Computers & Industrial Engineering 75 (2014) 176–186

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Optimization of one-dimensional Bin Packing Problem with island parallel grouping genetic algorithms ☆

Tansel Dokeroglu *, Ahmet Cosar

Department of Computer Engineering, Middle East Technical University, Inonu Bulvari, 06800 Ankara, Turkey

A R T I C L E I N F O

Article history:
Received 20 September 2013
Received in revised form 19 May 2014
Accepted 3 June 2014
Available online 30 June 2014

Keywords:
One-dimensional bin packing
Grouping genetic algorithms
Parallel computation
Hybrid-heuristics

A B S T R A C T

The well-known one-dimensional Bin Packing Problem (BPP) of whose variants arise in many real life situations is a challenging NP-Hard combinatorial optimization problem. Metaheuristics are widely used optimization tools to find (near-) optimal solutions for solving large problem instances of BPP in reasonable running times. With this study, we propose a set of robust and scalable hybrid parallel algorithms that take advantage of parallel computation techniques, evolutionary grouping genetic metaheuristics, and bin-oriented heuristics to obtain solutions for large scale one-dimensional BPP instances. A total number of 1318 benchmark problems are examined with the proposed algorithms and it is shown that optimal solutions for 88.5% of these instances can be obtained with practical optimization times while solving the rest of the problems with no more than one extra bin. When the results are compared with the existing state-of-the-art heuristics, the developed parallel hybrid grouping genetic algorithms can be considered as one of the best one-dimensional BPP algorithms in terms of computation time and solution quality.

1. Introduction

Many practical optimization problems require efficient grouping (clustering) of a set of items into a collection of disjoint subsets according to some specific constraints. The well-known one-dimensional Bin Packing Problem (BPP) is such a NP-hard combinatorial grouping problem that often occurs in real life including engineering, logistics, and manufacturing (Garey & Johnson, 1979; Johnson, Demers, Ullman, Garey, & Graham, 1974). The BPP appears as the main problem or a significant subproblem in a large number of industrial applications (Camacho, Terashima-Marin, Ochoa, & Conant-Pablos, 2013; Fleszar & Charalambous, 2011; Fleszar, 2012). Two/three dimensional versions of BPP are frequently faced during manufacturing processes. For instance, in clothing, construction, glass, plastic, or metal industries, components need to be cut from the fewest sheets of material (Dahmani, Clautiaux, Krichen, & Talbi, 2013). Similarly, when designing the layout of the pages of a newspaper, the editor needs to arrange the articles on pages of fixed dimensions. In the shipping and transportation industries, packages of identical heights have to be loaded in the minimum number of rectangular bins.

Informally, the BPP is the process of packing n items into a number of same size and shape (for ease of planning and transportation) bins with a capacity of c where the objective is to minimize the number of bins required to contain n items (Gupta & Ho, 1999; Martello & Toth, 1989). The number of bins is assumed to be unlimited, each with capacity $c > 0$. A set of n items is to be packed into bins and the size of item $i \in \{1, \ldots, n\}$ is $s_i > 0$, as shown in Eq. (1):

$$\forall k : \sum_{i \in bin(k)} s_i \leqslant c \tag{1}$$

The goal of one-dimensional BPP is to find the minimum number of bins, M, required for packing all n items (Eq. (2)):

$$M \geqslant \left\lceil \left(\sum_{i=1}^{n} s_i \right) \Big/ c \right\rceil \tag{2}$$

Modern parallel computation environments such as grid and high performance computing have received intensive attention from industry and academia in recent years because of their ability to solve hard problems more efficiently and accurately. In order to overcome the processing power limits, multi-core processors integrate many cores into one chip and deliver higher total computing power. By developing parallel algorithms that can efficiently use all

☆ This manuscript was processed by Area Editor Mitsuo Gen.
* Corresponding author. Tel.: +90 312 210 2080; fax: +90 312 210 5544.
E-mail addresses: tansel@ceng.metu.edu.tr (T. Dokeroglu), cosar@ceng.metu.edu.tr (A. Cosar).

http://dx.doi.org/10.1016/j.cie.2014.06.002
0360-8352/© 2014 Elsevier Ltd. All rights reserved.

of the available processors, the processing power supplied by multi-core processors can increase the solution quality of one-dimensional BPP (Fernandez, Gil, Banos, & Montoya, 2013).

In addition to these benefits of multi-core processors, genetic algorithms (GAs) continue to be promising tools for numerous NP-Hard optimization problems where exact solution methods tend to fail because of the exponential search spaces (Cantu-Paz, 2000; Holland, 1975; Luque & Alba, 2011; Mitchell, 1996; Zitzler & Thiele, 1999). The study of Rohlfshagen and Bullinaria (2007) that we have parallelized is such a recent technique for solving the one-dimensional BPP that views individuals as (eukaryotic) genes instead of (prokaryotic) genomes in order to improve the traditional design of GAs.

In this study, our main motivation was to solve the one dimensional BPP with island parallel grouping genetic algorithms (GGAs) that take advantage of state-of-the-art computation tools. Island parallel GAs are very efficient tools for the solution of many NP-Hard problems (Cantu-Paz, 2000; Gordon & Whitley, 1993; Lim, Yuan, & Omatu, 2000; Luque & Alba, 2011). They bring many of the state-of-the-art computational tools together for better results.

With the proposed novel algorithms, we have combined state-of-the-art computation tools; parallel processing, GGAs, and bin oriented heuristics to efficiently solve the intractable one-dimensional BPP. The majority (88.5%) of the 1318 benchmark problem instances are solved optimally while the rest of the solutions produce only one extra bin. Novel hybrid bin-oriented heuristics with polynomial running times are proposed for GGAs and used to reinsert the remaining items into bins after the process of crossover and mutation. The solution quality of sequential GGAs is improved by the use of subpopulations on multi-core processors and it is shown that there is a potential to solve the BPP even more accurately if additional processors can be supplied.

In Section 2, we briefly review the relevant and the best performing sequential and parallel recent methods in chronological order. Section 3 explains the GGAs, canonical GA crossovers, Falkenauer's crossover, molecular genetics exon shuffling crossover, mutation, and inversion operator. Section 4 gives brief information about the use of BPP heuristics in the developed algorithms. Section 5 defines the proposed parallel hybrid GGAs and the parameter settings. Experimental comparisons of the developed algorithms are given in Section 6. Finally, conclusions and further research directions are discussed in Section 7.

2. Related work

One-dimensional BPP has been solved by numerous heuristics and exact techniques. A classical reference for the BPP is the book by Martello and Toth (1990), in which they describe a number of simple heuristics and lower bounds, introduce a reduction procedure (MTRP), and an exact algorithm (MTP). Best-Fit-Decreasing (BFD) and the First-Fit-Decreasing (FFD) algorithms are the two best known heuristics. Both heuristics sort items in decreasing order and place the largest item in either the first bin it fits (FFD) or the bin with the smallest but sufficient remaining capacity (BFD). A new bin is added whenever no suitable bin can be found.

Studies of Falkenauer (1994) were the early experiments in evolutionary algorithms to describe a Grouping Genetic Algorithms (GGA). Coffman, Garey, and Johnson (1997, 1999) studied approximation algorithms. Scholl, Klein, and Jurgens (1997) developed BISON, an efficient exact algorithm. Schwerin and Wascher (1997) introduced an improved version of the MTP. Valério de Carvalho (1999) developed an exact algorithm with branch-and-bound and studied linear programming models. Vanderbeck (1999) developed an exact algorithm based on column generation. Gupta and Ho (1999) developed the Minimum-Bin-Slack (MBS) heuristic. Fleszar and Hindi (2002) introduced a new heuristic MBS$'$, which fixes the largest remaining item in a bin before proceeding with the enumeration. Ross, Marin-Blázquez, Schulenburg, and Hart (2003) described a GA-based approach that learns a heuristic combination for solving the BPP. The worst-case performance of heuristics was investigated by Caprara and Pferschy (2004, 2005). Bhatia and Basu (2004) described a multi-chromosomal GGA. A highly effective hybrid improvement heuristic is described by Alvim, Ribeiro, Glover, and Aloise (2004). Singh and Gupta (2007) introduced a compound heuristic, combining a hybrid steady-state GGA. Stawowy (2008) proposed a non-specialized and non-hybridized algorithm that uses a modified permutation with separators encoding scheme, unique concept of separators movements during mutation, and separators removal as a technique of problem size reduction.

Rohlfshagen and Bullinaria (2007) developed an algorithm modeled on the theory of exon shuffling. Poli, Woodward, and Burke (2007) described an algorithm with discrete item sizes by matching the item-size histogram with the bin-gap histogram. Crainic, Perboli, Pezzuto, and Tadei (2007) studied fast lower bounds and their worst-case performance. Loh, Golden, and Wasil (2008) presented a weight annealing heuristic.

Khanafer, Clautiaux, and Talbi (2010) proposed a framework for deriving new data-dependent dual feasible functions. Fleszar and Charalambous (2011) proposed a method of controlling the average weight of items packed by bin-oriented heuristics in which constructive heuristics and an improvement heuristic are introduced. Memetic algorithms are successfully used for the one dimensional BPP and one of these approaches used separate individual learning or local improvement procedures (Le, Ong, Jin, & Sendhoff, 2009; Ong, Lim, Zhu, & Wong, 2006). Segura, Segredo, and Leon (2011) described a multi-objectivized memetic algorithm for the two-dimensional BPP which performs faster than traditional genetic algorithms.

Parallelization studies date back to 1990s. In a theoretical study, Anderson, Mayr, and Warmuth (1989) studied the parallel complexity of polynomial heuristics for the BPP and showed that some well-known simple heuristics are P-complete and they are not likely to be parallelized efficiently. Bestavros, Cheatham, and Stefanescu (1990) studied the asymptotic behavior of the different heuristics and proposed a set of simple data parallel algorithms to provide linear speedup for the packing of hundreds of thousands of bins. Coleman and Wang (1992) introduced a bin packing heuristic that is well-suited for implementation on massively parallel SIMD or MIMD computing systems. The average-case behavior of the technique was predictable when the input data has a symmetric distribution. The method is asymptotically optimal, yields perfect packing and achieves the best possible average case behavior with high probability. A systolic based parallel approximation algorithm that obtains solutions to the one dimensional BPP is presented by Berkey and Wang (1994). The algorithm has an asymptotic error bound of 1.5 and time complexity of $O(n)$. Fernandez et al. (2013) have analyzed how to solve two-dimensional BPPs with rotations and load balancing using parallel and multi-objective memetic algorithms that apply a set of search operators specifically designed to solve this problem. Results obtained using a set of test problems show the good performance of parallel and multi-objective memetic algorithms in comparison with other methods found in the literature.

Although there are existing parallel algorithms for the solution of two/three dimensional BPP (Bozejko, Uchronski, & Wodecki, 2010; Fernandez et al., 2013; Kroger, Schwenderling, & Vornberger, 1991; Pargas & Jain, 1993), to the best of our knowledge, there is no proposed island parallel hybrid GGAs like ours in the literature for the solution of one dimensional BPP. Parallel GAs are efficient tools for the solution of many different NP-Hard problems (Cantu-Paz, 2000; Gordon & Whitley, 1993; Lim et al., 2000; Luque & Alba, 2011) however, they have not been applied to the one dimensional BPP. Parallel GGAs facilitate different sub-populations to evolve in diverse directions concurrently, thus potentially producing higher-quality solutions by producing larger number of individuals and generations.

3. Grouping genetic algorithms for one dimensional Bin Packing Problem

A group is defined as a set of similar objects on which a measure of similarity can be defined. The grouping satisfies the constraints that limit the groups into which a given item can be placed (Falkenauer, 1994). The goal of optimization is to find the grouping with the minimum cost. In this section, we give information about the solution representation (chromosome) of Holland-style and Falkenauer's chromosomes, degeneracy, crossover, mutation, and inversion.

3.1. Falkenauer's chromosome representation

Holland-style GA chromosomes are the first representations used by Raf and Robert (1992), Ding, El-Keib, and Smith (1992), Gregor (1991) to solve the BPP and they are shown to be inefficient. Fig. 1 shows a Holland-style representation for one dimensional BPP. In the example chromosome, there are 6 bins (with capacity $c = 10$) packing the item set $n = \{2,3,3,3,4,4,5,5,6,7\}$. Each letter (gene) represents the group name of an item in set n, which means that (starting from the left-hand side of the chromosome) item 2 is in group E, item 3 is in group C, and etc.

Holland-style crossover operators generate chromosomes that have nothing in common with the parents and lose the knowledge with respect to the solution. As it can be seen from Fig. 2, child 1 and 2 are invalid chromosomes because group B in child 1 and group A in child 2 exceed the bin capacity ($c = 10$).

Another important drawback of Holland-style chromosome for GGAs is that it may represent the same solution with different chromosomes. This problem is called *degeneration* (Radcliffe, 1991) which leads to inefficient coverage of the search space where the same configuration of groups are explored repeatedly. This leads to repeatedly exploring the same subset of chromosomes in search space. The minimization of degeneracy is a key component of a good design and improves the performance of GGAs (Falkenauer, 1996; Radcliffe, 1991).

GGAs work better with a special encoding scheme that was proposed by Falkenauer (1996) to make relevant structures of grouping problems become genes in chromosomes (Radcliffe & Surry, 1995). With this encoding scheme, more efficient crossover, mutation, and inversion operators are described to maintain the information gained by the chromosomes. The new chromosome is augmented with a group part. In addition to the genes representing the groups of each item, the new chromosome structure represents the groups. In order to avoid the problems of Holland-style chromosomes the representation of Falkenauer is employed in our proposed algorithms (see Fig. 3 for Falkenauer's chromosome).

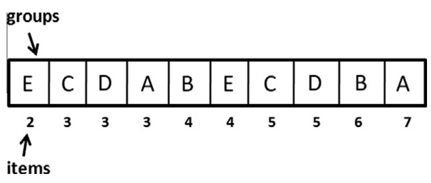


Fig. 1. A sample Holland-style chromosome representation for one dimensional BPP with 5 bins (A, B, C, D, E).

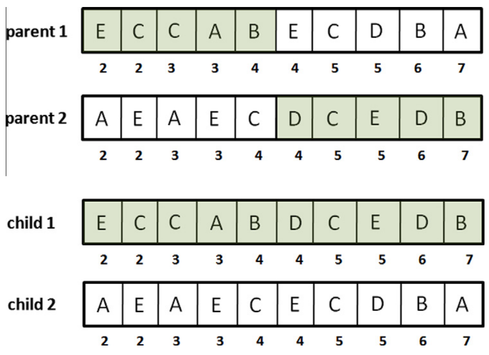


Fig. 2. Holland-style crossover that generates invalid new chromosomes.

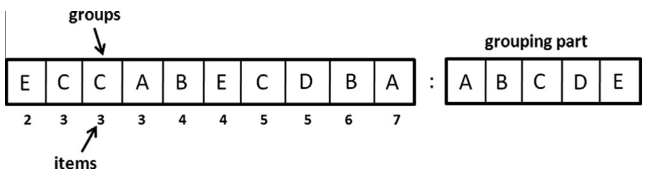


Fig. 3. GGA chromosome.

There are five groups (bins) in the example and they are represented with an additional "grouping part" array that is attached to the end of the item list.

3.2. Crossover, mutation, and inversion operators of Falkenauer's chromosome

Falkenauer's chromosome operators work with the grouping part of a chromosome. The left-hand side of the chromosomes identifies only the groups of the items. The grouping part has more meaning than the other part of the chromosome. In order to explore the search space more effectively and find the more promising solution regions, operators work on the groups rather than items. If chromosomes do not allow the subsets to be exploited, then the GGA fails and the algorithm performs only a little better than a random search. Under these constraints, GGA operators need to handle variable length chromosomes of whose group orders are irrelevant.

The crossover operator of Falkenauer's chromosome works with the variable length chromosomes that represent the groups (bins). The procedure used for combining genes from both parents depends on the specific problem and its constraints. The process of the crossover operator for the one dimensional BPP can be seen

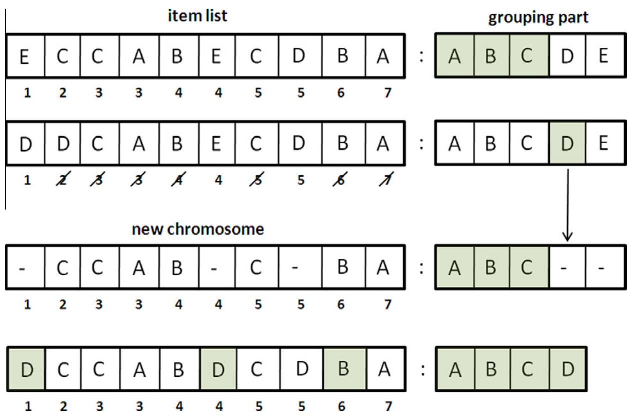

item list
grouping part
E C C A B E C D B A : A B C D E
1 2 3 3 4 4 5 5 6 7
D D C A B E C D B A : A B C D E
1 2 3 3 4 4 5 5 6 7
new chromosome
- C C A B - C - B A : A B C - -
1 2 3 3 4 4 5 5 6 7
D C C A B D C D B A : A B C D
1 2 3 3 4 4 5 5 6 7

Fig. 4. GGA crossover operator for Falkenauer's chromosome.

in Fig. 4. After selecting two parents from the population, a crossing point is chosen in the first parent. Later, the bins selected from the left-hand side of the crossing point of the first parent are copied to the new and empty chromosome. The items used in the new chromosome are eliminated from the second parent. During this process, some of the bins in the second parent remain unchanged (bin D is unchanged in the example). These bins are added to the new chromosome in their original form. The other remaining items of the second parent are reinserted into the new bins by using heuristics. Falkenauer suggests FFD for the reinsertion of the remaining items Falkenauer (1994). In this study, we propose efficient hybrid variants of Minimum Bin Slack (MBS) heuristic for this phase of the crossover operator.

The mutation operator for Falkenauer's chromosome inserts new characteristics into the population to enhance the search space by diversification. The mutation operator works on a single chromosome. Two genes (bins) that have slacks are randomly selected from the chromosome (bins A and D are selected in the example given in Fig. 5). The items of the selected bins are removed from the chromosome and new bins are built by reinserting these items (items 1, 4, and 5 are reinserted to a new bin) with heuristics. The five-bin chromosome is improved to a four-bin chromosome by using the mutation operator. The details of the operator can be seen in Fig. 5.

Inversion operator proposes the same solution with a different representation. In order to increase the chance for fitter bins to be selected together in crossover and mutation operations, an inversion operator should be used. Fig. 6 gives an example of the inversion operator which increases the probability of fitter gene pairs (A,D) being passed together to future generations. Item list does not change during the operation.

3.3. Exon shuffling crossover

In addition to Falkenauer's crossover operator, we have used exon shuffling crossover (Kolkman & Stemmer, 2001; Rohlfshagen & Bullinaria, 2007), a recent technique borrowed from molecular genetics, for our proposed parallel algorithms. Molecular genetics is the field of biology and genetics that studies genes at a molecular level and employs methods to elucidate the molecular function and interactions among genes. An offspring is generated by a two phase crossover. In the first phase, all mutually exclusive segments are combined. In the second phase, the remaining items are used to build a new bin.

An example of exon shuffling crossover using a bin of size 100 with 20 items is given in Figs. 7 and 8. In Fig. 7, two parent chromosomes that will produce an offspring with exon shuffling are presented. Parents 1 and 2 have 11, 13 bins and 223, 423 total empty spaces respectively. In Fig. 8, the bins of the parents are

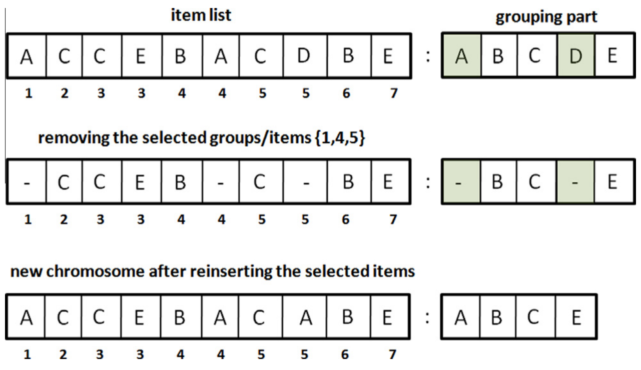

item list
grouping part
A C C E B A C D B E : A B C D E
1 2 3 3 4 4 5 5 6 7
removing the selected groups/items {1,4,5}
- C C E B - C - B E : - B C - E
1 2 3 3 4 4 5 5 6 7
new chromosome after reinserting the selected items
A C C E B A C A B E : A B C E
1 2 3 3 4 4 5 5 6 7

Fig. 5. Mutation operator for Falkenauer's chromosome.

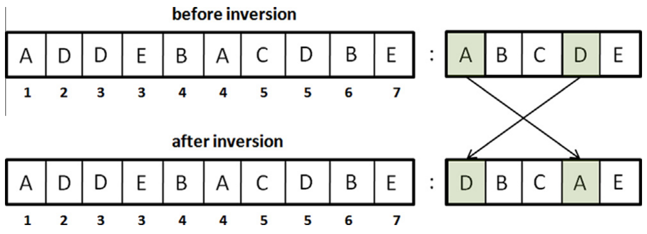


Fig. 6. GGA inversion operator.

combined together and sorted according to their fitness values. Later, if the current bin in the queue can be built from the remaining items shown on the right-hand side, it is appended to the offspring as a new gene. Otherwise, the current bin is skipped and the next bin is evaluated. As it can be seen from the example, it was possible to produce a new chromosome (offspring) with 10 bins and 133 total empty space, which is better than the parents.

3.4. Fitness function

The fitness value of a chromosome is defined as proposed by Falkenauer (see Eq. (3)):

$$FF = \sum_{i=1}^{nb} \left(\frac{F_i}{c}\right)^k \tag{3}$$

Using the number of bins as a fitness function (FF) may lead to a useless landscape of search space with suboptimal solutions.

In Eq. (3), nb is the number of bins, F_i is the sum of weights of the elements packed into the bin i ($i = 1, \ldots, nb$), c is the bin capacity, and k is a heuristic exponential factor. The value k expresses a concentration on the almost full bins in comparison to less filled ones. Falkenauer (1994) uses $k = 2$ but Stawowy (2008) reported that $k = 4$ gives slightly better results.

4. Bin-oriented heuristics for one-dimensional BPP

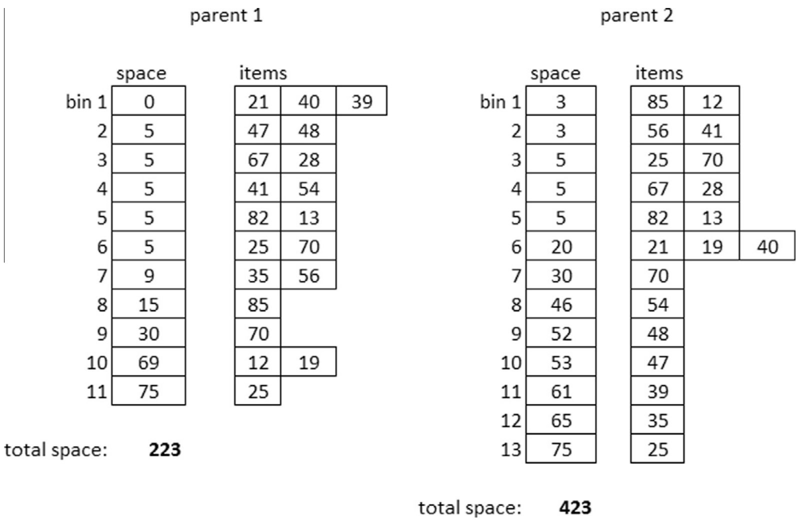
In this section, we give information about the well-known minimum bin slack (MBS) heuristic and our proposed hybrid heuristics, MBS with Best First Decreasing (MBS-BFD) and MBS with First Fit Decreasing (MBS-FFD). The computation time of the proposed heuristics has polynomial complexity instead of exponential and provides a good reinsertion mechanism when the search space of MBS becomes prohibitively large.

4.1. Minimum bin slack (MBS) heuristic

MBS is a bin-oriented heuristic that is first proposed by Gupta and Ho (1999). In each iteration, an attempt is made to fill the bin with the minimum slack. The items that exist in a new bin are removed from the list of items and all the subsets of unassigned items are considered for the selection of the items subset that will yield the minimum slack in the current bin. MBS can explore the search space more efficiently than the BFD and FFD algorithms and provides bins with the minimal slack but its optimization time is prohibitive. The computational complexity of MBS is $O(2^n)$, if the maximum number of items that fits a bin is limited by u, then its complexity reduces to $O(n^{u+1})$.

4.2. Proposed hybrid heuristics (MBS-BFD, MBS-FFD)

With the two proposed novel hybrid heuristics, we calculate the possible number of subsets to be inspected to find the optimal solution before starting to insert the remaining items into a new



parent 1

bin	space	items		
bin 1	0	21	40	39
2	5	47	48	
3	5	67	28	
4	5	41	54	
5	5	82	13	
6	5	25	70	
7	9	35	56	
8	15	85		
9	30	70		
10	69	12	19	
11	75	25		

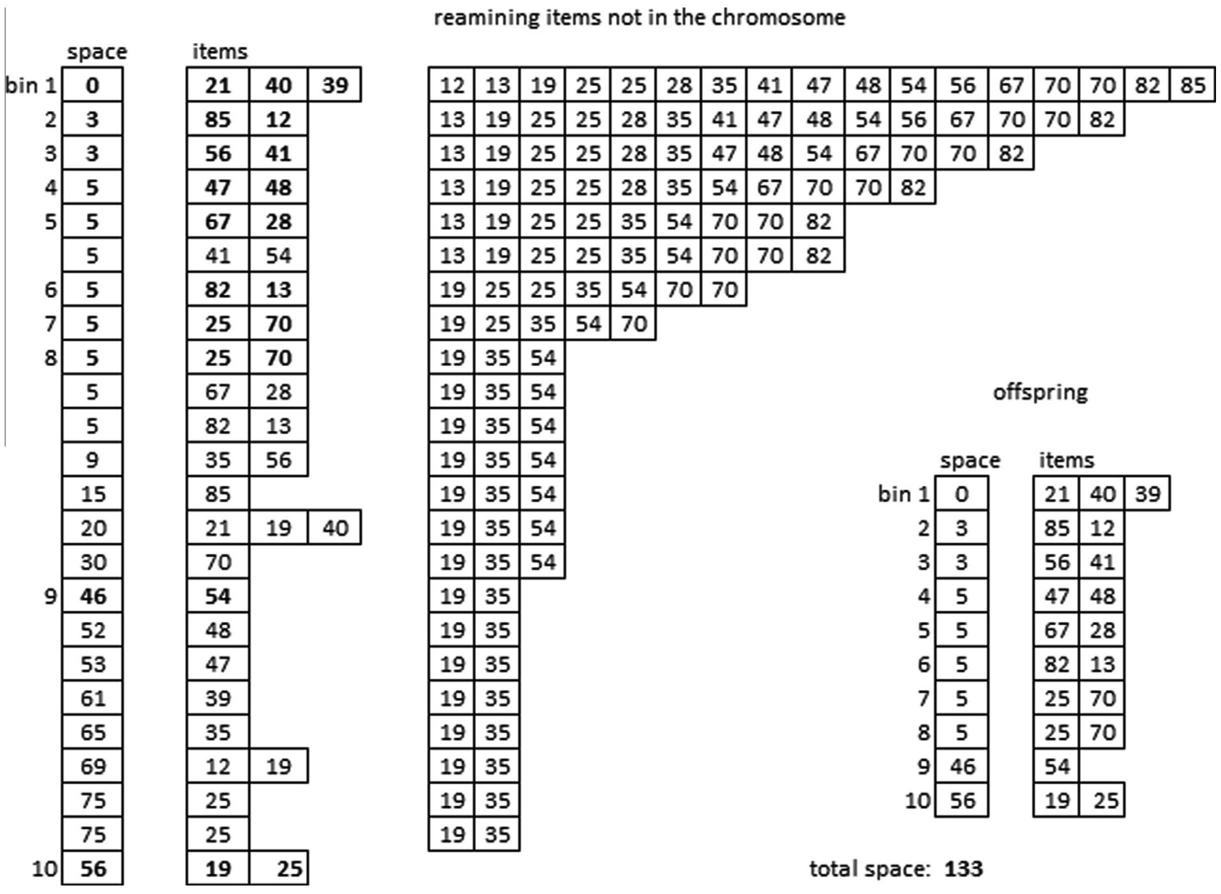
total space: **223**

parent 2

bin	space	items		
bin 1	3	85	12	
2	3	56	41	
3	5	25	70	
4	5	67	28	
5	5	82	13	
6	20	21	19	40
7	30	70		
8	46	54		
9	52	48		
10	53	47		
11	61	39		
12	65	35		
13	75	25		

total space: **423**

Fig. 7. Parents selected for exon shuffling.



bin	space	items		
bin 1	**0**	**21**	**40**	**39**
2	**3**	**85**	**12**	
3	**3**	**56**	**41**	
4	**5**	**47**	**48**	
5	**5**	**67**	**28**	
	5	41	54	
6	**5**	**82**	**13**	
7	**5**	**25**	**70**	
8	**5**	**25**	**70**	
	5	67	28	
	5	82	13	
	9	35	56	
	15	85		
	20	21	19	40
	30	70		
9	**46**	**54**		
	52	48		
	53	47		
	61	39		
	65	35		
	69	12	19	
	75	25		
	75	25		
10	**56**	**19**	**25**	

reamining items not in the chromosome

12	13	19	25	25	28	35	41	47	48	54	56	67	70	70	82	85
13	19	25	25	28	35	41	47	48	54	56	67	70	70	82		
13	19	25	25	28	35	47	48	54	67	70	70	82				
13	19	25	25	28	35	54	67	70	70	82						
13	19	25	25	35	54	70	70	82								
13	19	25	25	35	54	70	70	82								
19	25	25	35	54	70	70										
19	25	35	54	70												
19	35	54														
19	35	54														
19	35	54														
19	35	54														
19	35	54														
19	35	54														
19	35	54														
19	35															
19	35															
19	35															
19	35															
19	35															
19	35															
19	35															
19	35															

offspring

bin	space	items		
bin 1	0	21	40	39
2	3	85	12	
3	3	56	41	
4	5	47	48	
5	5	67	28	
6	5	82	13	
7	5	25	70	
8	5	25	70	
9	46	54		
10	56	19	25	

total space: **133**

Fig. 8. Exon shuffling crossover.

bin with MBS. If it is not possible to obtain results in reasonable time, we apply the BFD or FFD heuristics instead of the MBS (10^6 combinations is chosen as a threshold in our experiments and gives good results). MBS explores the search space more accurately but because the duration of its execution time is prohibitive, we needed to prune the search space with efficient and polynomial time heuristics. These hybrid heuristics we have introduced are called MBS-BFD and MBS-FFD. Algorithm 1 gives the details of the MBS-BFD. Its only difference from MBS is the statement:

if the number of subsets of (Z') is below threshold value (10^6) **then**

MBS-FFD is also designed with the same principles of MBS-BFD. The computational complexity of these novel heuristics is polynomial and depends on the threshold value selected by the algorithm to apply the BFD/FFD heuristic when the search space of MBS becomes prohibitively large. Comprehensive experimental results of these novel heuristics are given in Section 6.

Algorithm 1. Minimum Bin Slack with BFD (MBS-BFD)

```
n′: number of items not assigned to any bins (sorted in
    decreasing order)
q: index of items in n′
Z′: list of n′ items
A: the set of items in the current bin
s(A): slack in A
A*: the set of items in the best packing, initially empty
t_i: weight of item i

Procedure MBS-BFD (q)
for r:=q to n′ do
  i:=Z′_r
  if t_i ≤ s(A) then
    A:=A ∪ {i}
    if the number of subsets of (Z′) is below (10^6), then
      MBS (r+1)
    else
      BFD (Z′) // details of BFD are explained previously.
      return
    end if
    A:=A\{i}
    If s(A*)=0 then return
  end if
end for
If s(A) < s(A*) then A*:= A
```

5. Description of proposed algorithms

In this section, we give information about eight different parallel hybrid GGAs that we have proposed for the solution of one-dimensional BPP. Our main goal with the proposed algorithms is to take advantage of multiprocessors to generate several non-redundant populations and explore the search space more efficiently rather than reducing the optimization time of GGA with a single population by calculating the fitness values with different processors. We have developed algorithms with two types of crossovers, Falkneaur's crossover (Falkenauer, 1996) and the molecular genetics exon shuffling crossover (Rohlfshagen & Bullinaria, 2007).

The first step of the island parallel GGAs is the generation of an initial population at each processor. The most appropriate way that the knowledge about BPP can be incorporated into the proposed algorithms is to include the solutions from the heuristics as initial solutions. Therefore, we added individuals produced by the BFD (5%) and FFD (5%) heuristics to the initial population in addition to a set of randomly generated individuals. This cooperation with the other heuristics guarantees the lower bounds obtained by these algorithms for the proposed parallel hybrid algorithms. After generating individuals more than the population size, truncation selection is applied to the population and the best the individuals are selected as an initial population. Later, we have applied crossover, mutation, and inversion operators throughout the generations. No reduction method is applied to reduce the problem size such as MTRP (Martello & Toth, 1990) and Reeves reduction rule (RRR) (Reeves, 1996).

After crossover and mutation operations, remaining items are reinserted into new bins with heuristics in our proposed algorithms. The heuristics BFD and FFD have been used by previous studies for this purpose (Falkenauer, 1994, 1996) however; MBS is observed to have better performance in our studies. BFD and FFD have polynomial optimization times where MBS has an exponential optimization running time. Therefore, although MBS explores the solution space more efficiently and builds bins with minimum slack, it is prohibitive to finish the optimization process in practical running times for larger item sets. In order to overcome this drawback of MBS, we have used the heuristics MBS-BFD and MBS-FFD and to evaluate their performances against BFD and FFD. Our proposed algorithms are presented in Table 1.

Algorithm 2. Parallel Falk-MBS-BFD Algorithm

```
if MASTER then
  for k:=1 to migration limit do
    for rank:=1 to number of slaves do
      receive the best individual from slave[rank]
      update the global best individual that is found
      send the global best individual to slave[rank]
    end for
  end for
  for rank:=1 to number of slaves do
    receive the best individual from slave[rank]
  end for
  show the global best individual received from slaves
end if

if SLAVE then
  p ←generate 90% random and 10% heuristic
  (5% BFD, 5% FFD) individuals
  for k:=1 to number of generations do
    for m:=1 to (population size/2) do
      (p_1,p_2)←select pair of parents(p)
      s ← crossover (p_1,p_2)
      s ← mutation (p,s)
      s ← inversion (p,s)
    end for
    if k ≤migration limit then
      send the best individual to the master
      receive the global best individual from the master
    end if
  end for
  send the best individual in the population to the master
end if
```

The parameters used in the algorithms can be listed as:

Population size: Total number of chromosomes (individuals) in a generation.

Number of generations: Each iteration of a GGA in which a number of crossovers, mutations, and inversions are applied.

Crossover ratio: The length of the crossed segment genes transferred in a Falkenaur crossover operation.

Mutation ratio: The probability of mutation for a gene in the individual.

Table 1
Crossover operators and reinsertion heuristics used in the proposed algorithms.

Algorithm name	Crossover	Reinsertion heuristic
Falk-BFD	Falkenauer	BFD
Falk-FFD	Falkenauer	FFD
Falk-MBS-BFD	Falkenauer	MBS-BFD
Falk-MBS-FFD	Falkenauer	MBS-FFD
Exon-BFD	Exon Shuffling	BFD
Exon-FFD	Exon Shuffling	FFD
Exon-MBS-BFD	Exon Shuffling	MBS-BFD
Exon-MBS-FFD	Exon Shuffling	MBS-FFD

Inversion ratio: The probability of inversion for a gene in the individual.

Threshold for MBS: The number of combinations that the MBS heuristic produced cannot exceed during the optimization.

Selection type (tournament): r chromosomes are chosen from the population and the individuals with the best fitness value are passed to the next generation from the *r*-element group.

Tournament size: Number of individuals in the tournament selection technique before the best two are selected.

Exponent k: Exponential value for fitness function.

Termination Condition: The proposed algorithms terminate their execution either when they find the optimal result (that are given in the problem instances library) or when they arrive at the end of the defined number of generations.

We have used a migration method to increase the quality of the populations. The illustration of the migration topology is presented in Fig. 9. The number of exchanged individuals is carefully controlled so that even when the number of processors is increased, the communication overhead does not adversely affect the optimization time of the algorithm. Each sub-population is assigned to a processor by constituting an island which evolves in isolation for most of the duration of the parallel optimization. Communication between the master node and slaves leads to better solutions. Therefore, a well-designed migration policy helps the sub-populations evolve more effectively. Since the parallel computation strategy in this study is based on the island model migrations are essential operations and they allow the collaboration among islands/processors. The migration stage is applied only at the initial generations of the algorithm. Slaves send the local best individuals to the master node and receive the best global ones. This procedure provides a successful collaboration for a parallel GGA. The solution search space can be better explored and higher quality solutions could be obtained with this method (Cantu-Paz, 2000; Luque & Alba, 2011). Unsuitable migration methods can be a worse choice than having separate executing on several processors without any communication. Therefore, our migration policy is designed carefully that the rate is very small to provide scalability. Migration is only undertaken with the initial generations (first three generations) of the algorithms to prevent an excessive communication overhead. Details of the proposed parallel hybrid GGAs are presented in Algorithm 2.

6. Experimental results and discussion

In this section, we give information about the environment of our experiments (high performance cluster (HPC) machine), problem instances, parameter settings, performance evaluations (in terms of execution time and produced results), robustness, and scalability of the proposed algorithms. In the last part, we report

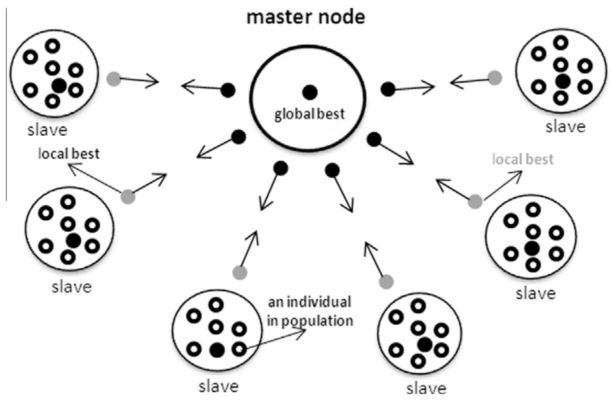


Fig. 9. Migration topology of the proposed parallel algorithms.

the results of the best performing algorithm (Exon-MBS-BFD) with 1318 problem instances and compare its performance on the hard28 (Belov, Scheithauer, & Mukhacheva, 2007) problem instances with other state-of-the-art algorithms in the literature.

6.1. Experimental environment and problem instances

Our experiments are performed on a high performance cluster (HPC) computer. The machine has 46 nodes, each with 2 CPUs giving a total of 92 CPUs. Each CPU has 4 cores with a total of 368 cores. Each node has two 24 port Gigabit ethernet switches and one 24 port high performance switch. The software comprises; a Scientific Linux v4.5 64-bit operating system, Lustre v1.6.4.2 parallel file system, Torque v2.1.9 resource manager, Maui v3.2.6 job scheduler, Open MPI v1.2.4, and the C++.

Three sets of problem instances are used in the experiments. The instances are (u_120, u_250, u_500, u_1000) (Falkenauer, 1996), set_1, set_2, set_3 (Scholl et al., 1997), hard28 (Belov et al., 2007). The details of the problem instances are presented in Table 2.

6.2. Parameter settings for the proposed algorithms

Before analyzing the performance of the proposed algorithms on our HPC with different problem instances, we have performed some experiments with sequential version of the algorithms to decide the (near-) optimal values for the population size and the number of generations. The other parameters (crossover ratio, mutation ratio, MBS-threshold, and inversion ratio) are decided according to the previous studies in Stawowy (2008), Falkenauer (1994) and Holland (1975).

First, we have performed experiments with increasing number of individuals to find reasonable values for the algorithms. During the experiments, the number of generations was 50, MBS-threshold value was 10^6, crossover ratio was 10–50%, mutation ratio was (1%), inversion ratio was (10%), and selection mechanism was tournament. Although these experiments are performed to set (near-)optimal parameters for the population size and the number of generations, we were also able to observe the performance of the algorithms and decide the best performing one with these results.

Table 3 gives the number of optimal solutions and extra bins produced by the proposed algorithms. The quality of the solutions produced by the algorithms using MBS-BFD heuristic are observed to be better than the solutions that are produced by the algorithms using BFD and FFD heuristics. Increasing the number of individuals has a positive effect on the solution quality of the algorithms. The number of optimal solutions and the number of extra bins improves as the number of individuals is increased. Although larger number of individuals provide better results, due to the optimization time limitations we have decided 40 individuals as our (near-) optimal parameter for the experiments that we have performed on the HPC. This population size was also proposed by Stawowy (2008).

The number of generations was another important parameter that we need to tune for the proposed algorithms. We have set all of the other parameters to a reasonable constant value and increased the value of generations starting with 20 up to 100 individuals and observed the quality of the solutions that are found by the algorithms. Table 4 presents the results of the solutions. The quality of the solutions improves as the number of generations is increased. The best performing algorithms are observed to be the variants with MBS-BFD heuristic. For later experiments, we have decided 100 generations as a (near-)optimal value for our proposed algorithms. During all these experiments the best performing algorithm was Exon-MBS-BFD.

Table 2
Information about the problem instances.

Problem instance	# Instances	Item weights	Bin capacity (*c*)	# Items (*n*)
u_120	20	[20, 100]	150	120
u_250	20	[20, 100]	150	250
u_500	20	[20, 100]	150	500
u_1000	20	[20, 100]	150	1000
set_1	720	[1,100]	{100, 120, 150}	{50, 100, 200, 500}
set_2	480	[3, 9] items at each bin	1000	{50, 100, 200, 500}
set_3	10	[20,000, 35,000]	100,000	200
hard28	28	[1, 800]	1000	{160, 180, 200}

Table 3
Results of experiments with increasing population sizes. (opt.)=# of optimal solutions, (ext.)= # of extra bins produced (Bold values are the best results obtained by the algorithms).

Algorithm	20		40		60		80		100		Average	
	Opt.	Ext.	Opt.	Ext.	Opt.	Ext.	Opt.	Ext.	Opt.	Ext.	Opt.	Ext.
Falk-BFD	58	69	59	64	61	56	63	55	64	55	60.7	60.6
Falk-FFD	62	65	65	54	67	52	67	51	68	50	65.5	56.3
Falk-MBS-BFD	67	58	68	50	70	43	73	41	75	40	70.2	47.0
Falk-MBS-FFD	65	61	68	50	71	46	72	44	73	42	69.5	50.0
Exon-BFD	61	55	68	37	71	32	72	31	73	30	68.4	39.8
Exon-FFD	61	60	68	40	70	34	73	30	76	28	68.5	41.1
Exon-MBS-BFD	68	48	72	33	74	29	76	26	78	24	**73.0**	**33.8**
Exon-MBS-FFD	67	50	70	34	74	29	77	27	78	25	72.6	34.3

Table 4
Results of experiments with increasing number of generations. (opt.)=# of optimal solutions, (ext.)= # of extra bins produced (Bold values are the best results obtained by the algorithms).

Algorithm	20		40		60		80		100		Average	
	Opt.	Ext.	Opt.	Ext.	Opt.	Ext.	Opt.	Ext.	Opt.	Ext.	Opt.	Ext.
Falk-BFD	58	66	59	62	59	61	60	59	61	58	59.3	61.9
Falk-FFD	60	62	63	57	65	55	66	54	67	53	63.9	57.1
Falk-MBS-BFD	65	53	69	46	71	43	72	42	74	40	69.9	46.1
Falk-MBS-FFD	66	55	67	49	68	47	69	46	70	44	67.5	49.1
Exon-BFD	64	47	67	40	68	36	69	35	72	33	67.2	39.1
Exon-FFD	66	43	68	36	70	35	73	33	75	31	69.8	36.5
Exon-MBS-BFD	69	37	72	30	75	28	76	26	77	25	**73.6**	**29.9**
Exon-MBS-FFD	67	40	72	32	74	28	76	26	78	25	73.1	30.9

Table 5
Parameter settings for the proposed GGAs.

Parameter	Value
Population size	40
Number of generations	100
Max. number of genes to transfer	50%
Min. number of genes to transfer	10%
Tournament size	10
Mutation ratio	1%
Inversion ratio	10%
Threshold for MBS	10^6
Exponent *k*	4

We have set 10^6 as our MBS threshold parameter, which performed well during the execution of the algorithms. After performing several experiments, we have set the (near-)optimal parameters that are presented in Table 5 for the proposed algorithms. These parameters are obtained from our experiments and previous studies of Holland (1975), Mitchell (1996), Tosun, Dokeroglu, and Cosar (2013) and Dokeroglu, Sert, and Cinar (2014).

6.3. Selecting the best performing sequential algorithm (Exon-MBS-BFD)

In order to select the best performing algorithm of those proposed, we performed additional experiments with set_1 and set_2 problem instances of Scholl et al. (1997). The algorithms are run 10 times with settings given in Table 5 and the average results are presented in Tables 6 and 7. The number of the optimal solutions found, the number of extra bins produced by the algorithms,

Table 6
The results of the sequential experiments with Set_2 (480) problem instances for the proposed algorithms (Bold values are the best results obtained by the algorithms).

Algorithms	# Optimal solutions	# Extra bins
BFD	225 (46.88%)	767
FFD	236 (49.17%)	748
Falk-BFD	256 (53.30%)	691
Falk-FFD	265 (55.21%)	667
Falk-MBS-BFD	**335 (69.79%)**	527
Falk-MBS-FFD	330 (68.75%)	540
Exon-BFD	275 (57.29%)	629
Exon-FFD	265 (55.21%)	639
Exon-MBS-BFD	**342 (71.25%)**	**494**
Exon-MBS-FFD	328 (68.33%)	552

Table 7
The results of the sequential experiments with Set_1 (720) problem instances for the proposed algorithms (Bold values are the best results obtained by the algorithms).

Algorithms	# Optimal solutions	# Extra bins
Falk-MBS-BFD	609 (84.58%)	144
Falk-MBS-FFD	594 (82.50%)	164
Exon-MBS-BFD	**612 (85.00%)**	**137**
Exon-MBS-FFD	600 (83.33%)	166

and the total optimization times of instances (as seconds) were the criteria used to decide the best performing algorithm. BFD and FFD algorithms are also added to the results. Exon-MBS-BFD algorithm is reported to be the best performing algorithm. It finds more optimal solutions than the other algorithms and the number of extra bins generated by the algorithm is the smallest.

6.4. Comparing the execution time of the algorithms

The optimization time of an algorithm is a crucial criterion to evaluate its efficiency. For our algorithms the most time consuming part is the reinsertion of items with heuristics. BFD and FFD algorithms have $O(n^2)$ optimization time. On the other hand, MBS has an exponential optimization time and its computational complexity is $O(2^n)$, but by restricting number of items that fit in one bin to u, it can be reduced to $O(n^{u+1})$. In order to prevent this side effect, we have pruned the search space of MBS and continued with the BFD/FFD heuristics to fill the bins with remaining items when the number of the combinations exceeds a threshold value. The optimization times of the BFD and FFD algorithms are smaller than the other algorithms but their performance in generating a single bin with the minimal slack is not as good as that of MBS. The hybrid heuristics we have introduced have a longer execution time than BFD and FFD but shorter than the MBS heuristic. Although the execution times of the proposed GGAs are longer, they are practical and can produce good results in a few minutes even for very large BPP instances. As the number of the items increases, the running time of the proposed GGAs increases (with a computational complexity of $O(n^2)$).

Also, the execution time depends on the total number of fitness evaluations performed during the optimization and it is equal to (number_of_generations × number_of_individuals). Additional processors provide only small overhead due to the messaging. In case of a need for rapid solutions, the optimization time of the proposed algorithms can be reduced by limiting the number of individuals and generations.

6.5. Experiments with parallel Exon-MBS-BFD algorithm

In this part of the study, we have performed experiments with the proposed parallel Exon-MBS-BFD algorithm. In order to see the benefits of a parallel algorithm against its sequential version, we have performed experiments with increasing number of processors by using parallel Exon-MBS-BFD algorithm on the first 100 problem instances of set_2. The solution quality of the algorithm increased with every additional processor that when the number of processors was 200 processors, the improvement in the number of optimal solutions became 20%.

Later, we have performed more comprehensive tests on 1318 problem instances to analyze the capabilities of parallel Exon-MBS-BFD algorithm. The experiments are performed with 40 slaves and a master node. The obtained results showed the remarkable performance of Exon-MBS-BFD algorithm. 88.54% of the instances are solved optimally and the solutions found for rest of the instances were only one bin more than the best solutions. The results of the solutions can be seen in Table 8.

Since the proposed parallel algorithm involve randomization, achieving robustness becomes an important issue. Each problem instance in the experiments is solved 10 times and their averages are reported to clarify this point. For 90% of the experiments, the same solutions are found. Compared to the remaining problem instances, the solutions have only one extra bin for the optimal results. Scalability is also an important aspect for the performance evaluation of all parallel algorithms and measures the capability to continue to give good results in shorter time when the number of processors is increased. In other words, scalability shows that we can increase the number of processors to solve the problem instances in the shortest possible execution time. In our proposed parallel algorithms, the optimization times are not increased due to the messaging as we increase the number of the processors. As the number of processors are increased 40 times (from 5 to 200 processors), the execution time of the algorithm doubles due to the messaging overhead of the processors, which is acceptable and we continue to substantially benefit from increased number of processors.

6.6. Comparison with state-of-the-art algorithms

A novel algorithm needs to be compared to existing algorithms to give an idea of its capabilities. Therefore, we have reported our comparisons with the available solutions of the best algorithms in the literature. Although there are many reported parallel 2D/3D algorithms, there are not many parallel one dimensional BPP solvers in the literature. The existing ones are earlier studies and they do not present any results for the benchmarks that we have studied on Anderson et al. (1989), Berkey and Wang (1994), Bestavros et al. (1990) and Coleman and Wang (1992). The existing state-of-the-art algorithms are mostly sequential. They have different methods to optimize the one dimensional BPP, which can be very complicated, whereas we propose simple heuristics that take advantage of multiprocessors.

Most of the state-of-the-art algorithms have test results with hard28 problem instances that are among the most challenging instances in the literature. Although most of the compared algorithms are not parallel and it may not seem fair to compare them with our results, it can give some idea about the efficiency of the parallel Exon-MBS-BFD algorithm. Since our algorithms are heuristic-based, the experimental results of the exact algorithms such as the branch-and-bound of Valério de Carvalho (1999) are not included in the comparisons. The algorithms we have made comparisons with are Best Fit Decreasing, First Fit Decreasing, MBS, MBS′, Best-two-fit (Friesen & Langston, 1991), SAWMBS′, and Pert-SAWMBS (Fleszar & Charalambous, 2011).

The results of the comparisons are reported in Table 9. Abbreviation (opt. sol.) is the number of optimal solutions reported by the

Table 8
Total number of optimal solutions found by the parallel hybrid Exon-MBS-BFD Algorithm with 100 processors.

Problem instance	# Instances	# Optimal solution	# Extra bins
u_120	20	20 (100%)	0
u_250	20	20 (100%)	0
u_500	20	17 (85.0%)	3
u_1000	20	18 (90.0%)	2
set_1	720	667 (92.6%)	53
set_2	480	412 (85.8%)	68
set_3	10	8 (80.0%)	2
hard28	28	5 (21.7%)	23
Total	1318	1167 (88.54%)	151

Table 9
Comparing the solutions of parallel Exon-MBS-BFD algorithm with the solutions of state-of-the-art algorithms for hard28 problems.

Algorithm	Opt. sol.	Avg. dev. %	Time (ms.)
Best Fit Decreasing	2	1.82	2.26
MBS′	2	1.39	3.64
MBS	3	1.29	4.20
Best-two-fit	4	1.07	3.65
SAWMBS′	5	0.82	129.92
First Fit Decreasing	5	0.82	2.24
Pert-SAWMBS	5	0.82	6,946.24
Parallel Exon-MBS-BFD	5	0.82	5,341.02

algorithm, (avg. dev. %) is the average percentage of the extra bins over the number of problem instances (28), time (ms.) is the optimization time of the algorithm for 28 problem instances. The BFD, MBS′, MBS, and Best-two-fit (Friesen & Langston, 1991) algorithms can find smaller number of optimal solutions compared to the Exon-MBS-BFD algorithm. The SAW heuristics of Fleszar and Charalambous (2011) that is reported to solve a large number of problem instances (not the same instances as used in our study) with more than 90% optimal solutions and FFD algorithms have the same results as the Exon-MBS-BFD. The non-optimal solutions found by the Exon-MBS-BFD for hard28 instances are only one bin away from the optimal values. Exon-MBS-BFD algorithm has solved 88.54% of 1318 problem instances with a remarkable success and practical optimization time and this shows that our algorithm is competitive moreover, Exon-MBS-BFD algorithm is scalable and has a potential to solve the one-dimensional BPP more accurately when additional processors are provided.

7. Conclusions and future work

One-dimensional Bin Packing (BPP) is an NP-hard combinatorial optimization problem with many practical applications in real life. Grouping genetic algorithms (GGAs) are robust tools to optimize this well-known problem when the conventional exact solution methods tend to fail. Although GGAs work well in combination with the problem specific heuristics, they do not make best use of the abilities of the multi-core processors. This paper reports the results of our efforts to increase the solution quality of the GGAs even further by using island parallel GGAs. The characteristics of the best BPP heuristics, evolutionary GGAs, and the parallel computation capabilities are combined to obtain optimal solutions in a scalable way without increasing the communication overhead of the GGAs and achieving increased benefits. The presented parallel hybrid GGAs were successfully applied to a benchmark suite of one-dimensional BPPs. The success rate of the algorithms is superior to most other approaches in the literature. The 1318 benchmark problems are tested from three different sources and our algorithms are shown to be capable of achieving the optimal solutions for 88.54% of the instances within practical optimization times. For future work, we plan to apply the developed algorithms to two/three-dimensional, online, semi-online BPPs, and other real life industrial problems that can be modeled with BPP.

References

Alvim, A. C. F., Ribeiro, C. C., Glover, F., & Aloise, D. J. (2004). A hybrid improvement heuristic for the one-dimensional bin packing problem. *Journal of Heuristics, 10*(2), 205–229.

Anderson, R. J., Mayr, E. W., & Warmuth, M. K. (1989). Parallel approximation algorithms for bin packing. *Information and Computation, 82*(3), 262–277.

Belov, G., Scheithauer, G., & Mukhacheva, E. A. (2007). One-dimensional heuristics adapted for two-dimensional rectangular strip packing. *Journal of the Operational Research Society, 59*(6), 823–832.

Berkey, J. O., & Wang, P. Y. (1994). A systolic-based parallel bin packing algorithm. *IEEE Transactions on Parallel and Distributed Systems, 5*(7), 769–772.

Bestavros, A., Cheatham, T., Jr., & Stefanescu, D. (1990). Parallel bin packing using first fit and k-delayed best-fit heuristics. In *Proceedings of the second IEEE symposium parallel and distributed processing* (pp. 501–504).

Bhatia, A. K., & Basu, S. K. (2004). Packing bins using multi-chromosomal genetic representation and better fit heuristic. *Lecture Notes in Computer Science, 3316*, 181–186.

Bozejko, W., Uchronski, M., & Wodecki, M. (2010). Parallel hybrid metaheuristics for the flexible job shop problem. *Computers & Industrial Engineering, 59*(2), 323–333.

Camacho, E. L., Terashima-Marin, H., Ochoa, G., & Conant-Pablos, S. E. (2013). Understanding the structure of bin packing problems through principal component analysis. *International Journal of Production Economics, 145*(2), 488–499.

Cantu-Paz, E. (2000). *Efficient and accurate parallel genetic algorithms*. Kluwer Academic Publishers.

Caprara, A., & Pferschy, U. (2004). Worst-case analysis of the subset sum algorithm for bin packing. *Operations Research Letters, 32*(2), 159–166.

Caprara, A., & Pferschy, U. (2005). Modified subset sum heuristics for bin packing. *Information Processing Letters, 96*(1), 18–23.

Coffman, E. G., Galambos, G., Martello, S., & Vigo, D. (1999). Bin packing approximation algorithms: Combinatorial analysis. In *Handbook of combinatorial optimization* (pp. 151–207).

Coffman, E. G., Garey, M. R., & Johnson, D. S. (1997). Approximation algorithm for bin packing: A survey. In D. Hochbaum (Ed.), *Approximation algorithm for NP-hard problems* (pp. 46–93). PWS Publishing.

Coleman, N. S., & Wang, P. Y. (1992). An asymptotically optimal parallel bin-packing algorithm. In *Fourth symposium on the frontiers of massively parallel computation* (pp. 515–516). IEEE.

Crainic, T., Perboli, G., Pezzuto, M., & Tadei, R. (2007). New bin packing fast lower bounds. *Computers and Operations Research, 34*(11), 3439–3457.

Dahmani, N., Clautiaux, F., Krichen, S., & Talbi, E. G. (2013). Iterative approaches for solving a multi-objective 2-dimensional vector packing problem. *Computers & Industrial Engineering, 66*(1), 158–170.

Ding, H., El-Keib, A. A., & Smith, R. E. (1992). Optimal clustering of power networks using genetic algorithms TCGA Report No. 92001, March 5. University of Alabama, Tuscaloosa, AL.

Dokeroglu, T., Sert, S. A., & Cinar, M. S. (2014). Evolutionary multiobjective query workload optimization of cloud data warehouses. *The Scientific World Journal, 2014*. Article ID 435254.

Falkenauer, E. (1994). A new representation and operators for GAs applied to grouping problems. *Evolutionary Computation, 2*(2), 123–144.

Falkenauer, E. (1996). A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics, 2*(1), 5–30.

Fernandez, A., Gil, C., Banos, R., & Montoya, M. G. (2013). A parallel multi-objective algorithm for two-dimensional bin packing with rotations and load balancing. *Expert Systems with Applications, 40*(13), 5169–5180.

Fleszar, K. (2012). Three insertion heuristics and a justification improvement heuristic for two-dimensional bin packing with guillotine cuts. *Computers and Operations Research, 40*(1), 463–474.

Fleszar, K., & Charalambous, C. (2011). Average-weight-controlled bin-oriented heuristics for the one-dimensional bin-packing problem. *European Journal of Operational Research, 210*, 176–184.

Fleszar, K., & Hindi, K. S. (2002). New heuristics for one-dimensional bin-packing. *Computers and Operations Research, 29*(7), 821–839.

Friesen, D., & Langston, M. (1991). Analysis of a compound bin packing algorithm. *SIAM Journal on Discrete Mathematics, 4*, 61.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. W.H. Freeman.

Gordon, V. S., & Whitley, D. (1993). Serial and parallel genetic algorithms as function optimizers. In *ICGA* (pp. 177–183).

Gregor, V. L. (1991). Intelligent structural operators for the k-way graph partitioning problem. In *Proceedings of the fourth international conference on genetic algorithms*. San Mateo, CA: Morgan Kaufman Publishers, Inc. University of California, San Diego, July 13–16.

Gupta, J. N. D., & Ho, J. C. (1999). A new heuristic algorithm for the one-dimensional binpacking problem. *Production Planning and Control, 10*(6), 598–603.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press.

Johnson, D. S., Demers, A., Ullman, J. D., Garey, M. R., & Graham, R. L. (1974). Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM Journal on Computing, 3*(4), 299–325.

Khanafer, A., Clautiaux, F., & Talbi, E-G. (2010). New lower bounds for bin packing problems with conflicts. *European Journal of Operational Research, 206*, 281–288.

Kolkman, J. A., & Stemmer, W. P. C. (2001). Directed evolution of proteins by exon shuffling. *Nature Biotechnology, 19*, 423–428.

Kroger, B., Schwenderling, P., & Vornberger, O. (1991). Parallel genetic packing of rectangles. In *Parallel problem solving from nature* (pp. 160–164). Berlin, Heidelberg: Springer.

Le, M. N., Ong, Y.-S., Jin, Y., & Sendhoff, B. (2009). Lamarckian memetic algorithms: Local optimum and connectivity structure analysis. *Memetic Computing, 1*(3), 175–190.

Lim, M. H., Yuan, Y., & Omatu, S. (2000). Efficient genetic algorithms using simple genes exchange local search policy for the quadratic assignment problem. *Computational Optimization and Applications, 15*(3), 249–268.

Loh, K.-H., Golden, B., & Wasil, E. (2008). Solving the one-dimensional bin packing problem with a weight annealing heuristic. *Computers and Operations Research, 35*(7), 2283–2291.

Luque, G., & Alba, E. (2011). *Parallel Genetic Algorithms, Theory and Applications*. Springer.

Martello, S., & Toth, P. (1989). Lower bounds and reduction procedures for the bin packing problem. *Discrete applied mathematics* (Vol. 22, pp. 59–70). North-Holland: Elsevier Science Publishers B.V.

Martello, S., & Toth, P. (1990). *Knapsack problems*. Wiley. Available online on <http://www.or.deis.unibo.it/knapsack.html>.

Mitchell, M. (1996). *An introduction to genetic algorithms*. MIT Press.

Ong, Y.-S., Lim, M.-H., Zhu, N., & Wong, K. W. (2006). Classification of adaptive memetic algorithms: a comparative study. *IEEE Transactions on Systems, Man, and Cybernetics, Part B, 36*(1), 141–152.

Pargas, R. P., & Jain, R. (1993). A parallel stochastic optimization algorithm for solving 2D bin packing problems. In *Artificial intelligence for applications, proceedings, ninth conference* (pp. 18–25).

Poli, R., Woodward, J., & Burke, E. K., (2007). A histogram-matching approach to the evolution of bin-packing strategies. In *IEEE congress on evolutionary computation* (pp. 3500–3507).

Radcliffe, N. (1991). Equivalence class analysis of genetic algorithms. *Complex Systems, 5*, 183–205.

Radcliffe, N., & Surry, P. D. (1995). Fundamental limitations on search algorithms: Evolutionary computing in perspective. *LNCS, 1000*, 275–291.

Raf, V. D., & Robert, P. (1992). Load balancing with genetic algorithms. In *Proceedings of the second conference on parallel problem solving from nature*. Amsterdam, The Netherlands: North-Holland, Elsevier Science Publishers B.V. Brussels, Belgium, September 28–30.

Reeves, C. (1996). Hybrid genetic algorithms for bin-packing and related problems. *Annals of Operations Research, 63*(3), 371–396.

Rohlfshagen, P., & Bullinaria, J. (2007). A genetic algorithm with exon shuffling crossover for hard bin packing problems. *Proceedings of the ninth annual conference on genetic and evolutionary computation*, 1365–1371.

Ross, P., Marin-Blázquez, J. G, Schulenburg, S., & Hart, E. (2003). Learning a procedure that can solve hard bin-packing problems: A new ga-based approach to hyper-heuristics. In E. Cantu-Paz, et al. (Eds.), *GECCO, LNCS* (Vol. 2724, pp. 1295–1306).

Scholl, A., Klein, R., & Jurgens, C. (1997). BISON: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem. *Computers and Operations Research, 24*(7), 627–645.

Schwerin, P., & Wascher, G. (1997). The bin-packing problem: A problem generator and some numerical experiments with FFD packing and MTP. *International Transactions in Operational Research, 4*(5/6), 377–389.

Segura, C., Segredo, E., & Leon, C. (2011). Parallel Island-based multiobjectivised memetic algorithms for a 2D packing problem GECCO, July 12–16, Dublin, Ireland.

Singh, A., & Gupta, A. K. (2007). Two heuristics for the one-dimensional bin-packing problem. *OR Spectrum, 29*(4), 765–781.

Stawowy, A. (2008). Evolutionary based heuristic for bin packing problem. *Computers & Industrial Engineering, 55*, 465–474.

Tosun, U., Dokeroglu, T., & Cosar, A. (2013). A robust Island parallel genetic algorithm for the optimization of quadratic assignment problem. *International Journal of Production Research, 51*(14), 4117–4133.

Valério de Carvalho, J. M. (1999). Exact solution of bin-packing problems using column generation and branch-and-bound. *Annals of Operations Research, 86*, 629–659.

Vanderbeck, F. (1999). Computational study of a column generation algorithm for bin packing and cutting stock problems. *Mathematical Programming, 86*(3), 565–594.

Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation, 3*(4), 257–271.