AN APPLICATION OF HEURISTIC AND EVOLUTIONARY ALGORITHMS FOR SOLVING SOME PRACTICAL TWO-DIMENSIONAL BIN PACKING PROBLEMS

Andile Ntanjana
Department of Mechanical Engineering
Durban University of Technology
Durban, South Africa
ntanj2009@yahoo.com

Sibusiso Moyo
Institute of Systems Science
Durban University of Technology
Durban, South Africa
moyos@dut.ac.za

Pavel Y. Tabakov
Department of Mechanical Engineering
Durban University of Technology
Durban, South Africa
pashat@dut.ac.za

ABSTRACT
Bin packing problems are a class of optimization problems that have numerous applications in the industrial world, ranging from efficient cutting of material to packing various items in a larger container. We consider here only rectangular items cut off an infinite strip of material as well as off larger sheets of fixed dimensions. This problem has been around for many years and a great number of publications can be found on the subject. Nevertheless, it is often difficult to reconcile a theoretical paper and practical application of it. The present work aims to create simple but, at the same time, fast and efficient algorithms, which would allow one to write high-speed and capable software that can be used in a real-time application.

KEY WORDS
Two-dimensional bin packing problem, rectangular items, heuristics, evolutionary algorithms, three-stage cutting problem.

1 Introduction

The two-dimensional bin-packing problem consists of placing a given set of smaller items into one or more larger objects (bins) without overlapping so as to minimize the waste of material and the total number of bins used. This problem has an important application in manufacturing such as wood, steel, glass and paper cutting. Depending on the particular application, two-dimensional packing problems can be divided into, at least, four different types: strip packing problems, bin packing problems, knapsack problems and cutting stock problems. However, it is not always possible to clearly separate one type of problem from another one, except, probably, the knapsack problem, which considers packing of a number of items associated with some value.

There is a number of algorithms available in the literature for the solution of such problems. Many of the simple algorithms are greedy, where items are placed one by one and never rearranged thereafter. A survey of such techniques can be found in [1, 2]. With single-phase algorithms, items are placed directly into the bins, whereas in two-phase algorithms the bin is first divided into strips, and every strip is then filled separately. A detailed description of the classical finite first fit heuristic for one-phase algorithms is given in [3]. An obvious approach to the solution of cutting or packing problem is linear programming [4, 5]. However, such a solution becomes too complex as the number of variables increases. Monaci and Toth [6] proposed a heuristic approach for bin packing problems. In a first phase, patterns are generated using a greedy and fast constructive heuristic; in a second phase the Lagrangian-based heuristic algorithm is used. A comprehensive overview and comparison is provided of a number of guillotine heuristics for the two-dimensional strip packing problem, in which rectangular items are placed without rotation and can be found in [7].

In this study we are dealing with the above-mentioned three packing methods in order to efficiently solve the real-world three-stage two dimensional cutting problem. The three-stage two-dimensional cutting problem with specific additional properties was addressed by Vanderbeck [8] and Puchinger et al [9]. Vanderbeck solves a three-stage two-dimensional cutting stock problem where the main objective is to minimize waste altogether with other issues such as urgency or optimality of the order and fixed set-up costs. Puchinger et al. consider a three-stage problem appearing in glass cutting. The problem is heuristically solved using an evolutionary algorithm based on order representation, specific recombination, mutation operators and a greedy decoding heuristic. A fast and efficient algorithm of the three-stage two-dimensional bin packing problem, based on a real-world problem, was proposed by Tabakov and Walker [10]. Various algorithms of strip packing problems,
also known as open dimension packing problems, are well presented in [11, 12]. More complex approaches, particularly, exact algorithms for the two-dimensional strip packing problem with and without rotations are developed by Kenmochi et al. in [13]. An interesting heuristic for selecting the next pieces to be placed when solving the 2D irregular bin packing problem is proposed by López-Camacho et al. in [14]. The literature on the subject of the bin packing problems is vast and the above references only are given because of their relevance to the subject.

Bin packing problems are known to be non-deterministic polynomial-time hard (NP-hard), and hence it is impossible to solve them exactly in polynomial time. Thus heuristics are very important to design practical algorithms for such problems. In this research we avoid the use of linear programming because we consider it to be a very cumbersome approach to analysing this type of problems and instead propose a simple and very efficient algorithm, which is a combination of the finite first fit heuristic algorithm in combination with other auxiliary techniques and evolutionary optimization, for a selection and a recombination of the items, using the evolutionary algorithms. Both infinite and finite length strips as well as larger rectangular bins are considered here. In simple cases, a sufficient, optimal or near-optimal, solution can be obtained without the use of the evolutionary optimization procedure.

2 Problem formulation

This section describes two-dimensional bin packing problems of a set of rectangular items of various dimensions, which must be placed without overlapping either onto an infinite strip or into a number of rectangular bins of larger size, in such a way as to minimize wastage of the used material.

2.1 Bin packing problem

Let $I = \{1, 2, \ldots, N\}$ be a set of $N$ rectangular items of width $w_i$ and height $h_i$, where $i \in I$. The objective is to place them into a minimum number of rectangular bins of a larger size having identical width $W$ and height $H$, and thus minimize wastage. The items may not overlap; however, they can be rotated by $90^\circ$ if rotation is allowed. Rotation of items might not be allowed if the items to be cut off are decorated with specific patterns (for example furniture manufacturing) or corrugated, whereas it is allowable in the case of plain material (e.g. glass or steel). For simplicity, we also ignore here the thickness of the blade or guillotine and assume that items can “touch” each other. This assumption does not influence the algorithm in any way, since the blade width can be simply incorporated into the dimensions of items. The location of each item is represented by a pair of the coordinates $\{(x_i, y_i)\mid i \in I\}$ of its lower left corner. Each set of the coordinates is unique and called a placement of $I$. The problem can be formally formulated in the following form:

\[
0 \leq x_i \leq W - w_i, \quad \forall i \in I \quad (1)
\]
\[
0 \leq y_i \leq H - h_i, \quad \forall i \in I \quad (2)
\]

The above constraints mean that none of the rectangular items can be larger in dimensions than the container (bin). Furthermore, at least one of the following four inequalities must be true for every pair of items $i$ and $j$:

\[
x_i + w_i \leq x_j \text{ or } x_j + w_j \leq x_i \quad (3)
\]
\[
y_i + h_i \leq y_j \text{ or } y_j + h_j \leq y_i \quad (4)
\]

These four constraints prevent overlap (i.e. each inequality signifies one of the four relative locations: left-of, right-of, above and below).

We consider the case when the cutting machine works in three stages. The first stage only cuts a whole sheet horizontally into an arbitrary number of strips. Strips are further processed in the second stage where they are cut vertically into a number of pieces, sometimes called stacks. The third stage performs, again, only horizontal cuts, producing the final elements from the stacks. It is obvious that the second and third stages might not be necessary, especially if the items are relatively large. Any feasible three-stage cutting pattern can always be reduced into its so-called normal form by moving each element to its uppermost and leftmost position, so that the waste may only appear at the top of the stacks and on the right of the rightmost stack in each strip. This approach reduces the general infinite search space of arbitrary three-stage cutting patterns to a finite number of possible solutions.

Figure 1: An example of a three-stage cutting pattern in normal form.

Besides, we found that it is much faster to start each strip with only one item (it must be the largest one in the set),
or in other words, the dimension of the largest item defines the width of the strips. Smaller items can be placed later. An example of this cutting pattern in normal form is given in Figure 1.

2.2 Strip packing problem

The strip packing problem can be considered as a generalization of the bin packing and, thus, its formulation is similar to the above. While the bin has fixed sizes, the strip does not have a fixed height, – it is either infinite or of variable height. Since we consider only guillotine cuts in this study, a level-by-level (level algorithms) approach seems to be most appropriate here. In this algorithm, the rectangular items fill the horizontal strips, which are termed levels to avoid confusion with the main strip, in a similar manner as they are placed in the vertical strips of the rectangular bin. Thus, the level algorithms make possible three-stage guillotine cuts. A modification of the level algorithms where the items are placed as they arrive, i.e. without pre-sorting, is called shelf algorithms. In this paper we do not discriminate strictly between the two terms. An example of the level layout is given in Figure 2. However, sometimes it is convenient to arrange the layers likewise the strips in the bin packing problem, so that they run perpendicular to the strip width.

\[ H \]

W

\[ H \]

W

Figure 2: An example of the level arrangement of rectangular items on a strip.

3 Three-stage packing algorithm

In the case of the bin packing problem a feasible solution of a two-dimensional three-stage packing problem implies that there is a number of identical bins (large rectangular sheets of material), which are divided into a set of strips, and each strip can consist of a set of stacks. In turn, each stack consists of a number of the final items. Contrary to the bin packing, the strip packing implies that the strip is of infinite length or its length is not defined explicitly.

3.1 Heuristic approach

Since we are going to use heuristic algorithms alongside with the evolutionary optimization in the design of a cutting pattern, it is essential to give a short explanation of them. The most popular algorithms are the next fit, first fit and best fit heuristic strategies, which originated from one-dimensional packing problems, particularly the log-cutting problem. First of all these algorithms require a proper initial arrangement of the design items (that is, it is necessary to construct a suitable permutation) and only after that they are placed one by one into the bin. One of the simplest approaches is called next fit. The idea behind this procedure is to open a bin and place the items into each level (when the strip is full, a new one is started), left justified if it fits. Otherwise, a new level is created and next item packed in it left justified. In first fit strategy we check whether or not an item fits from level one to the last level, and pack it left justified in the first level where it fits. If no level can accommodate it, then it is placed in a new level as in the next fit strategy. The best fit strategy is an extension of the first fit algorithm, with the difference that items are packed in the level that minimizes the unused horizontal or vertical space among those where it fits. If no level can accommodate it, a new level is opened as in the next fit strategy.

Next we propose the following combinatorial heuristic algorithm for the bin packing problem. A similar approach is used in the strip packing problem.

- The items are sorted in descending order according to their height \( h_1 \geq h_2 \geq \ldots \geq h_N \).
- If a number of items \( m \) have the same height \( h_i \), then this separate group is sorted in descending order according to their widths, so that \( h_i = h_{i+1} = \ldots = h_{i+(m-1)} \) and \( w_i \geq w_{i+1} \geq \ldots \geq w_{i+(m-1)} \).
- At this stage we consider every separate strip as a bin (to avoid confusion we use the term strips); however, remembering not to exceed the capacity of the primary bin (large rectangle sheet). We place the first item in the sorted list into the bin and the width of it will define the width of the first strip.
- We place the next item in the list into the first strip which has not been completely filled (numbered from left to right) into which it will fit. If rotation of items is allowed, they can be rotated if necessary. Both first and best fit strategies are employed. When strips are filled completely, they are closed and, if an item will not fit into any currently open strips, a new strip is opened.
- When all the items are already packed into the strips we need to rearrange the strips in such a way as to minimize the number of required bins as well as the waste of the material. For this purpose again the first fit heuristic strategy is used. The strips are sorted in descending order according to their widths and then
the bins are filled by strips. We employ the same strategy: keep bins open in the hope that we will be able to fill empty space with strips later. By doing so we will typically use fewer bins.

3.2 Evolutionary optimization

In many cases when the diversity of items to be cut off is not too large and their dimensions are consistent to some extent (e.g. furniture production) the first phase of the algorithm might be sufficient; that is optimal or very close to optimal, where further calculations are not justified. However, if the diversity in the dimensions is wide-ranging and the overall number of items is high, a near optimum solution can hardly be achieved. In order to get the best possible solution it is advisable to use one of meta-heuristic evolutionary methods. A great number of research publications is available on the subject of the application of the genetic algorithms for various bin packing problems. Nevertheless, a practical application of such algorithms is not always easy as they often have more theoretical than practical value. In fact, in many cases such problems can be solved in a much simpler way. Next we propose a couple of simple, fast and efficient algorithms which can be easily understood and implemented. The fitness function in all the cases is calculated as a ratio of the area of packed items to the area of the bin (strip). The objective of the optimization procedure is to maximize this ratio, making it approach one.

3.2.1 Strip packing with Genetic Algorithms

Taking into account a combinatorial nature of the problem the genetic algorithm (GA) [15] or micro-genetic ($\mu$GA) [16] algorithms are well suited for this purpose. Such evolutionary algorithms like the particle swarm optimization and the big bang – big crunch optimization often can overpower the genetic algorithms but, the advantage of the GA is that it does not require the coordinate information of the design parameters.

In this approach we pack the bins using the above heuristic algorithm and then try to improve the obtained solution, obviously not loosing it, by employing the $\mu$GA or GA:

- Each item $i = 1, 2, \ldots, N$ is assigned a number from 1 to $N$. The order is not significant here.
- With allowance made for the dimensions and number of items we determine a maximum possible number of items which can be contained in one strip ($s_{\text{max}}$). This number can be changed during calculations. Based on this, the number of bits in the chromosome is calculated as $\lambda = \frac{\ln(s_{\text{max}} + 1)}{\ln(2)}$, if binary coding is preferred.
- We analyse each strip separately, one by one. The initially built solution will be now among the rest of the parents in the initial population. It is likely to be the best in the population and the elitist strategy is absolutely essential to preserve it. Every time when the number of items in strips is less than the number of genes in the chromosome the difference is filled with zeros (no item). A variable length chromosome is another option; however it is a more complicated approach and was avoided in this research.

- First fit and best-fit decreasing height strategy is used for packing the strips.
- The two-point crossover works very well here, however, with the increase of the chromosome length the uniformly distributed crossover is probably a better choice for a combinatorial problem.
- It must be noted that some strips may change their width, and the total number of strips can be decreased as a result of the optimisation. In this case first fit heuristic is used for packing strips into the bins (one dimensional problem).
- If we feel that the strips can be arranged in a better way, the GA is the best tool for further improvement. In this case the length of the chromosome coincides with the number of strips. An initial arrangement is saved via elitism if a better solution is not found.

It should be stressed that with correct implementation the genetic algorithms do not require a large number of iterations and, in turn, due to its simplicity is very fast, which means the whole two-phase algorithm can be used in real time applications. With a large number of items to pack as well as a wide variety of different dimensions, it might be easier to achieve the best result by dividing the items into smaller sets.

The population in GA is formed of the strings large enough to pack one bin. It means that the area of the selected items must not be smaller than the total area of the bin. The chromosome is coded by binary strings representing the sequence number of each item and its quantity. Obviously, the string lengths can be different within one generation. Next the algorithm above is repeated.

The same approach is used for strip packing. In the case of level packing we should not worry about the widths and number of levels, since the length of the strip is infinite. However, if the guillotine cuts are not required, it might be feasible to consider the entire width of the strip and pack it from the bottom to the left and to the top [13]. The chromosome strings will be of the same length and form the order in which the items will be placed, the rotation of items can be allowed. Then the strip is packed layer by layer, similar to “Tetris” computer game. Despite the great number of possible combinations, the convergence of the genetic algorithm is fast and the final near optimum result can be achieved in just a few seconds.
3.2.2 Bin packing with Big Bang – Big Crunch algorithm

The BB–BC relies on one of the evolutionary theories of the universe, namely the big bang – big crunch theory. In the big bang phase the population of feature vectors randomly fills the space, while in the big crunch phase these points are drawn into a dense cluster with the centre of gravity being the optimum solution of the optimization problem. The algorithm is a heuristic population-based evolutionary optimization method. Among the merits of this method is computational simplicity, the ability to handle multidimensional problems and very fast convergence. The algorithm is very simple, does not require much tuning and can easily handle much higher numbers of dimensionality than the genetic algorithms.

In this approach we pack and close the entire bin, rather than the strip. Contrary to the above algorithm the fitness function utilizes an integer string of constant length. Although the BB–BC does not directly support the integer optimization, it can easily be modelled by rounding off the real parameters. Usually a set \( I \) of items consists of the number of different types (sizes) \( NT \), in turn, each type \( t_1, t_2, \ldots, t_{NT} \) may represent any number of identical items. Thus, the algorithm can be briefly described in the following manner:

- A randomly generated parameter string (similar to a chromosome string in GA) has a constant length \( NT \). Each parameter represents a number of selected items of a particular type, from zero to the maximum available quantity for this size.

- The total volume (area) of the selected items is calculated and, if required, the number of items is adjusted as follows: if the total volume is greater than the bin volume by more than the largest item in the string, then the largest item is removed. If necessary, this action is repeated. If the total volume is smaller than the bin volume, then the number of items in the string is increased from the common pool by adding the largest possible item and, again, if necessary, the action is repeated.

- Next, the heuristic approach is used. The only difference is that we must pack only one bin and close it (the next run of the algorithm will be dealing with another bin). The optimization algorithm rapidly finds the best possible combinations and the number of items to pack the bin in a most efficient way. The voids in the strips are remembered as the coordinates of two corners and all the items in the string checked if they can fill in any particular void space.

The BB–BC optimization algorithm is capable to isolate quickly an optimum solution volume (the area where the optimum solution can be found) in multidimensional functional space even from an absolutely immense number of choices. An amazing feature of the algorithm is that the contraction rate of the space is the same for any number of dimensions. The multidimensional problems have more dense working space and would require a bit higher numbers of generations before achieving the optimal or near-optimal solution.

4 Numerical results

Let us consider a simple realistic example of the bin packing problem. A list of items is given below in the Table 1 and they are to be cut out from rectangular bins of dimensions \( 260 \times 160 \) mm. For convenience the list is already sorted according to the requirements described above. The numbers on the items in the example correspond to those numbers in the table. The dimensions given in the table are in millimetres. There are 36 different types of items and the quantity of each type of item is ranging from one to ten.

<table>
<thead>
<tr>
<th>No.</th>
<th>H</th>
<th>W</th>
<th>Qty</th>
<th>No.</th>
<th>H</th>
<th>W</th>
<th>Qty</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>39</td>
<td>38</td>
<td>4</td>
<td>19</td>
<td>30</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>39</td>
<td>26</td>
<td>4</td>
<td>20</td>
<td>30</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>39</td>
<td>7</td>
<td>10</td>
<td>21</td>
<td>28</td>
<td>20</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>38</td>
<td>23</td>
<td>3</td>
<td>22</td>
<td>25</td>
<td>14</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>38</td>
<td>21</td>
<td>4</td>
<td>23</td>
<td>25</td>
<td>12</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>38</td>
<td>20</td>
<td>2</td>
<td>24</td>
<td>25</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>37</td>
<td>28</td>
<td>9</td>
<td>25</td>
<td>25</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>37</td>
<td>23</td>
<td>1</td>
<td>26</td>
<td>24</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>37</td>
<td>9</td>
<td>10</td>
<td>27</td>
<td>21</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>36</td>
<td>33</td>
<td>10</td>
<td>28</td>
<td>20</td>
<td>20</td>
<td>6</td>
</tr>
<tr>
<td>11</td>
<td>36</td>
<td>27</td>
<td>10</td>
<td>29</td>
<td>20</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>12</td>
<td>35</td>
<td>32</td>
<td>4</td>
<td>30</td>
<td>20</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>13</td>
<td>34</td>
<td>27</td>
<td>8</td>
<td>31</td>
<td>19</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>14</td>
<td>34</td>
<td>5</td>
<td>5</td>
<td>32</td>
<td>17</td>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>15</td>
<td>33</td>
<td>17</td>
<td>3</td>
<td>33</td>
<td>14</td>
<td>12</td>
<td>6</td>
</tr>
<tr>
<td>16</td>
<td>32</td>
<td>14</td>
<td>1</td>
<td>34</td>
<td>13</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>17</td>
<td>32</td>
<td>7</td>
<td>4</td>
<td>35</td>
<td>11</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>18</td>
<td>30</td>
<td>13</td>
<td>7</td>
<td>36</td>
<td>8</td>
<td>7</td>
<td>9</td>
</tr>
</tbody>
</table>

Table 1: A set of 36 different rectangular items for packing into a \( 260 \times 160 \) mm rectangular bin. The dimensions of items are given in millimetres.

The Figures 3, 4 and 5 show the final optimal solution found using the proposed algorithm for the three-stage two-dimensional bin (strip) packing algorithm. The numbers on the items correspond to their sequence number in the Table 1. The thickness of the cutting blade is ignored in the example. The items first were placed using the heuristic first fit and best-fit decreasing height algorithm. The items were placed into the bins in the order of appearance in the list. The longest items were placed first and, where possible, grouped. It should be noted that there is a fundamental difference between the known examples in the literature (see for example [1]) and the proposed approach. In these models the width of the strips is determined using an optimisation technique, thus it is not fixed from the beginning.
This means that a strip can be formed from a number of items spread across it starting from the bottom of the bin. After a number of numerical tests we found that the packing algorithm is much faster and efficient when the strip width is determined only by one item placed at the bottom.

The first stage of packing was done using only the heuristic algorithm and the overall efficiency of the packing achieved is 79.5%, which is already a relatively good result. The efficiency is calculated the same way as the fitness function, as a ratio of the total area of the packed items divided by the total area used in the container. In simple cases the heuristic algorithms usually result in optimum or near optimum solution. However, it is unlikely in more complex problems, such as the one considered here. Then, further improvement is desirable. The genetic algorithm was used for this purpose in this example. The chromosome string contains only the information about the sequence of the items and their numbers. In our case it was possible to code the whole set of the parameters into one string. Otherwise, some fixed amount of items or bins is used at one time. The application of the genetic algorithm has resulted in fast improvement of the packing efficiency up to 92.8%, which can be considered as a very good result, and seems to be a near optimal one, taking into account a complexity of the problem. The results obtained by the big bang – big crunch optimization are similar to those obtained here.

Figure 3: Optimal placement of items in the first bin.

Figure 4: Optimal placement of items in the second bin.

Figure 5: Optimal placement of items in the third bin.
5 Concluding remarks

In this paper, we propose two practical algorithms for the two-dimensional rectangle bin-packing problem, which has many industrial applications. Three-stage cutting technology is considered. These algorithms are suitable for both the rectangular bins and long strips. In this research, we avoided the use of linear programming because we consider it to be a very cumbersome approach to treating this class of problems and instead implement two simple and very efficient algorithms. Firstly, we use the GA wherein the first phase the bins are packed by means of the finite first fit heuristic algorithm and with the help of other auxiliary techniques. The purpose of the second phase is to improve the initial arrangements by performing combinatorial optimization for either a limited number of bins or the whole set at one time without destroying the original pattern (elitist strategy). In the second algorithm we select the subset of items to pack a single bin by using the BB–BC optimization. Such an approach does not require much memory and is very fast.

These algorithms allow us to write high-speed and capable software, which can be used in real-time applications. Numerical results obtained by optimizing existing industrial problems demonstrated that in many cases it was possible to achieve the optimum solution within only a few seconds, whereas for large-scale complex problems the result was near optimum (efficiency over 90%) within the same period of time. The distinctive property of the algorithm is that the width of the strips is determined by only one item, large enough for this purpose.

In addition, after some modification, the proposed algorithm can easily be implemented for three-dimensional bin packing problems (for example packing of containers of different dimensions).

Acknowledgements

The authors would like to thank the Durban University of Technology and, particularly, the Department of Research and Postgraduate Support for their invaluable assistance and support that made this research work possible.

References


