11	0.00	**11**	0.0019	**11**	0.0038
ngcut08	7	20	19	9.00	**19**	0.0076	**19**	0.0100
ngcut09	10	20	31	358.00	–	–	–	–
ngcut10	5	30	46	2.00	**46**	0.0014	**46**	0.0036
ngcut11	7	30	33	7.00	**33**	0.0186	**33**	0.0046
SS01-20-15	15	20	6	13.00	8	0.0330	8	0.0326
SS02-20-15	15	20	6	246.00	8	0.6792	8	0.1049
SS03-20-15	15	20	7	7.00	8	0.1208	8	0.0321
SS01-20-20	20	20	9	117.00	13	0.2256	13	0.1244
SS02-20-20	20	20	8	68.00	12	1.3405	12	0.1231
SS03-20-20	20	20	8	42.00	12	0.1935	12	0.0706
SS01-40-8	8	40	12	42.00	15	0.0216	15	0.0213
SS02-40-8	8	40	11	144.00	16	2.0093	16	2.0055
SS03-40-8	8	40	10	12.00	14	0.0128	14	0.0062
SS01-40-15	15	40	15	3600.00	22	1.2599	22	0.1412
SS02-40-15	15	40	15	3600.00	21	0.3427	21	0.1051
SS03-40-15	15	40	15	3600.00	22	1.6642	23	0.3295
SS01-60-8	8	60	14	368.00	15	0.0113	15	0.0074
SS02-60-8	8	60	15	3600.00	16	0.0144	16	0.0078
SS03-60-8	8	60	16	1004.00	**16**	0.0150	**16**	0.0087

formula shown below in equation (4). So, the higher is the D value, more compact/dense is the packing. A dense packing is preferred as it will be more stable.

$$D = \frac{\sum_{i=1}^{N} w_i \times h_i}{W \times H} \times 100(\%) \tag{4}$$

where, W is the strip width and H is the packing height. The compactness of the packing obtained by the algorithms is greater than 75% on all instances. However, the BRKGA + BLF algorithm is the best in generating more compact packings. It obtained better D values than BRKGA + OS, SA + OS, SA + BLF, PSO + OS, and PSO + BLF on 11, 12, 11, 12, and 10 instances, respectively. On two instances (MQM6 and MQM11), the BRKGA + OS could obtain

Table 2 – Results on 2SPOS dataset MQM.

			BRKGA + BLF			BRKGA + OS			SA + OS			SA + BLF		
Instance	*N*	*W*	H	cpu time(s)	D(%)	H	cpu time(s)	D(%)	H	cpu time(s)	D(%)	H	cpu time(s)	D(%)
MQM1	125	80	733	1242.34	84.2070	729	1134.17	84.6691	733	852.984	84.2070	728	614.778	84.7854
MQM2	125	80	780	699.805	85.1538	793	711.381	83.7579	800	995.270	83.0250	774	1299.67	85.8140
MQM3	125	80	759	304.143	82.8969	761	117.584	82.6790	763	474.786	82.4623	760	316.522	82.7878
MQM4	125	80	748	1044.92	83.5695	762	443.628	82.0341	764	207.792	81.8194	769	1052.34	81.2874
MQM5	125	80	725	853.812	82.4431	732	374.847	81.6547	736	321.921	81.2109	735	185.524	81.3214
MQM6	100	60	442	459.213	83.8914	435	449.084	85.2414	443	368.207	83.7020	437	757.105	84.8513
MQM7	100	60	412	876.075	86.5170	416	571.907	85.6851	422	529.527	84.4668	419	669.116	85.0716
MQM8	100	60	444	877.138	87.7778	450	252.572	86.6074	460	317.313	84.7246	454	324.999	85.8443
MQM9	100	60	435	840.728	84.7203	461	414.899	79.9422	460	318.965	80.1159	445	526.485	82.8165
MQM10	100	60	467	872.743	83.9115	468	345.439	83.7322	472	281.141	83.0226	468	98.7686	83.7322
MQM11	75	40	231	138.207	85.5303	227	124.902	87.0374	230	115.291	85.9022	229	415.587	86.2773
MQM12	75	40	209	83.6365	87.2608	210	191.428	86.8452	217	214.585	84.0438	214	77.2622	85.2220
MQM13	75	40	226	174.278	85.7412	226	268.756	85.7412	232	239.637	83.5237	227	203.206	85.3634
MQM14	75	40	200	75.7704	84.5000	203	54.4951	83.2512	197	157.746	85.7868	202	24.4563	83.6634
MQM15	75	40	198	65.8791	85.3535	203	192.805	83.2512	206	51.3828	82.0388	202	12.7919	83.6634
Avg.	–	–	467.266	573.912	84.898	471.733	376.526	84.141	475.666	363.103	83.336	470.466	438.574	84.166

better *D* values than the others. On just one instance (MQM14), the compactness of the packing formed by the SA + OS is better. Whereas SA + BLF gave better *D* values than the others on two instances (MQM1 and MQM2). The BRKGA + BLF also found the most dense/compact packing with the *D* value of 87.7778% on MQM8. While the least dense packing was created by the BRKGA + OS on MQM9, obtaining the *D* value of 79.9422%. The (max, min) *D* values given by the BRKGA + BLF, BRKGA + OS, SA + OS, SA + BLF, PSO + OS, and PSO + BLF are (87.7778%, 82.4431%), (87.0374%, 79.9422%), (85.9022%, 80.1159%), (86.2773%, 81.2874%), (86.2773%, 80.5542%), and (86.9940%, 81.7124%), respectively. Furthermore, the BRKGA + BLF gave an average *D* value of 84.898%, whereas the BRKGA + OS, SA + OS, SA + BLF, PSO + OS, and PSO + BLF obtained average *D* values of 84.141%, 83.336%, 84.166%, 83.533%, and 84.449%, respectively. These values clearly show that BRKGA + BLF outperformed BRKGA + OS, SA + OS, SA + BLF, PSO + OS, and PSO + BLF with regard to the packing compactness. It happened so due the ability of the BLF to fill holes (empty spaces surrounded by previously packed rectangles) together with the greater search space exploration capability of BRKGA. The *D* value is inversely proportional to the packing height *H*. Hence, the BRKGA + BLF, which is better than the others in obtaining higher *D* values, is also the best in obtaining packings with minimum heights. It could not give better *H* value than BRKGA + OS, SA + OS, SA + BLF, PSO + OS, and PSO + BLF on just 3, 2, 4, 3, and 5 instances, respectively. The average *H* value given by BRKGA + BLF (467.266 units) is significantly better than that achieved by BRKGA + OS (471.733 units), SA + OS (475.666 units), SA + BLF (470.466 units), PSO + OS (475.933), and PSO + BLF (470.600). It was noticed that C_t corresponds to the (minimum) 20% of *N* on MQM1, MQM6, and MQM11. And, of these three instances, BRKGA + OS obtained packings with lesser heights than the others on two instances (MQM6 and MQM11) and SA + BLF gave better *H* value than the others on one instance (MQM1).

It is evident that BRKGA + BLF is the best among the applied algorithms for the 2SPOS. To further check its robustness, it was applied on the classical 2D-SPP and 2D-OPP datasets. Tables A1, A2, and A3 (Appendix I) show the results achieved on 2D-SPP datasets. The details of *N*, *W*, and *LB* in these tables can be found in Wei et al. (2011). The columns labeled *LB* and gap(%)

Table 3 – Results on 2SPOS dataset MQM.

			PSO + BLF			PSO + OS		
Instance	N	W	H	cpu time(s)	D(%)	H	cpu time(s)	D(%)
MQM1	125	80	746	583.789	82.7396	743	1973.68	83.0737
MQM2	125	80	798	1749.96	83.2331	809	442.012	82.1014
MQM3	125	80	752	2013.18	83.6686	759	356.229	82.8969
MQM4	125	80	765	1451.72	81.7124	772	429.193	80.9715
MQM5	125	80	726	1749.25	82.3295	742	816.730	80.5542
MQM6	100	60	445	1582.09	83.3258	441	1820.41	84.0816
MQM7	100	60	413	1006.71	86.3075	419	1046.38	85.0716
MQM8	100	60	448	1031.90	86.9940	456	844.202	85.4678
MQM9	100	60	437	1299.68	84.3326	456	1221.17	80.8187
MQM10	100	60	463	1049.00	84.6364	468	132.274	83.7322
MQM11	75	40	230	440.556	85.9022	229	1131.64	86.2773
MQM12	75	40	211	1595.21	86.4336	214	1152.48	85.2220
MQM13	75	40	225	886.854	86.1222	228	1154.00	84.9890
MQM14	75	40	200	676.215	84.5000	201	749.633	84.0796
MQM15	75	40	200	74.3602	84.5000	202	219.682	83.6634
Avg.	–	–	470.600	1146.032	84.449	475.933	899.314	83.533

in these tables show lower bounds and percentage deviations of obtained packing heights from lower bounds. The gap is defined as $100 \times (H - LB)/LB$. On the C dataset, BRKGA + BLF retrieved the lower bound on 4 instances. It produced the gap of less than 6% on 18 instances while giving the overall small average gap of 4.19%. The performance of BRKGA + BLF is also good and consistent on the *beng* dataset. On all instances, the solution quality deviates from the optimum/LB by less than 4%. It gave an average gap of just 2.93%, consuming an average computational time of 241.795 cpu seconds. Whereas on 19 NT(N) instances, the gap given by BRKGA + BLF is less than or equal to 6%. The average gap is even smaller, i.e, 5.67%. Table A4 (Appendix I) displays the results obtained on the 2D-OPP dataset CJCM. The ε is a parameter used in the generation of instances [see Clautiaux et al. (2007) for details]. The H_b stands for the height of the bin/strip. The notations F and NF under column *Feas.* stand for the feasible and infeasible instance, respectively. An instance is feasible if the packing height (H) obtained by the algorithm after placing all items is less than or equal to H_b; otherwise, it is infeasible. The performance of BRKGA+BLF is close to the exact algorithm of Clautiaux et al. (2007). Of 15 feasible instances, BRKGA+BLF found solutions for 12 instances. The BRKGA+BLF could provide solutions in less than 4 cpu seconds for 8 feasible instances. The infeasible instances remained infeasible. However, the packing height obtained by BRKGA+BLF deviates from the bin height by just 1 or 2 units on infeasible instances.

Based on the above comparative and analytical study of the obtained results on a variety of datasets, it can now be stated that the BRKGA achieves the targeted goals. In conjunction with the BLF, it not only produces vertically stable packings with minimum heights quickly on the

2SPOS datasets, but also turned out to be effective on the classical 2D-SPP and 2D-OPP datasets. Thus, BRKGA establishes itself as an attractive algorithm for cutting and packing problems.

6 CONCLUDING REMARKS AND FUTURE WORKS

In this research, the two dimensional strip packing problem with order and stability constraints (2SPOS) has been tacked by the integration of meta-heuristics and two placement heuristics. Motivated by the successful application of BRKGA in the field of combinatorial complex problems, it has been employed in conjunction with the BLF/OS placement heuristic to quickly obtain high-quality packings. A comparative and analytical study is performed between two versions of the algorithm: BRKGA + BLF and BRKGA + OS. They are compared with SA + BLF/OS and PSO + BLF/OS on the 2SPOS instances. The result demonstrates the superiority of BRKGA + BLF among the applied algorithms. Extending the research, the effectiveness of BRKGA + BLF is also tested on the classical 2D-SPP and 2D-OPP instances. The small deviations of packing heights from the lower bounds on the 2D-SPP instances and quick generation of solutions for most of the feasible 2D-OPP instances confirm its highly robust nature. Moreover, the distribution of forces in the packing by heuristic means also made BRKGA fast.

In future, this research can be extended by incorporating other practical constraints and objectives (e.g., load bearing, horizontal stability, and fragility requirements). To propose an enhanced version of BRKGA or a new solution methodology for the 2SPOS could be interesting. Finally, the stability heuristic proposed in this paper can be used in other problems such as the 2L-CVRP.

Acknowledgements

This work was financially supported by FAPESP - São Paulo Research Foundation (Proc. 2015/11937–9, 2022/06707-8, and 2022/05803–3), CNPq - National Council for Scientific and Technological Development (Proc. 313146/2022-5, 315555/2023-8, and 404315/2023-2) and FAPEG - the State of Goiás Research Foundation.

Statement of Publication Ethics

Plagiarism: The authors declare that the article submitted is of its own making and that any third-party material that have been used is legitimately referenced and authorized. The authors also declare that the article submitted for evaluation and its essential content is unpublished and is not in the process of being evaluated by another journal.

Data availability: All datasets used and/or analysed during the current study are available upon request.

Authors' contributions: All authors contributed to the study conception and design. The first draft of the manuscript was written by Santosh Kumar Mandal. All authors edited the manuscript and approved the final version.

Funding: This work was financially supported by FAPESP - São Paulo Research Foundation, CNPq - National Council for Scientific and Technological Development and FAPEG - the State of Goiás Research Foundation.

References

ALVAREZ-VALDES R, PARREÑO F & TAMARIT J. 2008. Reactive GRASP for the strip-packing problem. *Computers & Operations Research*, **35**(4): 1065–1083.

AŞIK OB & ÖZCAN E. 2009. Bidirectional best-fit heuristic for orthogonal rectangular strip packing. *Annals of Operations Research*, **172**: 405–427.

BAKER B, COFFMAN E & RIVEST R. 1980. Orthogonal packings in two dimensions. *SIAM Journal on Computing*, **9**(4): 846–855.

BEAN J. 1994. Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal of Computing*, **6**: 154–160.

BENGTSSON BE. 1982. Packing rectangular pieces-a heuristic approach. *The Computer Journal*, **25**(3): 353–357.

BORTFELDT A. 2006. A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces. *European Journal of Operational Research*, **172**(3): 814–837.

BRACHT EC, QUEIROZ TA, SCHOUERY RCS & MIYAZAWA FK. 2016. Dynamic cargo stability in loading and transportation of containers. In: *IEEE International Conference on Automation Science and Engineering (CASE)*. pp. 227–232. Fort Worth, TX, USA.

BURKE EK, HYDE MR & KENDALL G. 2011. A squeaky wheel optimisation methodology for two-dimensional strip packing. *Computers & Operations Research*, **38**(7): 1035–1044.

BURKE EK, KENDALL G & WHITWELL G. 2004. A new placement heuristic for the orthogonal stock-cutting problem. *Operations Research*, **52**(4): 655–671.

BURKE EK, KENDALL G & WHITWELL G. 2009. A simulated annealing enhancement of the best-fit heuristic for the orthogonal stock-cutting problem. *INFORMS Journal on Computing*, **21**(3): 505–516.

CHAZELLE. 1983. The bottom-left bin-packing heuristic: an efficient implementation. *IEEE Transactions on Computers*, **32**(8): 697–707.

CHEN Z & CHEN J. 2018. An effective corner increment-based algorithm for the two-dimensional strip packing problem. *IEEE Access*, **6**: 72906–72924.

CHRISTENSEN SG & ROUSØE DM. 2009. Container loading with multi-drop constraints. *International Transactions in Operational Research*, **16**(6): 727–743.

CLAUTIAUX F, CARLIERV J & MOUKRIM A. 2007. A new exact method for the two-dimensional orthogonal packing problem. *European Journal of Operational Research*, **183**: 1196–1211.

DERELI T & DAŞ GS. 2007. A Hybrid Simulated-Annealing Algorithm for Two-Dimensional Strip Packing Problem. In: *International Conference on Adaptive and Natural Computing Algorithms*. pp. 508–516.

GONÇALVES JF & RESENDE MG. 2013. A biased random key genetic algorithm for 2D and 3D bin packing problems. *International Journal of Production Economics*, **145**(2): 500–510.

GONÇALVES JF & RESENDE MGC. 2011. Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, **17**: 487–525.

GRANDCOLAS S & PAIN-BARRE C. 2022. A hybrid metaheuristic for the two-dimensional strip packing problem. *Annals of Operations Research*, **309**: 79–102.

GÓMEZ-VILLOUTA G, HAMIEZ JP & HAO JK. 2010. A Reinforced Tabu Search Approach for 2D Strip Packing. *International Journal of Applied Metaheuristic Computing*, **1**(3): 20–36.

HE K, JIN Y & HUANG W. 2013. Heuristics for two-dimensional strip packing problem with 90° rotations. *Expert Systems with Applications*, **40**(14): 5542–5550.

HOCHBAUM DS & MAASS W. 1985. Approximation schemes for covering and packing problems in image processing and VLSI. *Journal of the ACM*, **32**(1): 130–136.

HOLLAND JH. 1975. *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press.

HOPPER E. 2000. *Two-dimensional packing utilising evolutionary algorithms and other meta-heuristic methods*. Ph.D. thesis. University of Wales, Cardiff School of Engineering.

HOPPER E & TURTON BCH. 2001. An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem. *European Journal of Operational Research*, **128**(1): 34–57.

İSMAIL BABAOĞLU. 2017. Solving 2D strip packing problem using fruit fly optimization algorithm. *Procedia Computer Science*, **111**: 52–57.

JAKOBS S. 1996. On genetic algorithms for the packing of polygons. *European Journal of Operational Research*, **88**(1): 165–181.

JUNIOR BA, COSTA RL, PINHEIRO PR, ARAÚJO LJP & GRICHSHENKO A. 2020. A biased random-key genetic algorithm using dotted board model for solving two-dimensional irregular strip packing problems. In: *2020 IEEE Congress on Evolutionary Computation (CEC)*. pp. 1–8.

JÚNIOR AN, SILVA E, FRANCESCATTO M, ROSA CB & SILUK J. 2022. The rectangular two-dimensional strip packing problem real-life practical constraints: A bibliometric overview. *Computers & Operations Research*, **137**: 105521.

JÚNIOR BA, PINHEIRO PR & COELHO PV. 2017. A Parallel Biased Random-Key Genetic Algorithm with Multiple Populations Applied to Irregular Strip Packing Problems. *Mathematical Problems in Engineering*, Article ID 1670709, 11 pages.

LEUNG SCH, ZHANG D & SIM KM. 2011. A two-stage intelligent search algorithm for the two-dimensional strip packing problem. *European Journal of Operational Research*, **215**(1): 57–69.

LIU D & TENG H. 1999. An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles. *European Journal of Operational Research*, **112**(2): 413–420.

LODI A, MARTELLO S & VIGO D. 1999. Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. *INFORMS Journal on Computing*, **11**: 345–357.

MARTÍNEZ-FRANCO J, CÉSPEDES-SABOGAL E & ÁLVAREZ MARTÍNEZ D. 2020. Package-Cargo: A decision support tool for the container loading problem with stability. *SoftwareX*, **12**. 100601.

METROPOLIS N, ROSENALUTH AW & ROSENBLUTH MN. 1953. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, **21**(6): 1087–1092.

OMAR MK & RAMAKRISNAN K. 2013. Solving non-oriented two dimensional bin packing problem using evolutionary particle swarm optimisation. *International Journal of Production Research*, **51**(20): 6002–6016.

QUEIROZ TA & MIYAZAWA FK. 2013. Two-dimensional strip packing problem with load balancing, load bearing and multi-drop constraints. *International Journal of Production Economics*, **145**(2): 511–530.

QUEIROZ TA & MIYAZAWA FK. 2014. Order and static stability into the strip packing problem. *Annals of Operations Research*, **223**: 137–154.

RAKOTONIRAINY RG & VAN VUUREN JH. 2020. Improved metaheuristics for the two-dimensional strip packing problem. *Applied Soft Computing*, **92**: 106268.

RAMOS AG, OLIVEIRA JF, GONÇALVES JF & LOPES MP. 2016. A container loading algorithm with static mechanical equilibrium stability constraints. *Transportation Research Part B: Methodological*, **91**: 565–581.

SHALABY MA & KASHKOUSH M. 2013. A Particle Swarm Optimization Algorithm for a 2-D Irregular Strip Packing Problem. *American Journal of Operations Research*, **3**(2). article ID:29233.

SILVA LC, QUEIROZ TA & TOLEDO FMB. 2022. Integer formulations for the integrated vehicle routing problem with two-dimensional packing constraints. *Pesquisa Operacional*, **42**: e248686.

SILVEIRA JL, MIYAZAWA FK & XAVIER EC. 2013. Heuristics for the strip packing problem with unloading constraints. *Computers & Operations Research*, **40**(4): 991–1003.

SILVEIRA JL, XAVIER EC & MIYAZAWA FK. 2014. Two-dimensional strip packing with unloading constraints. *Discrete Applied Mathematics*, **164**(2): 512–521.

SOKE A & BINGUL Z. 2006. Hybrid genetic algorithm and simulated annealing for two-dimensional non-guillotine rectangular packing problems. *Engineering Applications of Artificial Intelligence*, **19**: 557–567.

SPEARS W & DEJONG KD. 1991. On the virtues of parameterized uniform crossover. In: *Fourth International Conference on Genetic Algorithms*. pp. 230–236.

TARANTILIS CD, ZACHARIADIS EE & KIRANOUDIS CT. 2009. A hybrid metaheuristic algorithm for the integrated vehicle routing and three-dimensional container-loading problem. *IEEE Transactions on Intelligent Transportation Systems*, **10**(2): 255–271.

VERSTICHEL J, CAUSMAECKER PD & BERGHE GV. 2013. An improved best-fit heuristic for the orthogonal strip packing problem. *International Transactions In Operational Research*, **20**(5): 711–730.

WEI L, HU Q, LEUNG SC & ZHANG N. 2017. An improved skyline based heuristic for the 2D strip packing problem and its efficient implementation. *Computers & Operations Research*, **80**: 113–127.

WEI L, OON WC, ZHU W & LIM A. 2011. A skyline heuristic for the 2D rectangular packing and strip packing problems. *European Journal of Operational Research*, **215**(2): 337–346.

WEI L, QIN H, CHEANG B & XU X. 2016. An efficient intelligent search algorithm for the two-dimensional rectangular strip packing problem. *International Transactions In Operational Research*, **23**: 65–92.

WEI L, WANG Y, CHENG H & HUANG J. 2019. An open space based heuristic for the 2D strip packing problem with unloading constraints. *Applied Mathematical Modelling*, **70**: 67–81.

WEI L, ZHANG D & CHEN Q. 2009. A least wasted first heuristic algorithm for the rectangular packing problem. *Computers & Operations Research*, **36**(5): 1608–1614.

WEI L, ZHANG Z, ZHANG D & LEUNG SCH. 2018. A simulated annealing algorithm for the capacitated vehicle routing problem with two-dimensional loading constraints. *European Journal of Operational Research*, **265**: 843–859.

YANG S, HAN S & YE W. 2013. A simple randomized algorithm for two-dimensional strip packing. *Computers & Operations Research*, **40**(1): 1–8.

YUAN C & LIU X. 2009. Solution to 2D Rectangular Strip Packing Problems Based on ACOs. In: *International Workshop on Intelligent Systems and Applications*. pp. 1–4. Wuhan, China.

ZHANG D, KANG Y & DENG A. 2006. A new heuristic recursive algorithm for the strip rectangular packing problem. *Computers & Operations Research*, **33**(8): 2209–2217.

How to cite

MANDAL SK, QUEIROZ TAQ & MIYAZAWA FK. 2024. A Biased Random-Key Genetic Algorithm for The 2D Strip Packing Problem with Order and Stability Constraints. *Pesquisa Operacional*, **44**: e284365. doi: 10.1590/0101-7438.2023.043.00284365.

APPENDIX I

Table A1 – Results on 2D-SPP dataset C.

Instance				BRKGA + BLF		
Name	*N*	*W*	*LB*	H	cpu time(s)	gap(%)
C11	16	20	20	20	2.2494	0.00
C12	17	20	20	21	0.0967	5.00
C13	16	20	20	20	0.2982	0.00
C21	25	40	15	15	37.0256	0.00
C22	25	40	15	16	0.1374	6.66
C23	25	40	15	15	41.614	0.00
C31	28	60	30	31	40.2822	3.33
C32	29	60	30	31	98.7852	3.33
C33	28	60	30	31	73.2538	3.33
C41	49	60	60	63	123.253	5.00
C42	49	60	60	63	131.865	5.00
C43	49	60	60	63	38.7805	5.00
C51	73	60	90	95	117.538	5.55
C52	73	60	90	95	173.526	5.55
C53	73	60	90	94	725.779	4.44
C61	97	80	120	127	337.362	5.83
C62	97	80	120	128	161.216	6.66
C63	97	80	120	127	380.323	5.83
C71	196	160	240	253	1197.09	5.41
C72	197	160	240	254	518.332	5.83
C73	196	160	240	255	1004.99	6.25
Avg.	-	-	82.14	86.52	247.7999	4.19

Table A2 – Results on 2D-SPP dataset beng.

Instance				BRKGA + BLF		
Name	*N*	*W*	*LB*	H	cpu time(s)	gap(%)
beng1	20	25	30	31	0.1229	3.33
beng2	40	25	57	59	91.5347	3.50
beng3	60	25	84	86	281.843	2.38
beng4	80	25	107	110	453.388	2.80
beng5	100	25	134	137	474.893	2.23
beng6	40	40	36	37	156.691	2.77
beng7	80	40	67	69	198.168	2.98
beng8	120	40	101	104	350.881	2.97
beng9	160	40	126	130	201.085	3.17
beng10	200	40	156	161	209.349	3.20
Avg.	-	-	89.8	92.4	241.795	2.93

Table A3 – Results on 2D-SPP dataset NT(N).

Instance				BRKGA + BLF		
Name	*N*	*W*	*LB*	H	cpu time(s)	gap(%)
n1a	17	200	200	200	68.0163	0.0
n1b	17	200	200	209	2.59804	4.5
n1c	17	200	200	211	0.10281	5.5
n1d	17	200	200	207	5.26973	3.5
n1e	17	200	200	200	48.5089	0.0
n2a	25	200	200	212	83.6716	6.0
n2b	25	200	200	209	50.3541	4.5
n2c	25	200	200	213	66.8893	6.5
n2d	25	200	200	214	16.5433	7.0
n2e	25	200	200	211	34.3348	5.5
n3a	29	200	200	214	18.0326	7.0
n3b	29	200	200	213	20.2273	6.5
n3c	29	200	200	211	25.5847	5.5
n3d	29	200	200	211	71.3916	5.5
n3e	29	200	200	210	25.9999	5.0
n4a	49	200	200	213	162.491	6.5
n4b	49	200	200	215	212.894	7.5
n4c	49	200	200	213	131.287	6.5
n4d	49	200	200	213	109.633	6.5
n4e	49	200	200	212	117.105	6.0
n5a	73	200	200	214	220.288	7.0
n5b	73	200	200	213	358.001	6.5
n5c	73	200	200	214	130.872	7.0
n5d	73	200	200	214	163.645	7.0
n5e	73	200	200	213	309.622	6.5
n6a	97	200	200	213	118.552	6.5
n6b	97	200	200	211	141.725	5.5
n6c	97	200	200	214	298.961	7.0
n6d	97	200	200	214	602.135	7.0
n6e	97	200	200	212	565.384	6.0
n7a	197	200	200	211	768.0	5.5
n7b	197	200	200	211	1057.68	5.5
n7c	197	200	200	209	1058.3	4.5
n7d	197	200	200	212	1196.25	6.0
n7e	197	200	200	211	1639.97	5.5
Avg.	-	-	200	211.34	282.87	5.67

Table A4 – Results on 2D-OPP dataset CJCM.

Instance					BRKGA + BLF		
ε	N	W	H_b	*Feas.*	*Feas.*	H	cpu time(s)
00	10	20	20	NF	NF	22	0.022256
00	15	20	20	NF	NF	21	1.526080
00	23	20	20	NF	NF	21	0.707631
00	23	20	20	NF	NF	21	22.77910
02	17	20	20	F	F	20	0.311232
02	20	20	20	F	NF	21	1.279720
02	22	20	20	F	F	20	3.776630
02	20	20	20	NF	NF	22	0.178222
03	10	20	20	NF	NF	21	0.149776
03	15	20	20	NF	NF	21	0.763581
03	16	20	20	NF	NF	21	854.6230
03	17	20	20	NF	NF	21	573.6290
03	18	20	20	F	NF	21	1.029370
04	15	20	20	F	NF	21	0.219845
04	17	20	20	F	F	20	66.47570
04	19	20	20	F	F	20	2.001320
04	20	20	20	F	F	20	2.556090
04	15	20	20	NF	NF	22	0.404958
04	17	20	20	NF	NF	21	195.3120
04	18	20	20	NF	NF	22	4.341920
05	15	20	20	F	F	20	169.0780
05	18	20	20	F	F	20	266.0550
05	20	20	20	F	F	20	0.870590
05	15	20	20	NF	NF	21	0.305335
05	17	20	20	NF	NF	21	0.741521
05	15	20	20	NF	NF	21	0.055521
07	15	20	20	F	F	20	1.157360
07	10	20	20	NF	NF	22	0.159582
07	15	20	20	NF	NF	21	0.063973
07	15	20	20	NF	NF	21	2.244130
08	15	20	20	F	F	20	96.95340
08	15	20	20	NF	NF	21	0.058781
10	10	20	20	NF	NF	21	0.024781
10	15	20	20	NF	NF	22	0.062777
10	15	20	20	NF	NF	21	204.1260
13	10	20	20	NF	NF	21	0.021681
13	15	20	20	NF	NF	22	0.058696
13	15	20	20	NF	NF	22	0.179233
15	10	20	20	NF	NF	21	0.045907
15	15	20	20	NF	NF	22	0.060220
20	15	20	20	F	F	17	0.608625
20	15	20	20	F	F	20	0.149543

APPENDIX II

Algorithm 1 Pseudocode for checking vertical stability

```
Input: A partial/complete packing;
    Check items one by one from top to bottom;
    If at the same $y$-coordinate, check items one by one from leftmost to rightmost;
 1: for each item $i$ do
    Determining transferred forces on item i from item j ($F_{ij}$) :
 2:     Determine set $A_i^+$;
 3:     for each $j \in A_i^+$ do
 4:         Determine set $A_j^-$;
 5:         if $A_j^-$ has only $i$ then
 6:             The vertical resultant force of $j$ is fully transferred to $i$, i.e, $F_{ij} = F_j^v$;
 7:             Determine $\delta_j$;        ▷ Coordinate on the $x$-axis where the transferred force is acting on $i$
 8:         else
 9:             Item $i$ receives a fraction of the vertical resultant force of $j$, i.e, $F_{ij} = \eta_{ij} F_j^v$;
10:             Determine $\delta_j$;
11:         end if
12:     end for

13:     Determine weight force of item $i$;
14:     Determine $\delta_i$;        ▷ Coordinate on the $x$-axis where the weight force is acting on $i$
15:     Determine the vertical resultant force on item $i$ ($F_i^v$);
16:     Determine $\Delta_i^v$ by equation (1);        ▷ Coordinate on the $x$-axis where the $F_i^v$ is acting on $i$;
17:     if $\Delta_i^v$ lies in a stable region then
18:         Item $i$ is stable;
19:     end if
20: end for
21: if All items in the packing are stable then
22:     Packing is vertically stable;
23: end if
```

Algorithm 2 Pseudocode of BRKGA

Step 1: Generate the initial population composed of p random-key vectors, where each component is a uniformly randomly generated real number in the interval [0, 1].

Step 2: Set the initial population as the *Current Population*.

Step 3: Apply the decoding procedure to each key vector in the *Current Population*.

Step 4: Compute the value of the objective function for each solution in the *Current Population*.

Step 5: Select the best p_e $(1 < p_e < p)$ solutions (designated elite) based on the values of the objective function from the *Current Population* and add them to the *New Population* that will be considered in the next iteration.

Step 6: Generate p_m $(1 < p_m < p))$ new random-key vectors as in ***Step 1***, called mutants, and add them to the *New Population* that will be considered in the next iteration.

Step 7: Generate the remaining $(p - p_e - p_m)$ vectors to complete the *New Population* that will be considered in the next iteration by crossing one of the p_e vectors corresponding to an elite solution with one of the $(p - p_e)$ vectors corresponding to one of the non-elite solutions in the *Current Population*.

Step 8: Set the the *New Population* as the *Current Population*.

Step 9: Iterate from ***Step 3*** while the stopping criteria are not satisfied.

Algorithm 3 Sorting algorithm

```
Input: $S[\,] = s_1, s_2, \ldots, s_N$;                    ▷ a vector permutation of $N$ items
 1: Get: $O[\,] = o(s_1), o(s_2), \ldots, o(s_N)$;         ▷ a vector of order numbers of items
 2: for $(i = 0;\ i < N - 1;\ i{+}{+})$ do
 3:     $copy1$;
 4:     $copy2$;
 5:     for $(j = i + 1;\ j < N;\ j{+}{+})$ do
 6:         if $(O[i] < O[j])$ then
 7:             $copy1 \leftarrow O[i]$;
 8:             $O[i] \leftarrow O[j]$;
 9:             $O[j] \leftarrow copy1$;
10:             $copy2 \leftarrow S[i]$;
11:             $S[i] \leftarrow S[j]$;
12:             $S[j] \leftarrow copy2$;
13:         end if
14:     end for
15: end for
Output: $S_{new} = S$;
```

Algorithm 4 Pseudocode of Simulated Annealing

```
Input: $S_o, T_o, T_f, \alpha, Len$;
Output: Best solution $S^*$;
 1: $S^* \leftarrow S_o$;
 2: $S \leftarrow S_o$;
 3: $T \leftarrow T_o$;
 4: while $T > T_f$ do
 5:     for $i \leftarrow 1$ to $Len$ do
 6:         $S' \leftarrow Swap(S)$;
 7:         $\triangle f = f(S') - f(S)$;
 8:         if $\triangle f < 0$ then
 9:             $S \leftarrow S'$;
10:         else
11:             if $r < e^{\frac{-\triangle f}{T}}$ then                    ▷ $r$: a random number in $[0,1]$
12:                 $S \leftarrow S'$;
13:             end if
14:         end if
15:         if $f(S') < f(S^*)$ then
16:             $S^* \leftarrow S'$;
17:         end if
18:     end for
19:     $T \leftarrow T * \alpha$;
20: end while
```

Algorithm 5 Pseudocode of Particle Swarm Optimization

```
 1: Initialize position and velocity vectors of particles;
 2: Set current positions of particles as their personal best positions;
 3: Determine global best;
 4: while stopping criterion not satisfied do
 5:     for $l \leftarrow 1$ to $P$ do                    ▷ $P$: number of particles
 6:         Update velocity of particle $l$ using equation (2);
 7:         Update position of particle $l$ using equation (3);
 8:     end for
    Updating personal best:
 9:     for $l \leftarrow 1$ to $P$ do
10:         if $f(X_l) < f(X_l^{best})$ then
11:             $X_l^{best} \leftarrow X_l$;
12:         end if
13:     end for
    Updating global best:
14:     $X_s^{best} \leftarrow X_1^{best}$
15:     for $l \leftarrow 2$ to $P$ do
16:         if $f(X_l^{best}) < f(X_s^{best})$ then
17:             $X_s^{best} \leftarrow X_l^{best}$;
18:         end if
19:     end for
20: end while
```