Hybrid genetic approach for 1-D bin packing problem

Roman Potarusov
Department of CAD,
Taganrog Institute of Technology,
Southern Federal University,
Taganrog, Russia
Email: potarusov.roman@gmail.com

Hamid Allaoui* and Gilles Goncalves
Faculté des Sciences Appliquées,
Université d’Artois,
Technoparc Futura, 62400 Béthune, France
Email: hamid.allaoui@univ-artois.fr
Email: gilles.goncalves@univ-artois.fr
*Corresponding author

Victor Kureychik
Department of CAD,
Taganrog Institute of Technology,
Southern Federal University,
Taganrog, Russia
Email: kur@tsure.ru

Abstract: This paper deals with the one-dimensional Bin Packing Problem (1-D BPP). Exact solution methods can only be used for very small instances, hence for real-world problems we have to focus on heuristic methods. In recent years, researchers have started to apply evolutionary approaches to this problem, including Genetic Algorithms and Evolutionary Programming. In this paper, we propose a Hybrid Genetic Algorithm (HGA) to solve 1-D BPP. We compare our approach with algorithms given by Scholl et al., Alvim et al. and Kok-Hua et al. Then we discuss the performance of the approach. We show that giving at least the same performance on term of quality solution, our HGA approach outperforms these algorithms on term of computational time. This performance is due to new mechanisms of hybridisation of genetic algorithms and local search.

Keywords: bin packing; genetic algorithms; hybrid approach; services operations.


Biographical notes: Roman Potarusov is Research Engineer in the CEA, the French Atomic Energy and Alternative Energies Commission. He obtained his PhD degree in Informatics and Automatics in 2008 in the University of Artois (France) and Southern Federal University (Rostov-on-Don, Russia). In 2009, he worked for the University of Artois as a post-doc. His current research interests are evolutionary algorithms, packing and routing optimisation problems.
Hamid Allaoui is currently Associate Professor (Maître de Conférences) at University of Artois in France. He holds a PhD degree in Computer Science and Management Science in 2004. His current research covers design, management and optimisation of supply chains especially scheduling and planning of operations. He has published in several international journals like International Journal of Production Economics and Journal of Computers and Industrial Engineering. He is involved in research projects as well as projects with industrial companies.

Gilles Goncalves is a Full Professor at the University of Artois, since 1994. He studied Computer Science at the University of Lille1 and obtained his PhD in 1985. During 14 years, he was lecturer and his works concerned parallel and distributed computing. His current research interests include information systems, security management, simulation modelling and discrete optimisation. He co-manages a research team named ‘LG2A’ whose main topics are organisation and management of logistic processes.

Victor Kureychik, Doctor of Science, Professor, is Vice-President of Taganrog Institute of Technology of Southern Federal University, Russian Federation. He obtained Doctor of Science degree in Informatics and Computer Engineering in 1979. During 15 years, he was lecturer and his works concerned artificial intelligence, computer-aided design, genetic algorithms, bio-inspired systems. His current research interests are evolution simulation, optimisation problems, artificial intelligence, computer-aided design, genetic algorithms, bio-inspired systems.

1 Introduction

Packing of items into boxes or bins is a recurring task in distribution and production activities. Concerning the size and shape of items as well as the form and capacity of bins, a large variety of different packing problems can be distinguished (Scholl et al., 1997). Similar problems concern the cutting of pieces into particular smaller ones so as to minimise the wastage of material and scheduling of identical parallel processors so as to minimise the total completion time. There are many other industrial problems that seem to be different, but have a very similar structure, such as capital budgeting, processor scheduling and VLSI design (Bischoff and Wäscher, 1995). 1-D BPP models several practical problems in computer science and industrial engineering, such as table formatting, pre-paging, file allocation, etc. (Johnson et al., 1974).

We consider a basic packing problem which is known as the one-dimensional bin packing problem (1-D BPP). This problem is well known in combinatorial optimisation theory. 1-D BPP aims to combine items with different weights into bins of the same capacity so as to minimise the total number of bins. In its general form, 1-D BPP is NP-hard in the strong sense (Garey and Johnson, 1979). Hence, there is a little hope of finding even pseudo-polynomial time optimisation algorithm to solve it. When we are confronted with the fact that the problem is NP-hard and the optimal solution is unattainable in reasonable time by exact methods, which is the case of our problem, it is reasonable to sacrifice optimality and settle for a ‘good’ feasible solution that can be computed efficiently. Of course, we would like to sacrifice as little optimality as possible, while gaining as much as possible in efficiency.
In this paper, we propose to use genetic algorithms (GA) as a global approximation method. Nowadays, the most challenging problems are too complex to be undertaken using classical optimisation techniques. In fact, there is an increasing need to develop innovative approaches by focusing on several schemes of combining optimisation methods. GAs are global optimisation methods and able to locate the right solution in large-scale response surfaces including many local optimaums. However, when the search space is too large, these algorithms have inherent difficulties to converge in the global optimum with an adequate precision. Local search techniques follow a different strategy, being able to find any local optimum with great precision, using information from the neighbouring candidate solutions. Advantages and drawbacks of GAs and local optimisation methods are reciprocal. The synergy between both methods can therefore give rise to a family of hybrid algorithms, simultaneously global and precise. The straightforward implementation of this idea consists of applying a genetic search and a local technique in two consecutive steps. More specifically, the GA explores the domain and finds a good set of initial estimates, which is further refined by the local technique in a second step, in order to locate the nearest, best solution (Vivo-Truyols et al., 2001).

This paper describes a new Hybrid Genetic Approach (HGA) to solve 1-D BPP which is aimed to provide approximation and optimal solutions for all benchmarks instances in an acceptable computational time. The rest of the paper is organised as fellows: Section 2 presents the problem formulation. Section 3 reviews existing works and results dealing with 1-D BPP problem. Section 4 discusses the proposed hybrid approach. Section 5 shows some of the computational results and analysis. Section 6 concludes the paper with some future research directions.

2 Problem statement

1-D BPP can be described as follows (Martello and Toth, 1990). Given \( n \) items and \( N \) bins, \( w_j \) is the weight of item \( j \) and \( c \) is the capacity of each bin, assign each item to one bin so that the total weight of the items in each bin does not exceed \( c \) and the number of bins used is a minimum. A possible mathematical formulation of the problem is presented by Martello and Toth (1990):

\[
\begin{align*}
\text{minimise} \quad & z = \sum_{i=1}^{N} y_i \\
\text{subject to} \quad & \sum_{j=1}^{n} w_j x_{ij} \leq c y_i, i \in \{1, \ldots, N\}, \\
& \sum_{j=1}^{n} x_{ij} = 1, j \in N \\
& y_i = 0 \text{ or } 1, i \in N, \\
& x_{ij} = 0 \text{ or } 1, i \in N, j \in N, \\
& y_i = \begin{cases} 
1, & \text{if bin } i \text{ is used} \\
0, & \text{otherwise}
\end{cases}, \\
& x_{ij} = \begin{cases} 
1, & \text{if item } j \text{ is assigned to bin } i \\
0, & \text{otherwise}
\end{cases}.
\end{align*}
\]

Hybrid genetic approach for 1-D bin packing problem
We assume that $w_j, c$ are positive integers, $w_j \leq c$ for $j \in N$. Falkenauer (1996) proposed the following objective function:

$$\text{maximise } f_{\text{BPP}} = \frac{\sum_{i=1}^{N} (S_i/c)^k}{N},$$

where $N$ is the number of bins used in the solution, $S_i$ is the sum of weights of the items in the bin $i$ (the fill of the bin), $c$ is the bin capacity and $k$ is a constant, $k = 2$. The constant $k$ expresses the concentration on the ‘extremist’ bins in comparison to the less filled ones. The larger $k$ is, the more we prefer well-filled ‘elite’ groups as opposed to a collection of about equally filled bins. In fact, the value of $k$ gives us the possibility to vary the ‘ruggeness’ of the function to optimise, from the ‘Needle-in-a-haystack’ ($k = 1, f_{\text{BPP}} = 1/N$) up to the ‘Best-filled Bin’ ($k \rightarrow \infty, f_{\text{BPP}} > \max_i [(F_i/c)^k]$). Falkenauer (1996) experimented several values of $k$ and found out that $k = 2$ gives good results. Larger values of $k$ seem to lead to premature convergence of the algorithm, as the local optima, due to few well-filled bins, are too hard to escape. For our HGA a slightly modified objective function is used to evaluate an entire BPP solution and to get the information about full bins:

$$\text{maximise } f_{\text{BPP}} = \frac{\sum_{i=1}^{Q} (S_i/c)^k + \sum_{j=1}^{F} (S_j/c)}{N},$$

where $N$ is the number of bins used in the solution, $Q$ is the number of bins, which have some free space, $S_i$ is the sum of item weights in bin $i$, $c$ is bin capacity, $F$ is the number of full bins, $S_j$ is sum of weights of the items in bin $j$, $k$ is the same constant, $k = 2$. For full bins, $S_j/c$ equals to 1. Therefore, modified objective function can be written as follows:

$$f_{\text{BPP}} = \frac{\sum_{i=1}^{Q} (S_i/c)^k + F}{N}.$$
variable generation and branch-and-bound. At each iteration, the sub-problem generates a set of columns, which altogether correspond to an attractive valid packing for a single bin. Heuristic based approaches have also been used. Some well-known constructive heuristics, such as First-Fit Decreasing (FFD) and Best-Fit Decreasing (BFD) (Martello and Toth, 1990) used no more than $11/9 \text{OPT} + 1$ bins, where OPT is the number of bins given by the optimal solution.

Meta-heuristic approaches have also been applied to 1-D BPP. Falkenauer (1996) introduced a new hybrid grouping genetic algorithm for the problem (HGGA). The algorithm is hybridised with a local search procedure and tested on a set of benchmark problems. HGGA algorithm is shown to be superior to MTP. Levine and Ducatelle (2003) proposed an Ant Colony Optimisation (ACO) approach to 1-D BPP. Apart from a pure ACO approach, they also developed a hybrid ACO algorithm. This approach combines the ACO meta-heuristic with a simple but effective iterated local search algorithm based on the Dominance Criterion of Martello and Toth (1990). The hybrid ACO approach is much more efficient for some problem instances. Fleszar and Hindi (2002) combined a minimal bin slack heuristic, proposed by Gupta and Ho (1999), with a variable neighbourhood search method. The algorithm obtains better results by achieving 1329 optimal solutions from 1370 benchmark problems. Moreover, Bai (2005) proposed a simulated annealing hyper heuristic and applied it to solve 1-D BPP. Alvim et al. (2004) developed a hybrid improvement procedure for 1-D BPP. The procedure improved the best known solutions for many of benchmark instances and found the largest number of optimal solutions with respect to the other available approximate algorithms. Kok-Hua et al. (2008) developed a powerful new procedure (WABP) that implements the concept of weight annealing to solve 1-D BPP problem. When applied to 1587 benchmark instances, WABP found the existing best known or optimal solutions to 1584 instances and generated new optimal solutions to the remaining three instances.

In this paper, we compare our HGA with the best existing algorithms in the literature to solve the 1-D BPP namely Scholl et al. (1997), Alvim et al. (2004) and Kok-Hua et al. (2008). We show by a deep experimental study that our approach gives at least the same performances on term of quality solution but it performs them on term of computational time. This is due to new mechanisms of hybridisation of genetic algorithms and local search taken into consideration.

4 Hybrid genetic approach

In this paper, we propose to combine GA and two replacement procedures RP1 and RP2. The main objective of these two procedures is to obtain feasible 1-D BPP solutions and improve them. Improved solutions are determined by steepest descent algorithms RP1 and RP2 inspired respectively from Fleszar and Hindi (2002) and Falkenauer (1996). This section is devoted to present the HGA.

4.1 Solution encoding

For HGA, we use the same encoding as in the Falkenauer’s HGGA (Falkenauer, 1996, 1998) genes encode whole bins with items. To understand the encoding, let us consider the following example. Let $A, B, C, D, E, F$ be different bins. Items to pack have the following weights 68, 70, 42, 59, 79, 25, 1, 35, 32. Capacity of each bin is 100. Insert those items into the bins by random, then the solution (chromosome) looks as follows
ABCDEF : ADCEFBBBA. Notation ABCDEF : ADCEFBBBA means that nine items are packed into six bins as follows: the 1st item (68) is packed into bin A, the 2nd item into bin D, the 3rd item into bin C, the 4th item into bin E, the 5th item into bin F, the 6th, the 7th, the 8th into bin B and the 9th item into bin A. Falkenauer (1996, 1998) called the left part of this notation as the ‘grouping part’ and the right part of the notation as the ‘element part’.

It is clear that the grouping part, which contains only full bins, plays important role in the solution process. We call it as ‘functional part’. Our genetic operators work with grouping part and functional part of the chromosome. As shown in Figure 1, the functional part consists of bin A only. The number of bins in functional part of the solution must not be decreased while the genetic operators are carried out.

**Figure 1** Solution encoding

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>(68,32)</td>
<td>(25,35,1)</td>
<td>(42)</td>
<td>(70)</td>
<td>(59)</td>
<td>(79)</td>
</tr>
</tbody>
</table>

4.2 **HGA: overall procedure**

The overall solution procedure can be formally summarised as follows:

Step 1. Generate the initial population of POPSIZE solutions (chromosomes)
Step 2. Divide the initial population into two equal parts (subpopulations) do in parallel (for each of two subpopulations)

{  
1) **Evaluate** chromosomes according to the objective function;

2) **Apply** the corresponding **crossover operator** (Dichotomy crossover – for subpopulation 1, Golden Section crossover – for subpopulation 2) to chromosomes in the subpopulation at a predefined probability $P_C$;

3) **Apply Directed Mutation operator** to random selected chromosomes with probability $P_M$;

4) **Apply Transposition operator** to the worst chromosomes with probability $P_T$;

5) **Reduction** of new subpopulation. We use $\mu + \lambda$ evolution strategy to create the next generation (Schwefel, 1995). In case of subpopulation 1, candidate solutions with objective function (OF) value more than 0.5 are selected to be a part of the next HGA generation, in case of subpopulation 2 solutions with fulfil bins are passed to new generation;

6) Perform the **migration** every $M$ generations according to the migration scheme, described below.

}

while (the **HGA termination criterion is not obtained**.)

Step 3. End
HGA is terminated when one of the following conditions is satisfied:

1. A total of $MaxGen$ generations have been processed.
2. Theoretical number of bins used in the solution is reached. It is defined by lower bound $LB = \lceil \text{total weight of items to pack/bin capacity} \rceil$ (Martello and Toth, 1990).

The flowchart of our HGA for the 1-D BPP is shown in Figure 2.

**Figure 2** The flowchart of HGA

4.3 **Initial population generation**

In this paper, an initial population of 1-D BPP solutions is created using First-Fit (FF) heuristic (Martello and Toth, 1990), as follows: given a list of items to pack, items from the list are taken randomly and inserted to the solution using FF.
4.4 Migration policy

Two sub-populations are connected to each other by a mechanism of migration. In our case, the following migration policy is proposed:

1. The number of chromosomes (candidate solutions), migrating from one sub-population to the other, is calculated using the equation:

\[ N_{\text{Mig}} = C_{\text{Mig}} \times \text{SUBPOPSIZE}, \]

where \( \text{SUBPOPSIZE} \) – size of sub-population in HGA, \( C_{\text{Mig}} \) – migration rate.

2. Selection of \( \frac{N_{\text{Mig}}}{2} \) of the best and \( \frac{N_{\text{Mig}}}{2} \) of the worst candidate solutions in two sub-populations is carried out;

3. Swapping selected solutions between sub-populations. \( N_{\text{Mig}} \) solutions in one sub-population replace the same number of selected solutions in the other sub-population.

The proposed migration policy aims to extend the search space for HGA and avoid early algorithm convergence to local optimum (Cantú-Paz, 1997).

4.5 Selection of parent chromosomes

Before the application of the crossover operator, a pair of parent chromosomes must be selected. For HGA, this is done using roulette wheel selection (in case of Dichotomy Crossover) and tournament selection (in case of Golden Section Crossover). We require that parent one and parent two be different chromosomes. Crossover begins, once two unique parents are chosen.

4.6 Dichotomy Crossover operator

The following steps detail the used Dichotomy Crossover operator:

1. Given two parent chromosomes (candidate BPP solutions) (see Figure 3a). Halve each chromosome. Cut a point in the middle of the chromosome. If the length of chromosome is odd, then take the nearest number of genes to any part (half).

2. Follow the rules of one-point crossover to get two offsprings (new 1-D BPP solutions) (see Figure 3b).

3. If both offsprings are feasible solutions, go to the step 5, else:
   - items, not found in offspring, are placed in list \( Z \) of items not assigned to bins;
   - items, found in offspring twice, are deleted together with respective bins. Items, that have not any copies, but located in deleted bins, are also placed into \( Z \) (see Figure 3c).

4. Use replacement procedure RP1 described below to reinsert items from \( Z \) into the solution (chromosome) (see Figure 3d). First iteration of crossover is finished.

5. At the second iteration of crossover the obtained offsprings become new parents. Divide each half of parent chromosomes into two equal parts and do the steps 3 and 4, following the rules of three-point crossover.
Figure 3  Dichotomy Crossover operator

<table>
<thead>
<tr>
<th></th>
<th>(68)</th>
<th>(1,35,25)</th>
<th>(42)</th>
<th>(70)</th>
<th>(59)</th>
<th>(79)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>59</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>79</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>25,68</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>70</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>g</td>
<td>42</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(a) Step 1

<table>
<thead>
<tr>
<th></th>
<th>(68)</th>
<th>(1,35,25)</th>
<th>(42)</th>
<th>(70)</th>
<th>(35)</th>
<th>(42)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>g</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b) Step 2 (first offspring)

<table>
<thead>
<tr>
<th></th>
<th>(68)</th>
<th>(70)</th>
<th>(35)</th>
<th>(42)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>g</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(c) Step 3 (first offspring)

(1)  | (68) | (79) | (35) | (42) |
| A |      |      |      |      |
| e |      |      |      |      |
| f |      |      |      |      |
| g |      |      |      |      |

Z = {79, 59, 25, 1}

(2)  | (70) | (79) | (35) | (42) |
| A |      |      |      |      |
| e |      |      |      |      |
| f |      |      |      |      |
| g |      |      |      |      |

Z = {79, 59, 25, 1}

(3)  | (70) | (79) | (59) | (68) |
| A |      |      |      |      |
| e |      |      |      |      |
| f |      |      |      |      |
| g |      |      |      |      |

Z = {68, 59, 25, 1}

(4)  | (70,25,1) | (79) | (59,35) | (68) | (42) |
| A |      |      |          |      |      |
| e |      |      |          |      |      |
| f |      |      |          |      |      |
| g |      |      |          |      |      |
| X |      |      |          |      |      |

Z = {42, 35, 25, 1}

(d) Step 4 (first offspring)

4.7  **Golden Section Crossover operator**

The used Golden Section Crossover operator is as follows:

1. Given two parent chromosomes, candidate 1-D BPP solutions. Cut points are defined by the nearest integer value 0.618*L from both ends, where L – the length of a chromosome (see Figure 4a).

2. Follow the rules of two-point crossover to get two solutions (offsprings) (see Figure 4b).

3. If both offsprings are feasible solutions, go to the step 5. Else:
   - items, not found in offspring, are placed in list Z of items not assigned to bins;
   - items, found in offspring twice, are deleted together with respective bins. Items, that have not copies, but located in deleted bins, are also placed into Z (see Figure 4c).

4. Use replacement procedure RP2 described below to reinsert items from Z into the solution (chromosome) (see Figure 4d).
Figure 4  Golden Section Crossover operator

(a) Step 1

(b) Step 2 (first offspring)

(c) Step 3 (first offspring)

(d) Step 4 (first offspring)

4.8 Replacement procedures

The main goal of the two replacement procedures respectively RP1 and RP2 is to obtain feasible 1-D BPP solutions and improve them. At each step, the algorithms perform the replacement which minimizes the bin’s slack (fits the bin capacity as much as possible). Time complexity of RP1 and RP2 is $O(ka)$. We present respectively RP1 and RP2 as follows:

**RP1.** Given child chromosome, partial invalid solution. Given a list $Z$ of items not assigned to bins. The list is sorted in the non-increasing order of their weights. First item from the list is set as a current item.

1. Bins in the solution are considered one by one. The value of replacement $\Delta f$ is calculated for all potentially possible replacements of items from considered bin by current item from $Z$ using following equation:

$$\Delta f = (w_j)^2 + [l(\beta) + w_i - w_j]^2 - (w_i)^2 - (l(\beta))^2,$$

where $w_i$ – weight of item $i$ in $Z$, $w_j$ – weight of item $j$ in considered bin $\beta$, $l(\beta)$ – total weight of items in bin $\beta$.

Replacements involving items from full bins do not increase the objective function, therefore they are excluded from consideration. Go to the next bin.
Hybrid genetic approach for 1-D bin packing problem

2 Replacement with maximal value of $\Delta f$ is carried out.
3 Item from list $Z$ is packed into the current bin, corresponding item with smaller weight from the current bin is packed into the list $Z$. Go to the next item in $Z$.
4 When replacements are not possible, FFD heuristic is used for inserting items from $Z$ into the solution.

**RP2.** Given a child chromosome, partial invalid solution. Given a list $Z$ of items not assigned to bins. The list is sorted in the non-increasing order of their weights.

1 Items from $Z$ are considered one by one. First bin is set as a current bin.
2 If the weight of item with minimal weight in the current bin is less than the weight of considered item from the list $Z$, then the corresponding replacement is carried out in such a way that the total content of the bin is increased without exceeding the bin capacity. Otherwise, the next bin is set as a current bin.
3 If the current bin is the final bin in the solution, the next item in the list $Z$ is considered.
4 When replacements are not possible, FFD heuristic is used for inserting items from $Z$ into the solution.

4.9 Directed mutation operator

Mutation operation alters each child at a pre-defined probability. The operator can help the HGA to reach further solutions in the search space. The idea of the mutation operation is to randomly or directly mutate the number of bins and hence produce a new solution that is not very far from the original one. The step by step presentation of the proposed mutation operator is as follows. Given a chromosome (candidate 1-D BPP solution), init a constant $d_{\text{Mutation}}$ by any real number from the range $[0.01, 0.99]$.

1 Calculate the partial objective function value $pf_{\text{BPP}}$ for every bin (gene) in the chromosome using the following equation:

$$ pf_{\text{BPP}} = \frac{\sum_{i=1}^{K} w_i}{c}, $$

where $K$ – the number of items in considered bin,
$w_i$ – the weight of item $i$,
$c$ – capacity of each bin.

2 If there are no any bins in chromosome with a partial objective function value less than $d_{\text{Mutation}}$, go to step 4. Else:
   - delete those bins from the chromosome;
   - items, located in the deleted bins, are placed in the list of free items.
3 Use First-Fit heuristic to reinsert items from the list into the solution (chromosome).
4.10 Transposition operator

Transposition is a process, in which some parts of the chromosome change their positions. In HGA, the proposed transposition operator is applied to the functional part of the chromosome. Consider the operator step by step. Given a chromosome (Figure 5).

1. Find all the full bins in the chromosome (bins A and E);
2. Place them into the beginning of the chromosome. The result is shown in Figure 6.

The operator is aimed to avoid the changing of the number of fulfil bins by collecting them in the beginning of the solution and thus creating a right bound of the functional part. At the next step, the genetic operators would not consider that (functional) part of chromosome for permutations.

Figure 5  The initial chromosome

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>{68,32}</td>
<td>{42}</td>
<td>{70}</td>
<td>{59}</td>
<td>{64,35,1}</td>
<td>{79}</td>
</tr>
</tbody>
</table>

Figure 6  The resulting solution

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>E</th>
<th>C</th>
<th>D</th>
<th>B</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>{68,32}</td>
<td>{64,35,1}</td>
<td>{70}</td>
<td>{59}</td>
<td>{42}</td>
<td>{79}</td>
</tr>
</tbody>
</table>

5 Experimental results and analysis

The experimental results were realised using a computer with CPU PM 1.7 GHz, memory 512 Mb, Microsoft Visual Studio and NET as software development environment, are used. Two classes of benchmark bin packing problem instances were used. They are available in the OR Library (http://www.ms.ic.ac.uk/info.html) (Beasley, 1990).

The first class (U class) consists of items of weights uniformly distributed in (20, 100) to be packed into bins of capacity 150. There are four sets in this class, namely U120, U250, U500 and U1000. Each consists of 20 instances with 120, 250, 500 and 1000 items, respectively. The second class (T class) consists of ‘triplets’ of items from (25, 50) to be packed into bins of capacity 100. There are also four sets in this class, namely T60, T120, T249 and T501. Each set contains 20 instances with 60, 120, 249 and 501 items, respectively.

All problem instances in both classes U and T have been solved to optimality with an exact algorithm by Valerio de Carvalho (1999). For the T class, all the instances are designed such that in the optimal solution each bin is filled up with three items. Thus the number of bins in the globally optimal solution for each instance is equal to the number of items divided by 3. The difficulty of the T class is caused by the lack of any slack in the optimal solution. A third class of benchmark bin packing problem instances (H class) contains three sets, which for the purposes of this article are called H1, H2 and H3. The H1 set consists of 720 instances, the H2 set consists of 480 instances. In both H1 and H2 sets, the number of items varies from 50 to 500. The H3 set consists of ten instances,
Hybrid genetic approach for 1-D bin packing problem

each of 200 items of a weight uniformly distributed in (20,000, 35,000) to be packed into bins of capacity 100,000. This set is considered to be very hard. Detailed description of all parameters for the three sets of the H class can be found in (Scholl et al., 1997).

The parameters used by HGA are as follows: number of generations – 70–100, population size –100, migration ratio – 0.15–0.25, migration performed every 5–10 iterations of GA, crossover probability – 0.8–0.9, mutation probability – 0.1–0.15, transposition probability – 0.25. Under those parameters, the best convergence of HGA is obtained.

For evaluating and comparing methods, we use the following definitions of the relative deviation and the absolute deviation of a value x from a value y, where x and y are values of lower or upper bounds: abs = |x−y| and rel = |x−y|*100%/y. All results are compared to the reference solutions, which are the optimal solutions or the best-known lower bounds where the optimum is not known. Table 1 presents the results of HGA. N indicates the number of instances in each test set. The next column shows the absolute deviation (abs) from the reference solution. The absolute deviation is represented by the number of added bins. For both, the average (av.) and the maximum (max.) values over all members of each set are displayed.

<table>
<thead>
<tr>
<th>Class</th>
<th>Instances</th>
<th>Opt.</th>
<th>abs.</th>
<th>rel. (%)</th>
<th>Time</th>
<th>Opt.</th>
<th>abs.</th>
<th>rel. (%)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>U120</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>20</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>U250</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>20</td>
<td>0</td>
<td>0.12</td>
</tr>
<tr>
<td>U500</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>20</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>U1000</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.01</td>
<td>20</td>
<td>0</td>
<td>0.01</td>
</tr>
<tr>
<td>T60</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0.15</td>
<td>0.85</td>
<td>20</td>
<td>0</td>
<td>0.37</td>
</tr>
<tr>
<td>T120</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0.45</td>
<td>2.20</td>
<td>20</td>
<td>0</td>
<td>0.85</td>
</tr>
<tr>
<td>T249</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0.12</td>
<td>0.31</td>
<td>20</td>
<td>0</td>
<td>0.22</td>
</tr>
<tr>
<td>T501</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>1.29</td>
<td>9.51</td>
<td>20</td>
<td>0</td>
<td>2.49</td>
</tr>
<tr>
<td>H1</td>
<td>720</td>
<td>720</td>
<td>0</td>
<td>0</td>
<td>0.09</td>
<td>9.43</td>
<td>720</td>
<td>0</td>
<td>0.19</td>
</tr>
<tr>
<td>H2</td>
<td>480</td>
<td>480</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.38</td>
<td>480</td>
<td>0</td>
<td>0.01</td>
</tr>
<tr>
<td>H3</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>2.31</td>
<td>23.38</td>
<td>10</td>
<td>0</td>
<td>4.60</td>
</tr>
</tbody>
</table>

We can see from the following tables that our approach gives for all classes of instances the best solution (reference solution). We now compare the results obtained by the HGA with those obtained by other approaches reported in the literature. We first report in Table 1 the results obtained by HI_BP (Alvim et al., 2004) and Perturbation MBS’ + VNS (Fleszar and Hindi, 2002). For each class, we give the number of instances and, for each heuristic, the number of instances for which the optimal solution is found, the maximum absolute deviation, the maximum relative deviation, and the average and maximum computation times in seconds. The results reported for Perturbation MBS’ + VNS were obtained on a 400 MHz Pentium and extracted from Table 8 of Fleszar and Hindi (2002). Similar statistics are reported in Table 2, concerning the comparison...
between HGA and WABP (Kok-Hua et al., 2008), and in Table 3, concerning the comparison between HGA and BISON (Scholl et al., 1997). The results for WABP were obtained on a 3 GHz Pentium 4 computer with 256MB of RAM. The results obtained for BISON, with the processing time limited to 1000 seconds, were obtained on a PC 80486 DX2-66 and extracted from Tables 4–6 of Scholl et al. (1997).

### Table 2  
HGA vs WABP (Kok-Hua et al., 2008)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>U120</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>U250</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>U500</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>U1000</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>0.00</td>
<td>0.01</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>T60</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>0.15</td>
<td>0.85</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>T120</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>0.45</td>
<td>2.20</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>T249</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>0.12</td>
<td>0.31</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>T501</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>1.29</td>
<td>9.51</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>H1</td>
<td>720</td>
<td>720</td>
<td>0</td>
<td>0.09</td>
<td>9.43</td>
<td>697</td>
<td>2</td>
</tr>
<tr>
<td>H2</td>
<td>480</td>
<td>480</td>
<td>0</td>
<td>0.00</td>
<td>0.38</td>
<td>480</td>
<td>0</td>
</tr>
<tr>
<td>H3</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>2.31</td>
<td>23.38</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>

Total 1370

### Table 3  
HGA vs. BISON (Scholl et al., 1997)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>H1</td>
<td>720</td>
<td>720</td>
<td>0</td>
<td>0.09</td>
<td>9.43</td>
<td>697</td>
<td>2</td>
</tr>
<tr>
<td>H2</td>
<td>480</td>
<td>480</td>
<td>0</td>
<td>0.00</td>
<td>0.38</td>
<td>473</td>
<td>1</td>
</tr>
<tr>
<td>H3</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>2.31</td>
<td>23.38</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

Total 1210

In conclusion, HGA outperforms BISON either on term of quality solution or computational time. Moreover, for the most difficult class of instances in 1-D BPP, H1, H2, H3, our approach improves significantly computational time comparing to BISON. Concerning HI_BP, our approach outperforms it only on term of computational time since it gives always, for the set of all instances, deviation equal to zero from the reference solution. WABP outperforms our approach, only for 5 under 11 classes of instances, on term of computational time. For the remaining classes, HGA has at least the same computational time.

### 6 Conclusion and perspectives

In this paper, we presented a Hybrid Genetic Approach (HGA) to solve the one-Dimensional Bin Packing problem (1-D BPP). New specific genetic operators are developed with two effective replacement procedures. The realised computational
Hybrid genetic approach for 1-D bin packing problem

Experiments establish that the presented HGA never decrease the quality of existing solutions in the literature. Moreover, HGV gives approximation solutions and optimal solutions for all benchmark instances and outperforms existing approaches on term of computational time. Efficiency of HGA is due to new hybridisation mechanisms of genetic algorithms and local search procedures. Future works could explore the possibility of designing more sophisticated architectures of genetic search with migration and applying the proposed approach to solve the two-dimensional bin packing problem.

References


