Thomas, Jaya; Chaudhari, Narendra S.

Article

A new metaheuristic genetic-based placement algorithm for 2D strip packing

Journal of Industrial Engineering International

Provided in Cooperation with:
Islamic Azad University (IAU), Tehran

Suggested Citation: Thomas, Jaya; Chaudhari, Narendra S. (2014) : A new metaheuristic genetic-based placement algorithm for 2D strip packing, Journal of Industrial Engineering International, ISSN 2251-712X, Springer, Heidelberg, Vol. 10, pp. 1-16, https://doi.org/10.1007/s40092-014-0047-9

J Ind Eng Int (2014) 10:47
DOI 10.1007/s40092-014-0047-9

ORIGINAL RESEARCH

A new metaheuristic genetic-based placement algorithm for 2D strip packing

Jaya Thomas · Narendra S. Chaudhari

Received: 1 August 2013 / Accepted: 18 January 2014 / Published online: 22 February 2014

Abstract Given a container of fixed width, infinite height and a set of rectangular block, the 2D-strip packing problem consists of orthogonally placing all the rectangles such that the height is minimized. The position is subject to confinement of no overlapping of blocks. The problem is a complex NP-hard combinatorial optimization, thus a heuristic based on genetic algorithm is proposed to solve it. In this paper, we give a hybrid approach which combined genetic encoding and evolution scheme with the proposed placement approach. Such a combination resulted in better population evolution and faster solution convergence to optimal. The approach is subjected to a comprehensive test using benchmark instances. The computation results validate the solution and the effectiveness of the approach.

Keywords Combinatorial optimization · Crossover · Fitness · Genetic algorithm · Operation research · Placement approach · Strip packing

Introduction

Manufacturing and production industries very often come across the problem where a given stock material must be cut into a smaller set of shapes. In this paper, we consider a two-dimensional orthogonal rectangular strip packing problem (SPP) NP-hard in nature (Garey and Johnson 1979; Bortfeldt 2013). This class of combinatorial optimization finds significant relevance in different domains of operation research. In industries like paper and pulp, wood and textile, the problem is to determine how the arbitrary rectangular block set would be cut from the available stock. The problem variant like arrangement of articles, reports and advertisement is considered in the newspaper field. In pharmaceutical packing industry, many strip packaging approach seems ideal for high-speed sealing of coated or uncoated tablets, capsules or lozenges of any shape or size in aluminum foils, polythene, cellophane, etc. It is an interesting real-world industrial problem where the objective is to provide the best arrangement with the aim of waste minimization. There are two broad categories of the solution approach, namely exact and inexact. The major bottleneck with the exact approaches are that as the problem size grow and become complex, the computation time also grows exponentially. Thus, the researchers focus more on the inexact approach in comparison to exact.

Industrial application

In today's world of industrialization, mass production and high material utilization are the crucial factors for growing industries that necessitate the need of finding correct cutting patterns which may result in a small improvement. In the long term, it leads to huge economical saving. These problems occur at wide scale in many industries such as wood, textile, rubber, glass, etc., where the complexity is determined by the shape of the item to be cut and the number of applicable constraints. Textile and shipping industries basically deal with irregular shaped items. Automation in a packing system saves much economic time and is preferred over the manual system. One case study discussing strip cuttings in wooden industries is

J. Thomas (✉) · N. S. Chaudhari
Department of Computer Science and Engineering, Indian Institute of Technology Indore, Indore, India
e-mail: jayathomassgsits@gmail.com

N. S. Chaudhari
e-mail: nsc183@gmail.com

discussed in Lefrançois and Gascon (1995) for a manufacturer of prefabricated doors and windows. These industries follow a daily production schedule where the desired length which may be wood, steel, plastic, etc., are cut to smaller length, either by automation or by manual labor. The policy followed is basically to meet the higher demand length first, followed by others in order of their decreasing length. The main concern for these industries is efficient utilization of available resources with minimum scrap production. The shorter length stock incapable of fulfilling any order demand is termed as scrap. The textile industry also tackles this problem, but under a different heading, namely Nesting and Marker making. It involves finding the best layout for the cutting of irregular shapes, where the shapes are allowed a rotation from 0° to 180°. Regular and irregular shape packing are addressed in the shipbuilding industries where it is required to investigate how the irregular size items can be packed and transported in huge containers. In today's scenario of surplus demand, an automated system is required for efficient packing thereby reducing the transportation damage risk. The packing problem is not limited to industries, but can also be seen in other dimensions like in very-large-scale integration design, memory allocation during storage and in the field of optical fiber communication.

The World Packaging Organization (http://www.worldpackaging.org/i4a/doclibrary/index.cfm?category_id=4) report presented in 2008 with global packaging market statistics shows packaging growth is tied to the world economy. The fast growing economy of countries in Asia and Eastern Europe has created new opportunities for packaging suppliers. The world packaging market was valued at $427 billion in 2003, growing at an annual rate of 3.5 % since 1999. The global market for industrial and bulk (transport) packaging was valued at $105 billion in 2004, representing an increase of 5 % in 2003. At $30.8 billion, North America represents the single largest market with a 30 % share—set to fall to 27 %, behind Asia by 2009. Overall, sales are forecast to grow at an average rate of 2 % over the period to reach $117 billion. Such growing industry calls for the use of intelligent automated system for quick and efficient packing approach, particularly for real-time applications where computation time is hard bound.

Our contribution

In this paper, we have proposed a metaheuristic algorithm, namely genetic algorithm-based placement approach (GPA) for solving orthogonal strip packaging problem. The approach modifies the complex placement strategies by proposing simple ones, whereas in the genetic algorithm we have modified the stages of initial population generation, crossover and mutation to achieve our objective. The paper presents a novel fitness function to evaluate the design layout, which facilitates the selection of a best design layout for population evolution. The evolution process of forwarding the best individual to the next generation is maintained by the appropriate selection pressure. In placement approach, we have considered the creation of empty rectangular blocks to place the next rectangular block in a sequence. As the problem grows, the number of such blocks is likely to increase. However, this problem of stacking of empty rectangle, i.e., it controls the growth in number is tackled in our approach by using rectangle merging routines. In comparison to other approaches, the computation task of fitness evaluation for the placed rectangle is effectively reduced. We have designed the fitness function for the overall layout pattern by considering two crucial parameters, height and area utilization factor. The algorithm finds the most optimal solution in numerous cases and results illustrates that it compete well with the existing heuristic and metaheuristic approaches. The experimental result obtained confirms the applicability on large-scale instances.

The rest of the paper is organized as follows. "Background study" outlines the literature review for 2D orthogonal SPP in light of exact, heuristic and metaheuristic approaches. "Proposed approach" presents the proposed work. The work discusses the placement policies, empty block creation, merging and various aspects of GA. "Experiments and results" gives the computation results as the work is compared against the standard benchmark, along with a detailed discussion of how it outperforms the existing approaches. In "Conclusion", we finally conclude our work by giving final remarks.

Background study

This subset of cutting and packaging problem is NP-hard in nature (Garey and Johnson 1979). Dyckhoff (1990) gave the first clear classification for cutting and packing problem. The characterization is based on the fact that many are similar in their logical structure, but different in application areas. There are various categories in packing which include cutting stock problem, knapsack, bin packing and loading problem. Cutting stock involves cutting off available raw stock to meet customer demand such that trim loss is minimized. Knapsack is mostly considered as a subproblem in many cases where certain weight is associated with the object. The objective is to pack these objects in a fixed size larger container to maximize the overall weight. Bin packing aims at packing items into bins. The dimensions are bounded, such that the remaining space in the used bin and the overall bin required to pack all items are

minimized. All these problems range from single to multidimensions. Strip packing is considered in higher dimensions like two (2D) and three (3D).

The literature reveals that rectangular strip packing is solved using many exact and inexact approaches. Christofides and Whitlock (1977) and Beasley (1985) used tree search methods to solve guillotine and non-guillotine variants of SPP. Exact approaches like a branch and bound were used by Martello et al. (2003). Lesh et al. (2004) modified the approach by adding the pruning method with branch and bound to solve a small subset of this problem. Another variant of branch and bound was presented by Kenmochi et al. (2009), where the branching scheme was based on some placement scheme and bounding operation was governed by dynamic and linear programming. However, a general observation follows with the exact approach, which is: as the number of rectangles to be packed grow in number, it is difficult to get the optimal solution in an acceptable time as the time also grows exponentially. This behavior of the exact motivates researchers to focus on heuristic and metaheuristic approaches to find optimal or near optimal solution. The benefit of using these approaches is their ability to give approximate solutions in a reasonable computing time. The most popular and common approach of placement bottom left heuristic (BL) was given by Baker et al. (1980). This approach considered sequential placement of rectangular blocks at the bottom left, where in the beginning each rectangle was placed at the top right corner and then slowly moved down to a position where it could not be further lowered; the block was then shifted toward left ensuring that it resulted in no rectangular overlap. The heuristic was improved by Hopper and Turton (2001) bottom left fill (BLF) heuristic in which the gap that existed in between the already placed rectangle was given preference over the placement of the new rectangle. A further improvement in these strategies was observed by Asik and Özcan (2009), who introduced a bidirectional best-fit (BBF) heuristic, sequencing the placement based on height or width of the rectangle. The method was studied by Imahori and Yagiura (2010), considering both horizontal and vertical gaps for the best placement location. They used the concept of placing rectangles, taking the reference of skyline, good searching technique to determine the placement position for the next rectangle to be packed and also used efficient data structure for storing the details of the remaining rectangles. Leung and Zhang (2011) proposed fast layer-based heuristic with the strategy of stacking rectangles. The approach determines the reference line by first placing a reference rectangle and then stacking some others on it. The techniques are based on a greedy search where the lowest available space is determined under the reference line; the fitness of the remaining unplaced rectangle is determined and based on the fitness value the best-fitted rectangle is selected. Özcan et al. (2013) presented an extension to the original BBF heuristic; rather than considering single rectangle placement, different combinations of rectangle pair were selected. The heuristic approach also includes brick layering (Zhang et al. 2008), which is inspired by the process of building the wall, as it involves a number of placement phenomena during creation of the wall. Here, we now discuss a number of metaheuristic approaches from literature which are also used for the evaluation of the proposed approach. These include genetic algorithms (Jakobs 1996; Ramesh Babu and Ramesh Babu 1999; Dowsland et al. 2006; Gonçalves 2007), simulated annealing algorithms (Dagli and Hajakbari 1990; Lai and Chan 1996; Martins and Tsuzuki 2010), neural network algorithms by Dagli and Poshyanonda (1997) and some hybrid metaheuristic algorithms by Hopper and Turton (2001), Hifi and M'Hallah (2003), Bortfeldt (2006).

High computing time is a major concern in hybrid neural-based model posed by Dagli and Hajakbari (1990). The scale factor concept was given by Martins and Tsuzuki (2010), where he used limited depth breath search and crystallization heuristic to fit the polygon in the container. Jakobs (1996) and Liu and Teng (1999) used GA for evolution of sequence for packing the rectangles, where GA was coupled with the bottom left heuristic. On the other hand, in another metaheuristic approach, Ramesh Babu and Ramesh Babu (1999) used deterministic heuristic in combination with GA. Hopper and Turton (2001) in their paper investigated a number of metaheuristic approaches which included simulation annealing and GA by combining them with a number of existing heuristics like BL and BLF. In addition, the non-deterministic algorithms are time-consuming and least applicable for problem dealing with huge enumeration of rectangle. Bortfeldt (2006) gave a sequence algorithm titled SPGAL that did not use any encoding and worked directly on the resolution layouts. Burke et al. (2009) enhanced the BF rule (Burke et al. 2004) with the crossbred simulated annealing and BLF rule of Hopper and Turton (2001). Alvarez-Valdes et al. (2009) planned an activated greedy randomized adaptive search procedure (GRASP) which improves for instances of smaller size as reported in Alvarez-Valdes et al. (2008). The main feature of this approach is its ability to fix suitable parameters by analysis and learning of information. Belov et al. (2008) presented two heuristics, namely sequential value correct (change knapsack job) [SVC (subKP)] and bubble search (BS, bottom left–right). The coercive rule of SVC proved to perform well in many instances. Wei et al. (2009) gave least waste first heuristic (LWF) based on the position evaluation which is improved after union with the simulated annealing algorithm. LWF outperforms many other approaches facing difficulty to

pack rectangles. The solution approaches like SPGAL, discernment, BF+SA, SVC and LWF, although are well-designed algorithm based on good formulating placement strategies and giving acceptable results in reasonable time. However, the computation time for these algorithms is subject to correct parameters setting. Moreover, as the problem becomes complex when the number of rectangles to be packed increases, the execution time for several algorithms generally rises to find an appropriate solution. In Leung and Zhang (2011), fast layer-based heuristic inspired by brick packing does not depend on parameter setting and was designed to handle large instance as by Huang et al. (2007). However, computation time is large for this approach. Wei et al. (2011) proposed iterative doubling binary search (IDBS) which when combined with tabu search outperformed many of the approaches from the literature. A number of new and improved level-based heuristics are reported in literature. Yang et al. (2013) recently presented a randomized algorithm based on least waste strategy with simplified parameter adaptation. It can quickly find a solution, but the quality of the solution is worth improving further. Efficiency and robustness are the two crucial factors for any algorithm applied to any real-world application, particularly in the logistic field.

Proposed approach

So far, the heuristic and metaheuristic approaches require a pre-processing on the available set of rectangles to select the best one among them which can fill the current gap exactly. Here, we propose a simple approach where the rectangles are placed in sequential order governed by placement policies in an appropriate empty rectangular block. Thus, it avoids the computational overhead of pre-processing and selection of the best-fit rectangle. The approach has major contributions like the rectangle placement strategy, a modified genetic algorithm and a novel fitness function for better convergence of optimum results. The new approach is constructive in nature which places each rectangular block one at a time in the container. The role of genetic algorithm is to evolve the ordering of the rectangle or chromosomes which represent the rectangular block packing sequence.

Two-dimensional strip packaging

The two-dimensional SPP (2D-SPP) deals with a set of rectangular blocks that must be arranged in a given container of fixed width and infinite height with the objective of minimizing the highest point of any rectangle in the solution. A feasible placement is one where no rectangles overlap one another within the container and are arranged parallel to the container edge, i.e., orthogonally. An additional constraint like rotation of blocks by 90° is not considered in this work.

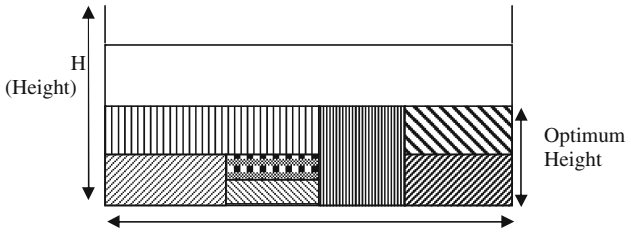

Fig. 1 A view of 2D strip packing

The 2D-SPP is presented as follows (similarly addressed by Thomas and Chaudhari 2013): consider a huge rectangular container of dimensions, say $W \times H$ where W stands for fixed width and H denotes the infinite height for the container. Consider m rectangles of smaller dimension as compared to the container to be packed, where the ith rectangle dimension is $w_i \times h_i$. The placement is subjected to non-guillotine cut, i.e., the rectangles are placed parallel to the edge of the container. Furthermore, they are not subjected to rotation by 90°. The problem can work with both integer and real dimensions easily without any extension or change. The placement problem is well explained with the help of Fig. 1. The shaded rectangle shows different arbitrary size rectangular blocks that are packed into the given container. The optimum height denotes the minimum possible height achieved after placing all the rectangles. The sub-problem for strip packing, i.e., the 0–1 knapsack is indeed NP-hard in nature. This degenerative case makes the entire problem NP-hard. The model can be formulated by maximizing the area occupancy that in turn results in minimizing the overall height of the layout which is as follows:

$$\max \sum_{i=1}^{m} w_i h_i \lambda_i. \tag{1}$$

λ_i can take the value 0 or 1 indicating whether the ith rectangle is placed or not. We designate each rectangle bottom the left most corner coordinate of the rectangle as (x_i, y_i). The governing constraints are defined as

$$x_i + w_i \le W \quad i = 1, \ldots, m \tag{1a}$$

$$y_i + h_i \le H \quad i = 1, \ldots, m \tag{1b}$$

$$\begin{array}{l} x_i + w_i \le x_j \text{ or } x_j + w_j \le x_i \text{ or} \\ y_i + h_i \le y_j \text{ or } y_j + h_j \le y_i \end{array} \quad \forall i, j \text{ where } i \ne j \tag{1c}$$

$$\lambda_i \in \{0, 1\} \tag{1d}$$

$$x_i, y_i \ge 0. \tag{1e}$$

The constraints 1a, 1b and 1e ensure that the rectangles are placed within the boundary of the designated container

used for packing. Constraint 1c checks the condition that no two rectangles overlap each other. Here, we have considered the generality that all the dimensions are integer. Constraint 1d indicates whether a rectangle is placed or not. The objective is to place the entire rectangle in the given area such that the occupancy is maximized and no constraints are violated. Thus, a design layout is feasible if it satisfies the above constraints.

Placement approach

The idea of the placement approach is to consider the given container to be divided into X and Y coordinates. The limit range of X-coordinate is from 0 to width (W) and that of Y-coordinate is from 0 to height (H) where theoretically H is considered to be of infinite height. However, practically we have set an upper limit for H based on the computed optimal height values. The placement of each rectangle results in the creation of new empty rectangular block space. The creation of empty space for rectangle placement is illustrated in Fig. 2. Each time a new rectangle is to be placed, one of the best-suited empty rectangle is selected from the available ones.

The selection of empty rectangle is governed based on the following constraints: area of the rectangle to be placed is less than or equal to the empty rectangle area. Among the empty rectangle fulfilling the premier first constraint, the one is chosen with the lowest Y-coordinate The third governing criteria in the selection are that if the first constraint is satisfied and there exist say more than one empty rectangle space with the same minimum Y-coordinate. Among all, the approach selects the one with the least X-coordinate.

Empty block creation

As the placement of the rectangle determines the empty block creation, we in this subsection discuss all possible cases for placement and their possible block creations.

Case 1 For Fig. 2a, the placement of initial rectangle into the container results in a number of empty block creations. E_1 denotes the block space of height equal to the placed rectangle height and width as W-place rectangle width. This creation of empty block is not essential because of E_2 creation, but such block is required when we deal with the general cases. Another empty block E_3 is created for the empty area above the placed rectangle. In the initial case, its width is W and height is equal to the assumed height—placed rectangle height.

Case 2 Figure 2b shows how the placement of the next rectangle results in the creation of new and merging/removal of old rectangles. The case discussed is when R_2 height is more than that of already placed R_1. The creation of E_1, E_2 and E_3 is the same as initial Case 1 and E_4 is the left section empty block created. The creation of new E_1 will remove the earlier block formed by the initial placing of R_1. The diagonal coordinates of E_4 block will be X-coordinate as (x_1 coordinate of R_1, y_2 coordinate of R_1) and the Y-coordinate as (x_1 coordinate of R_2, y_2 coordinate of R_2).

Case 3 Figure 2c shows the third placement case where the height of R_2 is less than that of the already placed R_1. In this case the new creation will replace most of the already existing empty rectangles.

The creation of empty block is tracked, as each iteration results in the creation of new as well as removal of some of the existing empty blocks. The rectangles removed are basically for the one which are selected for rectangle placement and for all those empty rectangle blocks that overlap with the newly created blocks. We have used a merge rectangle routine that merges the small rectangle formed during placement into a larger one. This routine keeps a check on the growing number and stacking of empty block rectangles.

Merge rectangle routine

The merge routine deals with two major cases. Let us consider the case where we have two empty rectangular block spaces, say E_1 and E_2 with the diagonal coordinates (x_{11}, y_{11}), (x_{12}, y_{12}) and (x_{21}, y_{21}), (x_{22}, y_{22}). The cases are as follows.

Case 1 Combine rectangles in which the difference of x_2-coordinate of first and the x_1-coordinate of the second is

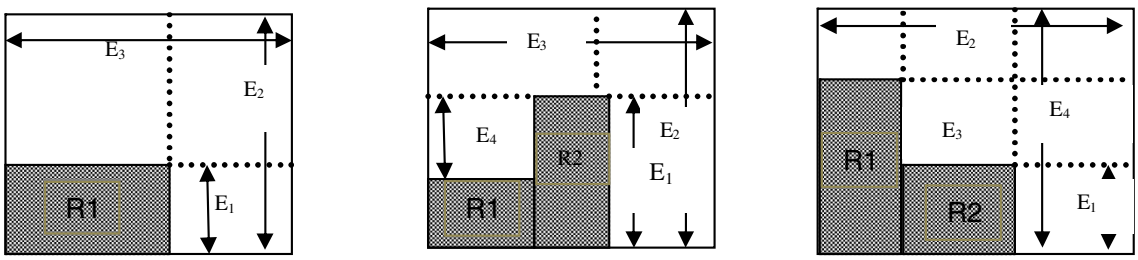

Fig. 2 Creation of empty rectangle

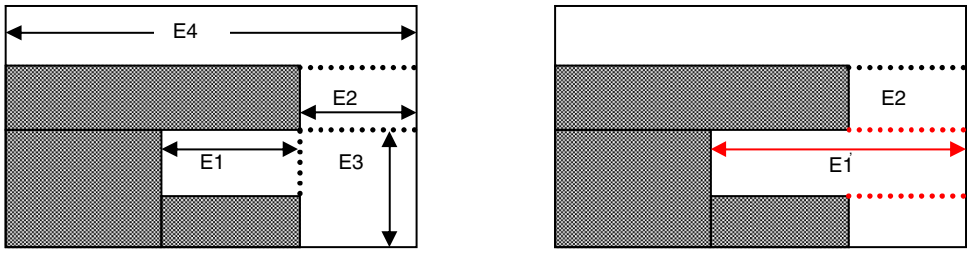

Fig. 3 Merging empty rectangular block (*X*-coordinate)

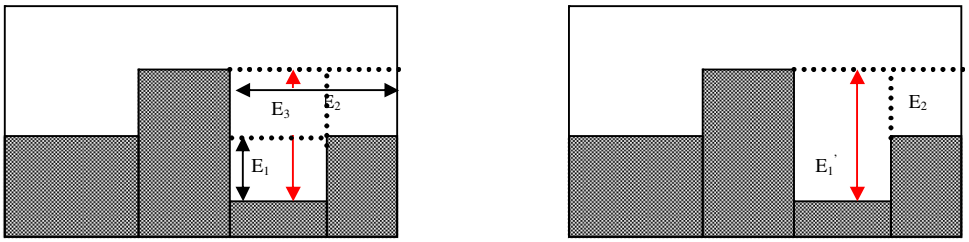

Fig. 4 Merging empty rectangular block (*Y*-coordinate)

zero, i.e., (x_{12}–x_{21}) or vice versa. The new empty block formed (E_1') as shown in Fig. 3 will have the diagonal coordinate as (x_{1*}, y_{1*}) and (x_{2*}, y_{2*}), where $x_{1*} = \min(x_{11}, x_{21})$, $x_{2*} = \max(x_{12}, x_{22})$ and $y_{1*} = y_{12}$ or y_{21}, $y_{2*} = \min(y_{12}, y_{22})$. Here, the convergence property of *X*-coordinate holds for E_1 and E_3.

Case 2 Combine rectangles having difference of y_2-coordinate of first and the y_1-coordinate of the second as zero (y_{12}–y_{21}) or vice versa. The new block coordinates as shown in Fig. 4 will be (x_{1*}, y_{1*}) and (x_{2*}, y_{2*}) where $x_{1*} = x_{11}$ or x_{21}, $x_{2*} = \min(x_{12}, x_{22})$ and $y_{1*} = \min(y_{11}, y_{21})$, $y_{2*} = \max(y_{12}, y_{22})$.

The merging module removes the smaller ones by combining smaller empty space into a large one, thus facilitating the placement of the larger rectangles at the lower *Y*-coordinate. The module also helps to overcome algorithm space limitations.

Genetic algorithm

GA is inspired by the Darwin's principle of survival of the fittest, where in the computing environment the stronger individual has a higher probability to be the survivor. GA uses a direct analogy to the biological process of evolution as shown in Fig. 5 and also discussed in Gómez and de la Fuente (2000). The process is based on the fact that individuals are the key elements for the potential solution to any problem, which can be represented by some feature set. These features are nothing, but genes corresponding to the chromosome which can be represented using data structure like string, binary or numeric array, tree structure or other representation. The evaluation of any chromosome to determine its feasibility in any population is based on its fitness value. The fitness value is related to the objective function.

```
Genetic Algorithm ()
{
        Initialization;
        Evaluation;
       While (! termination criterion reached)
        {
         Selection and Reproduction;
        Crossover;
         Mutation;
         Evaluation;
        }
}
```

Fig. 5 Overview of genetic algorithm

Biased random key-based genetic algorithm: an overview

For solving sequential problem, Bean (1994) introduced random key or also called random key generation algorithm. In this approach, representation of each chromosome is by randomly generated vectors having real values in the range lying between [0, 1]. The values obtained are sorted to obtain the chromosome sequence, i.e., in our case the rectangular block placement sequence into the container. The initial population is made up of p vector of m random keys, where p is the number of population in each generation G and m is the number of rectangular blocks to be placed. For each population, the fitness value of each individual is computed. The population is partitioned into two: elites (e) and non-elite. The elite is a small group containing individuals with maximum fitness values. The elite population is directly transferred from one generation to the other. The elite group size may vary from problem to problem. In this case we have set the group size to be 1/8 the total size of the population. We have kept this factor small to handle the worst case scenario where the

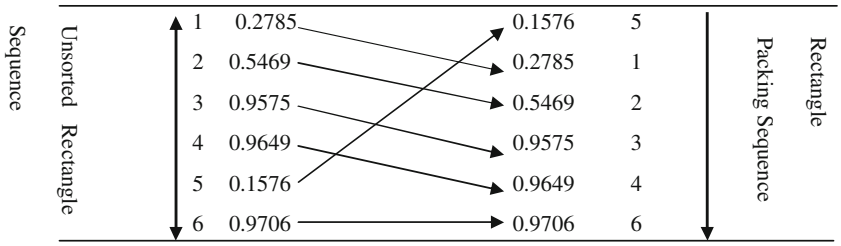

Fig. 6 Chromosome decoding

sequential pattern evolved may not be suboptimal or optimal. The elite population is carried forward to the next generation without change and directly plays a role in generating next population.

Thus, we keep a check so that not many chromosomes are transferred from one generation to another. To evolve the population for the next generation, the elite population is copied as it is to the next generation. The remaining *p*–*e* population is evolved from performing crossover between elite and non-elite chromosome as discussed crossover function section. A biased random key genetic algorithm proposed by Gonçalves and Resende (2011) has a different approach for selection of parents for mating. In this mating process, one parent from the elite population and another from a non-elite population are selected to produce the offspring. When the number of elites is less, it indicates that one parent can produce more than one offspring. The operation is detailed in the crossover section.

Chromosome representation

Each chromosome in the population represents a set of parameters to any problem that GA is trying to solve. The initial population in the proposed approach is generated by random strings of real numbers in the interval range of [0, 1] having the same length, where length represents the chromosome size. The chromosome size is nothing, but the number of strips to be packed in the given container. This evolutionary strategy was proposed by Bean (1994) and also used by Gonçalves (2007). The obtained random sequence is then sorted in an increasing order to generate the placement sequence for the strip. The placement sequence is illustrated in Fig. 6.

Crossover function

The placement sequence after sorting is further modified by the crossover operation. The evolution cycle for GA is enhanced by two fundamental operators: crossover and mutation. Here, we have considered biased random key method which integrates both these operators. The approach discussed by Gonçalves (2007), Goncalves and Resende (2011) says that rather than performing mutation on the entire population, certain mutants with specific mutation rates are introduced in the non-elite section of the population. In our case we have kept the mutation rate to be very low as the approach is capable of dealing with the low convergence rate problem by itself. Thus, we prefer the existing non-elite population than mutant introduction for crossover. As the elite parents are more feasible to be the solution for the given GA problem, BRKGA proceeds by selecting one random elite parent and another non-elite. Such a selection ensures that the new generated offspring would carry some features from the elite parent. As elite population individuals are less as compared to non-elite, repetition of parents is allowed. That is, one elite parent can generate multiple offsprings.

In contrast to the original approach, we have used maximal preservative crossover (MPX), as in Mathias and Whitley (1992), over the parameterized uniform crossover used by Gonçalves (2007) for the mating process of the chromosome. MPX is preferable as it produces offspring by directly copying a small segment from the elite parent to the offspring. In this crossover approach, two random crossover points are generated. These crossover points decide the range of segment that would be copied in the offspring from the elite parent. The remaining offspring is generated by copying the gene from the second parent provided that the gene does not already exist in the offspring.

For example in Fig. 7, while copying the gene from the second parent to the offspring, gene 7 is already present in the offspring so it is skipped and the next gene is evaluated and placed based on the nonexistence in the offspring. We assume that any random key vector can be decoded into a solution, and then the offspring resulting from mating is always valid, i.e., it can be decoded into a solution of the combinatorial optimization problem. When the next population is complete, i.e., when it has *p* individuals, fitness values are computed for all of the newly created random key vectors and the populations are partitioned into elite and non-elite individuals to start a new generation.

Fitness evaluation

The quality of any evolved solution is judged by its corresponding fitness value. These values help in the selection process, i.e., to select the fitter individual to be

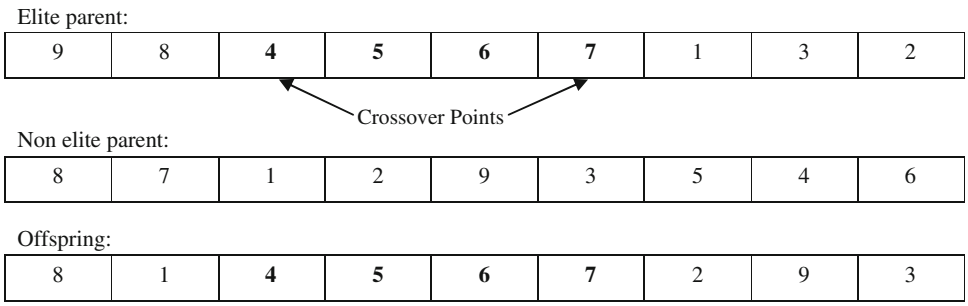

Fig. 7 Maximal preservative crossover (MPX)

the parent which can share their properties to evolve better offsprings for future generations. The goal is to minimize the packing height for the strip in the given container. Thus, the fitness evaluation of any individual is based on the height parameter. We have calculated individual score for each placed rectangle. These individual score for the rectangle is determined by the minimum space left in packing between the placed rectangles.

The overall score of the layout would be the summation of score for each individually placed rectangle. This layout score is considered in the evaluation of the fitness function. The score is given by

$$\text{Score}(r_i) = \frac{w_i h_i}{\left(WH - \sum_{i \neq j, j \in B} w_j h_j\right)} \tag{2}$$

where r_i is the rectangle with width w_i and height h_i placed in a container. W denotes the width of the container and H represents the height. In this problem, we assume that the container's height is infinite. However, to determine the score we set an approximate height of the container to a value slightly higher than the computed optimal height, to tackle worst cases as the ordering is random in nature. Here, B is the number of rectangles already placed. The optimal/expected height is calculated as

$$\text{optimal\_height} = \frac{\sum_i \text{Area}(r_i)}{W}. \tag{3}$$

The second parameter that we use for fitness evaluation is the best height (BH). This parameter indicates the height achieved by the layout. The fitness function must hold on both the aspects of the score and BH. Thus, utilization factor or score is not the only criterion for fitness value assignment. Hence, the fitness function is evaluated as

$$\text{Fitness}(j) = \text{score}(j) \times \text{BH}(j)\ j\text{th rectangle sequence pattern}. \tag{4}$$

Both score and BH are crucial in governing fitness function and are directly proportional to the fitness value of the layout.

Improvement on existing genetic algorithm

The genetic algorithm-based approaches such as those by Hopper and Turton (2001), Bortfeldt (2006) and Gonçalves (2007) show that the performance of the algorithm basically depends on the evolution of the rectangle placement sequence. But, in comparison to other approaches, we do not perform any pre-processing like sorting of rectangle based on maximum area, perimeter, etc. The selected MPX crossover as shown in Fig. 9 operation helps us to find a different ordering sequence to determine a better solution, and thus improve the convergence rate of the algorithm towards optimum. Figure 8 shows the various stages in the GPA algorithm. The algorithm has two main computation stages, one for a number of generations which involves the initial population, crossover and fitness evaluation and another for the empty block creation, selection and removal. For each population, the rectangle block placed is governed by simple placement policy. The crossover operation helps us to maintain the ordering sequence, and thus helps to improve the height generation after generation. Each population is assigned a fitness value that helps in further population evolution. Another improvement shown over existing GA algorithm is the reduced computation time to obtain the optimal solution and the ability to handle large instances. In the next section, we further compare our results with the latest algorithms on benchmark dataset (Fig. 9).

Experiments and results

Parameter selection: the population size is considered as ten with the number of generations 50. The mutation probability is set as low as 0.3. The results from various existing recently and published algorithms are used to verify the performance of GPA on the benchmark dataset. Most data instances used have known optimal solution and zero trim loss. The details of dataset with known optimal height used for comparison are as follows:

Jakobs (1996): two small instances $J1$ and $J2$ with the number of rectangles to be placed being 25 and 50, respectively.

Fig. 8 GPA algorithm

```
1: procedure GENETIC PLACEMENT APPROACH GPA ALGORITHM()
2:     repeat
3:         if not initial generation then
4:             Select 1/8 of the fittest offspring for next generation        ▷ elite population
5:         else
6:             Generate random initial population
7:         end if
8:         repeat
9:             MPX_CROSSOVER(parent1, parent2)        ▷ Decoding of rectangle packing sequence
10:                                ▷ Use of placement approach to obtain sequence for packing of rectangle
11:            Find empty block satisfying constraint 1 Amongst them select the one satisfying constraint 2 or
    3 based on lowest Ycoordinate
12:            Creation of new empty block
13:            call Merge Rectange Routine
14:            Use novel function to compute fitness of population        ▷ Evaluation of fitness function
15:        until number of population
16:    until number of generation
17: end procedure
```

Fig. 9 MPX_crossover

```
1: procedure MPX_CROSSOVER(parent1, parent2)
2:     Find two crossover point from elite population
3:     Copy the segment from elite to the offspring
4:     Scan the non elite parent to fill remaining gene of the offspring
5:     if gene already present in the offspring then
6:         Move to the next gene segment        ▷ Skipping already placed gene
7:     else
8:         Copy the gene to offspring
9:     end if
10: end procedure
```

Ramesh Babu and Ramesh Babu (1999): a single instance with 50 rectangles to be placed.

Hopper and Turton (2001): test set *C* containing 21 instances. The instances are divided into seven categories, with each category containing three instances with the number of rectangles to be packed ranging from 16 to 197. These groups are categorized on the basis of similarity in achieving optimum height and container width.

Wang and Valenzela (2001): the dataset contains real values of the rectangle dimension which are rounded down to an integer by multiplying by 10. In "Comparison with genetic based approach", while comparing with the GA-based approach, we have considered no rounding up odd dimensions. The dataset discusses two different types of instances grouped into two categories of nice and path. Nice contains data with similar dimensions; on the other hand, path has a rectangle with varying dimensions. The rectangle range in both the dataset is from 25 to 1,000.

Burke et al. (2004): Burke generated test instance to test the best-fit (BF) algorithm. The number of rectangles to be placed is subsequently increased from as small as 10 (i.e., *N*1) to too large as 3,152 (i.e., *N*13).

Figure 10a, b shows the results of 50 runs of the algorithm, all for a maximum of 75 generations, but terminating if the fitness function attained the value 150 (which is the optimum height) for *N*9 benchmark dataset. The optimum value was attained in 22 runs of 50. The algorithm gives optimal and suboptimal solution in most of the runs. The value above 153 is found in only one run. The number of generations taken to attain the 'optimum' value appears independent of the value of the optimum, but dependent on problem size.

Comparison of the proposed approach with the existing heuristic and metaheuristic approach

Table 1 reports the computational results for three different data instances. It shows a comparation table where the

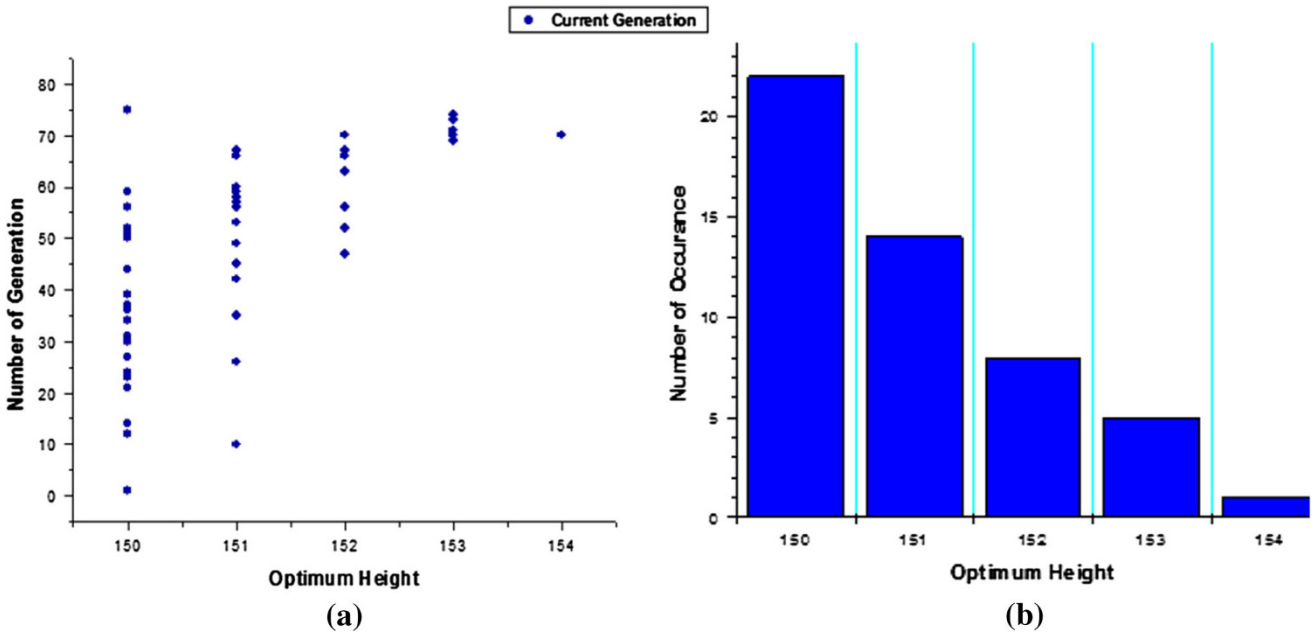

Fig. 10 **a** Number of generations. **b** Frequency of occurrence of different values of best fitness in 50 runs of the algorithm

Table 1 Computational results for Jakobs, Ramesh Babu and *C* test instance

	Instance			BL-DH	BLF-DH	BF	FH	BBF	BBFM	GPA	Time (s)
	m	*W*	Opt								
J1	25	40	15	–	–	–	–	16	15	15	30
J2	50	40	15	–	–	–	–	16	15	15	25
RB	50	1,000	375	–	–	400	–	400	375	375	50
C1.1	16	20	20	23	22	21	20	21	20	20	23
C1.2	17	20	20	22	22	22	20	21	21	20	18
C1.3	16	20	20	21	21	24	21	21	21	20	27
C2.1	25	40	15	17	17	16	16	16	16	15	36
C2.2	25	40	15	26	26	16	15	17	15	16	29
C2.3	25	40	15	17	17	16	15	16	16	16	34
C3.1	28	60	30	33	33	32	31	32	30	30	40
C3.2	29	60	30	33	32	34	31	33	31	30	31
C3.3	28	60	30	34	34	33	32	33	32	30	46
C4.1	49	60	60	67	66	63	61	62	62	60	56
C4.2	49	60	60	68	63	62	61	63	61	60	59
C4.3	49	60	60	64	63	62	61	62	61	61	62
C5.1	73	60	90	94	94	93	91	91	91	90	100
C5.2	73	60	90	99	95	92	90	92	91	90	89
C5.3	73	60	90	97	94	93	91	92	91	91	121
C6.1	97	80	120	130	126	123	121	123	121	120	220
C6.2	97	80	120	130	123	122	121	123	122	120	245
C6.3	97	80	120	131	128	124	121	123	121	120	231
C7.1	196	160	240	252	249	247	241	243	242	241	400
C7.2	197	160	240	264	247	244	241	242	242	240	452
C7.3	196	160	240	257	249	245	241	243	241	240	470

performance of the proposed approach is compared to the well-known existing algorithm, like BL-DH, BLF-DH, BF and so on, as discussed in the literature section of the paper. In comparison to many existing heuristic approaches, GPA is able to find the optimal solution for most of the instances reported. On Jakobs and Ramesh Babu data instance, the

Table 2 Computational result for test instance by Burke

	Instance			BF	BF+SA	FH	BBF	BBFM	Proposed approach (GPA)	Time (s)
	m	*W*	Opt							
*N*1	10	40	40	45	40	40	40	40	40	19
*N*2	20	30	50	53	50	52	52	50	50	15
*N*3	30	40	40	52	51	51	52	52	51	29
*N*4	40	80	80	83	82	83	82	82	80	70
*N*5	50	100	100	105	103	102	103	102	101	62
*N*6	60	50	100	103	102	101	102	101	100	90
*N*7	70	80	100	107	104	102	106	105	103	124
*N*8	80	100	80	84	82	81	82	81	80	96
*N*9	100	50	150	152	152	151	152	151	150	340
*N*10	200	70	150	152	152	151	151	151	151	589
*N*11	300	70	150	152	153	151	151	150	150	545
*N*12	500	100	300	306	306	301	302	302	300	842
*N*13	3,152	640	960	964	964	960	964	960	960	1,200

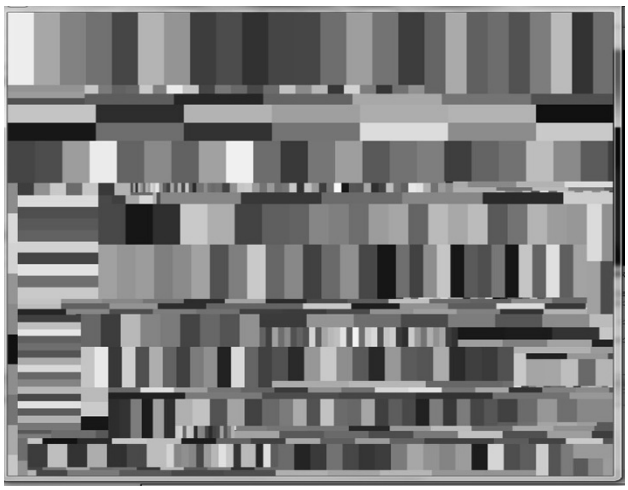

Fig. 11 Optimum packing for *N*13 (Burke et al.)

approach is at par with BBFM Özcan et al. (2013), both producing the optimum results. On C data set instance apart from C2.2, C2.3, C4.3 and C5.3 having a slight gap of one unit from the optimal, the results in all other remaining instances are optimum. It is further observed that GPA outperforms other approaches even for large-scale instances. The results marked in bold shows the distinction for the optimum solution found.

Table 2 shows the result for the Burke dataset N. The proposed approach is compared to 13 instances with the well-known BF approach and other methods. The method BF+SA is considered as the best algorithm among all the variants of BF+metaheuristic, fast heuristic better on *N* large instances, BBF and its modified version (BBFM). The table analyzes the result on all the 13 instances with respect to the solution found and GPA computation time. BF and BF+SA give nearly the same performance in many instances, whereas FH, BBF and BBFM improve the solution which is furthermore improved by our approach. Not only for small instances the algorithm performs comparatively better, but also for large instances, like *N*12 and *N*13, GPA is meta-heuristic, is able to find optimum solution and outperform the existing approaches. Figure 11 shows the optimum packing sequence for the *N*13 instance. Now we consider non-zero waste Valenzuela et al. nice and path instances. The result of comparing the GPA on these non-zero trim losses over the heuristic approach is reported in Table 3.

Table 3 Computational results for nice and path test instance (Valenzuela et al.)

	Instance			BF	BBF	BBFM	GPA approach
	m	*W*	Opt				
Nice1	25	1,000	1,000	1,074	1,083	1,069	1,070
Nice2	50	1,000	1,000	1,085	1,079	1,068	1,000
Nice3	100	1,000	1,000	1,070	1,067	1,063	1,040
Nice4	200	1,000	1,000	1,053	1,053	1,038	1,039
Nice5	500	1,000	1,000	1,035	1,033	1,024	1,000
Nice6	1,000	1,000	1,000	1,037	1,037	1,012	1,006
Path1	25	1,000	1,000	1,101	1,091	1,091	1,091
Path2	50	1,000	1,000	1,138	1,074	1,074	1,074
Path3	100	1,000	1,000	1,073	1,073	1,073	1,070
Path4	200	1,000	1,000	1,041	1,053	1,053	1,030
Path5	500	1,000	1,000	1,037	1,032	1,031	1,031
Path6	1,000	1,000	1,000	1,028	1,028	1,026	1,008

It is observed among the 12 instances that the reported GPA finds the optimal solution for eight instances. In the entire 12 instances reported, the GPA finds the smallest height and outperforms all the existing heuristic approaches within reasonable time. GPA performed worse only for one instance, i.e., Path 5. The analysis of this worst case behavior for GPA was observed to be due to a large number of small-scale instances and the variation in

Table 4 Computational results for large test instances

	Instance			GRASP	SVC	FH	GPA
	m	*W*	Opt				
NiceL1	1,000	100	100.1	102.2	101.5	100.9	100.3
NiceL2	2,000	100	100.1	101.5	100.7	100.5	100.2
NiceL3	5,000	100	100.1	101	100.4	100.2	100.0
PathL1	1,000	100	100.1	101.9	101.2	100.7	100.6
PathL2	2,000	100	100.1	101.5	101	100.2	100.0
PathL3	5,000	100	99.9	101	100.2	100	100.0

dimension was quite less. However, on large instances the approach gives an optimal solution. Consider the large non-zero trim loss dataset with real dimensions. Table 4 shows the computational results on NiceL1–NiceL3 and PathL1–PathL3 for FH, GRASP and SVC. As reported, GPA efficiently solves and gives an acceptable performance.

Table 5 makes a comparison against existing approaches in terms of % gap where it is defined as the ratio of (obtained solution − optimal height)/optimal height multiplied by 100. The results are reported for the Hooper and Turner data instance from the literature along with the total number of optimal solutions found. Further, these calculations are used for the statistical analysis of the approach.

The % gap is computed for well-known techniques from the literature. % gap is computed against different approaches where BF is used coupled with other techniques. BF+SA is one of the most used heuristic approaches that gave comparable results than BS+GA, BS+TS. A nonsystematic search technique like squeaky wheel optimization by Burke et al. (2011), iteratively single constructive heuristic SVC (subKP), reactive GRASP, stochastic approach like intelligent search algorithm (ISA), and IDBS combined with tabu search is also used for the comparison. IDBS is till now state of the art, outperforming the entire algorithm. Our approach finds the optimal solution in 16 out of 21 cases and is far better than existing approaches and at par with IDBS in most of the cases. In instances like C4, C5, C6 (all instances), C7.2, C7.3, the GPA is able to find the optimum height where other heuristic approaches fail. In other cases, the solution is near to optimal and misses only by a single unit. GPA stands second in finding the number of optimal solutions (i.e., 19) after IDBS which finds the optimal result for all the data instances. Our meta-heuristic approach is reported based on the average number of runs to find the optimal solution.

Statistical analysis for the heuristic approach

To validate the results, an ANOVA is applied to check if the observed differences between the algorithms are statistically significant. This will help to identify which algorithm and scenarios perform the best. ANOVA works by comparing the variation between groups to the variation

Table 5 Performance in terms of % gap and optimum result found

Instance	BF	BBF	BF-SA	FH	BBFM	SWP	SVC	GRASP	ISA	IDBS	GPA
C1.1	5.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C1.2	10.00	5.00	0.00	0.00	5.00	5.00	5.00	0.00	0.00	0.00	0.00
C1.3	20.00	5.00	0.00	5.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00
C2.1	6.67	6.67	6.67	6.67	6.67	6.67	0.00	0.00	0.00	0.00	0.00
C2.2	6.67	13.33	6.67	0.00	0.00	0.00	0.00	0.00	0.00	0.00	6.67
C2.3	6.67	6.67	6.67	0.00	6.67	0.00	0.00	0.00	0.00	0.00	0.00
C3.1	6.67	6.67	3.33	3.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C3.2	13.33	10.00	3.33	3.33	3.33	3.33	3.33	3.33	3.33	0.00	0.00
C3.3	10.00	10.00	3.33	6.67	6.67	0.00	0.00	0.00	0.00	0.00	0.00
C4.1	5.00	3.33	1.67	1.67	3.33	1.67	1.67	1.67	1.67	0.00	0.00
C4.2	3.33	5.00	1.67	1.67	1.67	1.67	1.67	1.67	1.67	0.00	0.00
C4.3	3.33	3.33	1.67	1.67	1.67	1.67	1.67	1.67	1.67	0.00	0.00
C5.1	3.33	1.11	1.11	1.11	1.11	1.11	1.11	1.11	1.11	0.00	0.00
C5.2	2.22	2.22	1.11	0.00	1.11	1.11	1.11	1.11	1.11	0.00	0.00
C5.3	3.33	2.22	2.22	1.11	1.11	1.11	1.11	1.11	1.11	0.00	0.00
C6.1	2.50	2.50	1.67	0.83	0.83	1.67	0.83	1.67	0.83	0.00	0.00
C6.2	1.67	2.50	0.83	0.83	1.67	0.83	0.83	1.67	0.83	0.00	0.00
C6.3	3.33	2.50	1.67	0.83	0.83	1.67	0.83	1.67	0.83	0.00	0.00
C7.1	2.92	1.25	1.67	0.42	0.83	1.25	0.83	1.67	0.83	0.00	0.42
C7.2	1.67	0.83	1.67	0.42	0.83	0.83	0.83	1.25	0.42	0.00	0.00
C7.3	2.08	1.25	2.08	0.42	0.83	1.25	0.83	1.25	0.83	0.00	0.00
# of optimum	0	0	3	5	3	6	7	8	8	21	19

within groups. Tukey's multiple comparison test is one of several tests that can be used to determine which means among a set of means differ from the rest. Tukey's multiple comparison test is also called Tukey's honestly significant difference test or Tukey's HSD.

Figure 12 shows the mean plot and Tukey CI for the evaluated algorithm where the mean values are encircled a and line is used for showing the respective confidence interval. Statistical analysis reveals that the proposed approach significantly differs from almost all the approaches except ISA and IDBS. The computational results and statistical analysis show the acceptable performance of the proposed GPA algorithm.

Comparison with genetic-based approach

The approach being metaheuristic is also compared to similar genetic-based approach based on the relative difference from the optimal solution and a statistical analysis is carried to justify the performance of GPA in the next section. Hopper and Turton (2001) investigated simulated annealing, naïve evolution and GAs metaheuristics approach in combination with the placement strategies of BL and BLF such as GA+BLF, SA+BLF, NE+BLF. Table 6 reports the comparison against such algorithm along with SPGAL, Iori et al. and the Goncalves approach. SPGAL is better and clearly dominates the comparison against the other earlier approaches with the % gap of just 1. HPA gives a better performance, but the result reported is the average result on the basis of the number of runs for each instance. The performance of GPA is best without averaging the number of runs.

Statistical analysis

Figure 13 shows the statistical analysis of different metaheuristic approaches. GPA shows a vast deviation from the mentioned approaches as it is able to find the optimal result in most of the cases. The plotting also shows that HPA and GPA are statistically not much different.

Algorithm complexity

The analysis of algorithms is considered for both space and time. Linear storage space is required for storing the parameters corresponding to the rectangle being packed

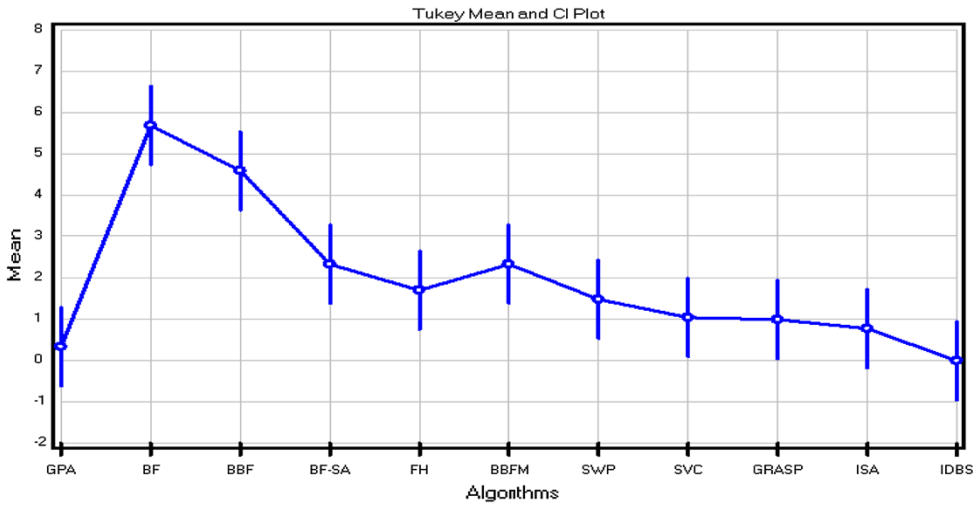

Fig. 12 Means plot and Tukey confidence intervals (CI) for the evaluated algorithms

Table 6 Comparison of % gap for genetic based approaches for Hooper and Turton test instance

Class	Hooper and Turton (2001)			Iori et al. (2002)	Bortfeldt (2006)	Gonçalves (2007) HPA	GPA
	GA+BLF	NE+BLF	SA+BLF				
C1	4.0	5.0	4.0	1.59	1.7	0	0
C2	7.0	7.0	6.0	2.08	0.9	0	0
C3	5.0	4.0	5.0	2.15	2.2	0.53	2.2
C4	3.0	4.0	3.0	4.75	1.4	0.70	0
C5	4.0	4.0	3.0	3.92	0.0	0.33	0
C6	4.0	4.0	3.0	4.0	0.7	0.42	0
C7	5.0	5.0	4.0	–	0.5	0.66	0.14
Average	4.6	4.7	4.0	3.08	1.0	0.38	0.33

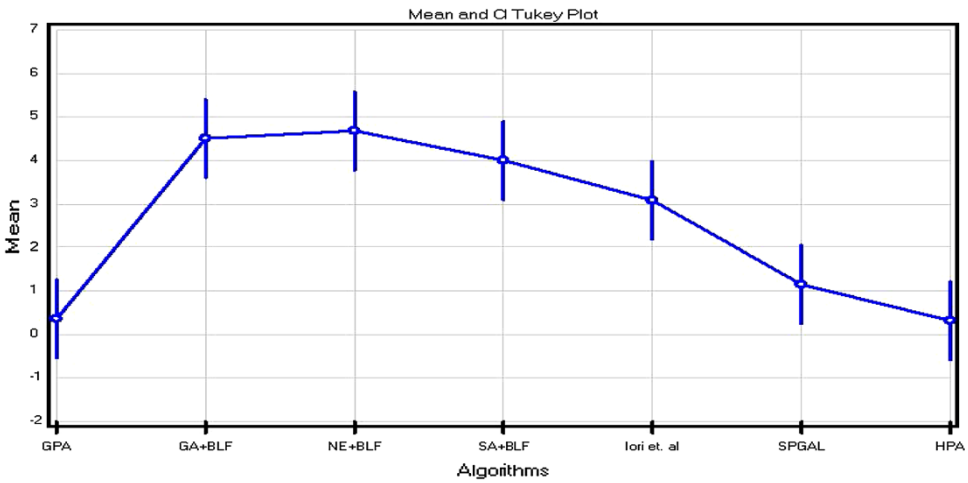

Fig. 13 Means plot and Tukey confidence intervals (CI) for the metaheuristic algorithms

and current status of the designed layout. The algorithm is analyzed for all the cases. In the best case, the initial placement sequence generated is the required one, thus the complexity of the algorithm is $O(1)$. The algorithm does not use level-oriented packing. The average and worst case behavior where the population is evolved to find the optimal pattern sequence is $O(n^2)$.

Conclusion

In this paper, a metaheuristic solution is proposed for the 2D-SPP. The approach couples GA approach with a proposed placement strategy to obtain the optimum solution. To our knowledge this is a first GA-based paper that provides optimal solution for such large instances of dataset. The proposed strategy is simple without inclusion of problem-specific operations. MPX crossover is used to preserve the mutual relation between rectangles and provides sufficient flexibility to search. The performance of metaheuristic is often governed by the selection of initial parameters. However, in this case the selection parameters are not crucial; only the crossover and evaluation of fitness function play an important role in the evolution. On the basis of exhaustive computation carried out on a benchmark dataset from literature, we conclude that the performance is far better in comparison to many known heuristic and metaheuristic approaches discussed in the paper. Our metaheuristic gives high-quality solutions in a reasonable computational time. GPA proves its dominance even in the larger instances. Obtained results motivate the use of these techniques in industries for more economical saving.

Appendix 1

The phases of GPA are discussed below:

Initial population: based on the random values generated, the initial population would be generated. This initial population is used to form the layout design and the fitness value is computed.

See Table 7; Fig. 14.

Table 7 Initial population

1	2	3	4	5	6	7	8	…	50
14	4	2	38	15	48	11	13	…	12
3	7	28	40	27	23	38	21	…	50
16	7	42	33	40	48	17	30	…	13
47	22	19	21	5	33	40	1	…	10
13	1	24	7	44	19	39	5	…	16
32	10	19	48	45	16	4	21	…	14
28	50	25	14	39	38	2	10	…	1
5	43	11	15	25	41	49	3	…	19

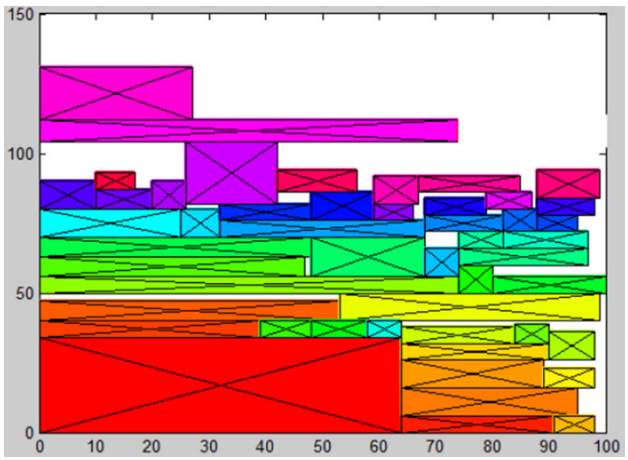

Fig. 14 Initial plot for Burke *N*5

Fitness function

The fitness value for each generation is evaluated taking into consideration all the populations. Few of the fitness values computed for the initial population is shown in the table below:

116.6326	122.8424	125.3914	130.4357	135.4801	150.98	177.86	194.6

In our case, the layout design with the lowest fitness value is fittest.

Crossover

In crossover one of the elite parent will be from 2, 5 because of their fitness values.

16	22	1	38	15	48	.....	9

Final evolution

See Fig. 15.

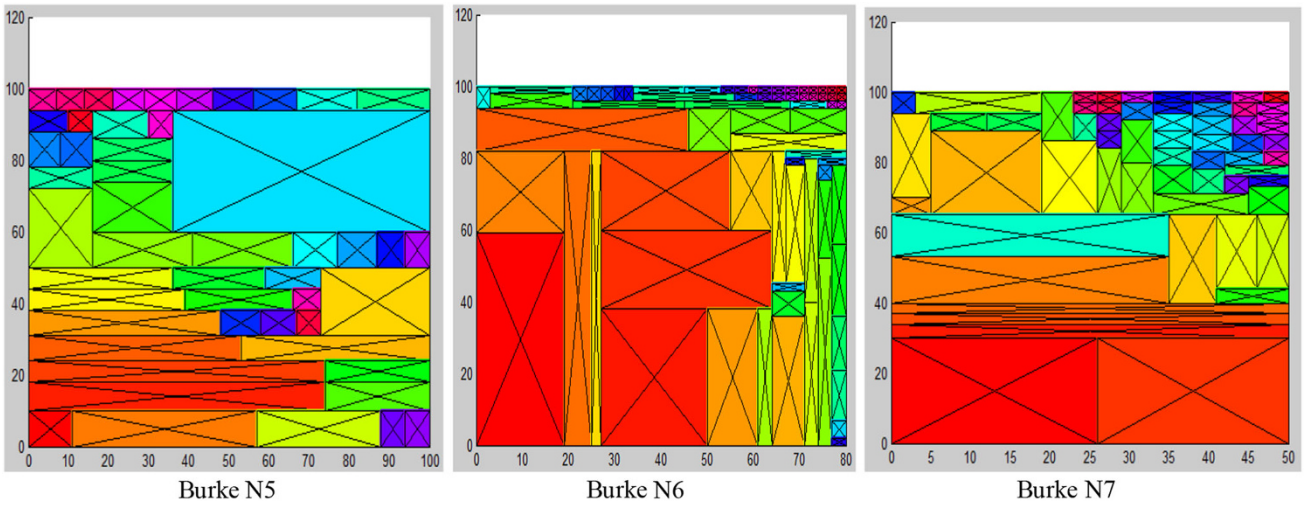

Fig. 15 Optimum results for Burke *N*5, Burke *N*6 and Burke *N*7

References

Alvarez-Valdes R, Parreño F, Tamarit JM (2008) Reactive GRASP for the strip packing problem. Comput Oper Res 35:1065–1083

Alvarez-Valdes R, Parreño F, Tamarit J (2009) A branch and bound algorithm for the strip packing problem. OR Spectr 31:431–459

Asik OB, Özcan E (2009) Bidirectional best-fit approach for orthogonal rectangular strip packing. Ann Oper Res 172:405–427

Baker B, Coffman EG Jr, Rivest RL (1980) Orthogonal packing in two dimension. SIAM J Comput 9:846–855

Bean JC (1994) Genetics and random keys for sequencing and optimization. ORSA J Comput 6:154–160

Beasley JE (1985) An exact two-dimensional non-guillotine cutting tree search procedure. Oper Res 33:49–64

Belov G, Scheithauer G, Mukhacheva EA (2008) One-dimensional heuristics adapted for two-dimensional rectangular strip packing. J Oper Res Soc 59:823–832

Bortfeldt A (2006) A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces. Eur J Oper Res 172:814–837

Bortfeldt A (2013) A reduction approach for solving the rectangle packing area minimization problem. Discret Optim Eur J Oper Res 224:486–496

Burke E, Kendall G, Whitwell G (2004) A new placement heuristic for the orthogonal stock-cutting problem. Oper Res 52: 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 J Comput 21:505–516

Burke EK, Hyde MR, Kendall G (2011) A squeaky wheel optimisation methodology for two-dimensional strip packing. Comput Oper Res 38:1035–1044

Christofides N, Whitlock C (1977) An algorithm for two-dimensional cutting problems. Oper Res 25:30–44

Dagli CH, Hajakbari A (1990) Simulated annealing approach for solving stock cutting problem. In: Proceedings of IEEE international conference on systems, man, cybernetics, Los Angeles. IEEE, Washington, DC, pp 221–223

Dagli CH, Poshyanonda P (1997) New approaches to nesting rectangular patterns. J Intell Manuf 8:177–190

Dowsland KA, Herbert EA, Kendall G, Burke EK (2006) Using tree search bounds to enhance a genetic algorithm approach to two rectangle packing problems. Eur J Oper Res 168:390–402

Dyckhoff H (1990) A topology of cutting and packing problems. Eur J Oper Res 44(2):145–159

Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. New York

Gómez A, de la Fuente D (2000) Resolution of strip-packing problems with genetic algorithms. J Oper Res Soc 51(11):1289–1295

Gonçalves JF (2007) A hybrid genetic algorithm-heuristic for a two-dimensional orthogonal packing problem. Eur J Oper Res 183:1212–1229

Gonçalves JF, Resende MGC (2011) Biased random-key genetic algorithms for combinatorial optimization. J Heuristics 7:487–525

Hifi M, M'Hallah R (2003) A hybrid algorithm for the two-dimensional layout problem: the cases of regular and irregular shapes. Int Trans Oper Res 10:1–22

Hopper E, Turton BCH (2001) An empirical investigation of meta-heuristic and heuristic algorithms for a 2d packing problem. Eur J Oper Res 128:34–57

Huang W, Chen D, Xu R (2007) A new heuristic algorithm for rectangle packing. Comput Oper Res 34(11):3270–3280

Imahori S, Yagiura M (2010) The best-fit heuristic for the rectangular strip packing problem: an efficient implementation and the worst case approximation ratio. Comput Oper Res 37:325–333

Iori M, Martello S, Monaci M (2002) Metaheuristic algorithms for the strip packing problem. In: Pardalos P, Korotkich V (eds) Optimization and industry: new frontiers. Kluwer Academic Publishers

Jakobs S (1996) On genetic algorithms for the packing of polygons. Eur J Oper Res 88:165–181

Kenmochi M, Imamichi T, Nonobe K, Yagiura M, Nagamochi H (2009) Exact algorithms for the two-dimensional strip packing problem with and without rotations. Eur J Oper Res 198:73–83

Lai KK, Chan JWM (1996) Developing a simulated annealing algorithm for the cutting stock problem. Comput Ind Eng 32(1):115–127

Lefrançois P, Gascon A (1995) Solving a one-dimensional cutting-stock problem in a small manufacturing firm: a case study. IIE Trans 27:483–496

Lesh N, Marks J, McMahon A, Mitzenmacher M (2004) Exhaustive approaches to 2d rectangular perfect packings. Inf Process Lett 90:7–14

Leung S, Zhang D (2011) A fast layer-based heuristic for non-guillotine strip packing. Expert Syst Appl 38(10):13032–13042

Liu D, Teng H (1999) An improved bl-algorithm for genetic algorithms of the orthogonal packing of rectangles. Eur J Oper Res 112:413–420

Martello S, Monaci M, Vigo D (2003) An exact approach to the strip-packing problem. INFORMS J Comput 15:310–319

Martins TC, Tsuzuki MSG (2010) Simulated annealing applied to the irregular rotational placement of shapes over containers with fixed dimensions. Expert Syst Appl 37(3):1955–1972

Mathias K, Whitley D (1992) Genetic operators, the fitness landscape and the traveling salesman problem. Parallel problem solving from nature, vol 2. North-Holland, Amsterdam, pp 219–228

Özcan E, Kai Z, Drake JH (2013) Bidirectional best-fit heuristic considering compound placement for two dimensional orthogonal rectangular strip packing. Expert Syst Appl 40:4035–4043

Ramesh Babu A, Ramesh Babu N (1999) Effective nesting of rectangular parts in multiple rectangular sheets using genetic and heuristic algorithms. Int J Prod Res 37(7):1625–1643

Thomas J, Chaudhari NS (2013) Hybrid approach for 2D strip packing problem using genetic algorithm. In: International work conference on artificial neural network (IWANN'13), Part I, LNCS 7902. Springer, Berlin, pp 566–574

Wang PY, Valenzuela CL (2001) Data set generation for rectangular placement problems. Eur J Oper Res 134(2):150–163

Wei L, Zhang D, Chen Q (2009) A least wasted first heuristic algorithm for the rectangular packing problem. Comput Oper Res 36(5):1608–1614

Wei L, Oon W-C, Zhu W, Lim A (2011) A skyline heuristic for the 2d rectangular packing and strip packing problems. Eur J Oper Res 215:337–346

Yang S, Han S, Ye W (2013) A simple randomized algorithm for two-dimensional strip packing. Comput Oper Res 40:1–8

Zhang D, Han S, Ye W (2008) A bricklaying heuristic algorithm for the orthogonal rectangular packing problem. Chin J Comput 23(3):509–515