Available online at www.sciencedirect.com
ScienceDirect
Computers & Industrial Engineering 55 (2008) 465–474
computers & industrial engineering
www.elsevier.com/locate/caie

Evolutionary based heuristic for bin packing problem

Adam Stawowy

Faculty of Management, AGH University of Science and Technology, Gramatyka 10, 30-067 Krakow, Poland

Received 5 April 2007; received in revised form 31 December 2007; accepted 13 January 2008
Available online 18 January 2008

Abstract

In this paper, we investigate the use of evolutionary based heuristic to the one-dimensional bin packing problem (BPP). Unlike other evolutionary heuristics used with optimization problems, a non-specialized and non-hybridized algorithm is proposed and analyzed for solving BPP. The algorithm 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. The set of experiments confirmed that the proposed approach is comparable to much more complicated algorithms.

Keywords: Bin packing; Evolutionary algorithms; Heuristics

1. Introduction

One of the principal rules regarding many human activities is the principle of simplicity. This rule should be taken into account when developing new algorithms as well. Algorithm should do its work and solve its problems in the simplest way possible, following an engineering tradition that puts a high value on simple solutions. Simple algorithms are also more readable, especially for engineers dealing with production problems.

Meanwhile, it can be observed that modern heuristic algorithms for hard combinatorial optimization problems become more and more complicated. In the case of evolutionary algorithms, new crossovers, mutations, representations, and new computing schemes appear. Moreover, in common opinion, those elements should be tailored (specialized) to the specific problem and should utilize problem-specific knowledge. At the same time, the power of simple and non-specialized algorithms that give good (not necessarily optimal) solutions is not fully utilized. The objective of this paper is to present such a simple heuristic for the bin packing problem. In spite of its simplicity the proposed approach performs as well as more complicated algorithms.

This paper continues in Section 2 with an explanation of the BPP. Section 3 gives the details on heuristic approach used to solve this problem. Section 4 presents computational results on published problems. The conclusions are drawn in Section 5.

E-mail address: astawowy@zarz.agh.edu.pl

0360-8352/$ - see front matter
doi:10.1016/j.cie.2008.01.007

2. The bin packing problem

The bin packing problem belongs to the large family of grouping tasks, which consist in dividing sets of elements into separate subsets. The one-dimensional BPP can be formulated as follows:

We have n bins with the capacity of C and n indivisible elements, each having the weight (i.e. value) of $w_i \leqslant C\,(1 \leqslant i \leqslant n)$. The elements have to be packed into the bins in such way that the minimal number of bins should be used and bins capacity should not be exceeded.

Greedy algorithms that give approximate solutions to the bin packing problem were presented by Garey and Johnson (1979). One of them, the First Fit Decreasing (FFD), yields the solutions that guarantees:

$$S_{\mathrm{FFD}} \leqslant \frac{11}{9}\mathrm{OPT} + 4 \tag{1}$$

Martello and Toth (1990) proposed the branch-and-bound method which gives very good results. Scholl et al. (1997) presented BISON – an exact branch-and-bound algorithm which uses new branching scheme with several bounds, reduction procedures and heuristics. Fleszar and Hindi (2002) introduced five new heuristics based on minimum bin slack and variable neighbourhood search. More recently, Alvim et al. (2004) introduced a hybrid improvement procedure based on progressive increase in the number of bins used by a possibly feasible solution. This very specialized algorithm outperforms any known heuristics.

Possible applications of evolutionary algorithms in various combinatorial optimization problems have been extensively studied lately. In relation to the bin packing problem Falkenauer (1994) presented the grouping genetic algorithm (GGA), which is focused on the bins, instead of the objects. The variable length chromosome consists of two parts: the first (object part), of length n, encodes the bin identifiers for each of n objects, while the second (group part) represents the actual bin identifiers used. The important point is that specialized genetic operators work only on the group part. GGA is considered to be the best evolutionary algorithm for the bin packing problem to date.

Khuri et al. (1996) proposed the evolutionary strategy wherein the representation is vector of n integer numbers encoding the bin identifiers; there is also the penalty term as the element of the fitness function. The authors reported very poor performance of their approach: the results obtained by the evolutionary based heuristic were worse than the ones obtained by the First Fit Decreasing algorithm.

Reeves (1996) made use of the hybrid genetic algorithm which hybridized the genetic algorithm for sequencing problem with existing simple heuristics for BPP. The author reported very satisfactory results for proposed approach.

3. Elements of the proposed algorithm

From the work by Falkenauer (1994) it appeared that standard scheme and elements of genetic algorithm are not suitable for the grouping problems. As a remedy, Falkenauer proposed a new encoding scheme and genetic operators adapted to the grouping problem, yielding GGA. These observations were later confirmed in the study by Khuri et al. (1996). Reeves (1996) did not agree with these conclusions and used the standard GA hybridized with simple heuristic originally written for the machine sequencing problem to solve the clustering and partitioning problems. The aim of this paper is to contribute to research by developing a simple, non-specialized and non-hybridized evolutionary based algorithm which is able to perform as well as specialized and/or hybridized heuristics. The similar algorithm was successfully applied to manufacturing cell formation problem (Stawowy, 2006).

3.1. The overall procedure

The variant of Evolutionary Strategy (ES) originally written for a permutation flowshop problem (Stawowy, 1997) is implemented. We have tried to use this idea in our previous work (Stawowy, 1998) but with limited success. Finally, the version presented in this paper – due to smart mutations and problem size reduction – performs very well. The proposed algorithm is variant of $(1, \lambda)$-ES (Biethahn and Nissen, 1995; Schwefel, 1995) where λ children are generated from one parent by means of the simple mutations; the crossover is not employed. The

best of the descendants becomes the new parent solution (deterministic selection). If the fitness value of the new parent is not better than the value of the best solution discovered so far, the *counter* is increased, otherwise the *counter* is reset to zero. If *counter* is equal to empirically determined value Max_C, the worst child from the last generation becomes the new parent. Thereby, the search shifts to a new area in the solution space which helps to escape from local optima. This element performs the role of what Nissen (1994) has called "destabilization procedure" in his combinatorial variant of ES for quadratic assignment problem. Note that the values of λ and Max_C regulate the balance between the exploitation and the exploration of the solution space. It should be emphasized that the concept of shift to the worst solution is effective if this individual is not unacceptably bad, so the mutation has to produce the solutions of reasonable good quality. After obtaining a parent solution some problem size reduction operations are possible; the details are given in Section 3.7.

The general outline of used ES as PASCAL pseudo-code is shown in Fig. 1.

3.2. The encoding

A *permutation with separators* representation introduced by Jones and Beltramo (1991) for a class of partitioning problems is used in proposed algorithm. The solution is represented by a list of n items and l separators of groups (bins); integers from range $\{1,\ldots,n\}$ represent the items, and integers $\{n+1,\ldots,n+l\}$ represent the separators. In contradistinction to Jones and Beltramo representation, separators can appear in the first or last position, and they can appear together (one next to the other).

The encoding mechanism is very simple. For example: for seven items and three separators the solution $S_1 = (1,3,9,8,5,2,7,10,6,4)$ means that the items are partitioned into three bins $(1,3)$, $(5,2,7)$ and $(6,4)$ when the solution $S_2 = (1,10,3,8,5,2,9,7,6,4)$ means that the items are packed into four bins (1), (3), $(5,2)$ and $(7,6,4)$.

Falkenauer (1994) has suggested that such a re-interpretation of the bin packing problem as a permutation problem may run into difficulties. In particular, the author argues that the permutation encoding is highly redundant, as different permutations encode the same solution of the original bin packing problem. We deal with this difficulty by forbidding the items' moves inside the bins. Furthermore, Reeves (1996) gives good practical argument: the permutation approach is not the best one but it has proved to be reliable and effective in a number of instances.

```
BEGIN
  generate and evaluate random starting solution;
  best solution:= start solution;
  counter:=0;
  REPEAT
    FOR i:= 1 to λ DO
    BEGIN
      copy parent to create child number i;
      mutate child i;
      evaluate child i according to fitness function FF;
    END;
    choose the best child based on fitness function value to be a new parent;
    IF FF(new parent) ≤ FF(best solution) THEN
      counter:= counter+1
    ELSE
      counter:=0;
      best solution:= new parent
    END;
    IF counter ≥ Max_C THEN
    BEGIN
      counter:=0;
      choose the worst child based on objective function value to be a new parent
    END;
  UNTIL (max. iterations) or (FF(best solution) = optimum);
  output the best solution;
END.
```

Fig. 1. ES pseudo-code for maximization problems.

3.3. Initial solution

The starting solution (solutions) in evolutionary approach is usually generated in a random way. Although, in general, it is a good idea, it is also a good practice not to introduce unacceptably bad solution (e.g. containing a lot of overhead bins). It is easy to achieve in the case of BPP because many good greedy heuristics exist (Coffman et al., 1996). We have tried BF (Best Fit), FF (First Fit) and FFD (First Fit Decreasing) heuristics to obtain the starting individual: after a set of experiments we have concluded that there are no significant differences between heuristics. Since the FF is the simplest among them, we have decided to generate the initial solution by applying this heuristic to items presented in random order. The heuristic creates feasible individual of reasonable quality.

3.4. The pseudo-genetic operators

Falkenauer (1994) argues that the inadequacy of ordering crossovers for grouping problems is not only backed by the theoretical arguments, but it has been confirmed by experimental evaluation. The author recognizes that the best ordering GA for a grouping problem would be one with *no* crossover. We agree with this observation, thus we rejected crossover operator and left the mutation one as the only operator.

The swap and insertion moves – both typical for permutation problems – are the mutation operators employed. Insertion removes item or separator and puts it in a new place while swap mutation exchanges a pair of items or a pair item-separator. Two essential assumptions make the algorithm powerful:

(1) the items as well as the separators are subject to the moves,
(2) the moves within the bins (groups) are forbidden.

Additionally, the following moves are prevented:

(1) the separators swap,
(2) the same value items swap,
(3) the items or the separators move which leads to bin's capacity overhead.

If generating valid move within, empirically determined, $m = 2 \cdot n/3$ trials is not possible the parent is copying to create the child. The above rules guarantee that only the feasible solutions are generated during the mutation stage.

It is interesting problem to find a good balance between swap and insertion operators. We have conducted several experiments to set probability p_s of applying swap and $p_i = 1 - p_s$ of insertion moves. They have lead us to the following observations:

(1) swap alone performs very poor, it hardly improves the initial solution,
(2) insertion alone is a very good operator, the initial solution improvement is clearly visible,
(3) swap is faster in execution than insertion move,
(4) applying both operators together gives synergy effect: the performance is much better than operators acting alone.

We have decided to set $p_s = 0.9$ and $p_i = 0.1$ as a good compromise between speed and quality. It means that on average $p_s \cdot \lambda$ children are generated by swap moves and the rest by insertion moves at a single step of algorithm.

3.5. The cost function

Falkenauer (1994) argues that using the number of bins as a cost function leads to an extremely unfriendly landscape of search space: all slightly suboptimal points (i.e. one unit above the optimum) yield the same cost value, so capacity of guiding an algorithm in the search is lost. Instead, the author proposed to maximize the following fitness function *FF*:

$$FF = \sum_{i=1}^{nb} \left(\frac{F_i}{C}\right)^k \tag{2}$$

where nb is the number of bins used in the given solution, F_i is the sum of weights of the elements packed into the bin i, $i = 1,\ldots,nb$, k is exponential factor, and C is the bin capacity.

The constant k expresses a concentration on the almost full bins in comparison to less filled ones. Falkenauer (1994) suggested to use $k = 2$ as exponential factor, but after some preliminary experiments we have found out that $k = 4$ gives slightly better results.

3.6. The stopping criterion

The algorithm is terminated when either the optimal solution has been found or after 350,000 function evaluations have been performed. It is comparable to the value applied by Khuri et al. (1000 generations, each yielding 500 solutions) or Falkenauer (134,000 or 335,000 cost function evaluations in relation to problem size).

3.7. Problem size reduction

Although the basic version of the proposed algorithm performs very well, it is not possible, in most cases, to find optimal solutions of Falkenauer instances, which are common benchmark. However, it is possible to further improve the performance of our heuristic by reducing the size of the search space. We have tried to apply three techniques: two known from literature and our own designed specially for used representation.

3.8. Martello and Toth Reduction Procedure (MTRP)

Martello and Toth (1990) proposed MTRP technique that repeatedly finds a bin that dominates all the others, adds that bin to the solution and reduces the problem by removing the items assigned to the bin. Since the heuristic could run in exponential time, the MTPR limits the search to sets of size three or less. That technique was implemented successfully by Falkenauer (1994), Scholl et al. (1997), and Alvim et al. (2004).

3.9. Reeves reduction rule (RRR)

Reeves (1996) proposed to remove from the problem any subset I' of the objects such that

$$\sum_{i \in I'} F_i \geqslant (1 - \varepsilon)C \tag{3}$$

where ε is a small positive number (the author suggested a value of $\varepsilon = 0.00001$). Reeves reported a substantial improvement in the performance of his hybrid algorithm relating to both quality and speed.

3.10. Separators removal (SR)

A lot of redundant separators (i.e. one next to the other) are present in the solutions. It is a good situation at the beginning stages of algorithm, when it has to find good clusters, but at the later stages mutation tries to perform unproductive moves. We decided to remove these separators from the parent solution at the end of single algorithm step so ES can resume on a smaller problem (remember that in our heuristic the separators are subjects to move).

These three approaches were tested with different values of λ and Max_C on Falkenauer *t_249* set. We have confirmed the quality of MTRP and RRR techniques only in part. It is clear that search space is reduced and algorithm finds solutions near the optimum much faster, but at the later stage the promising moves are impossible to perform as the necessary items are packed into deleted bins so optimum can not be found (in all cases *optimum* + 1 bin has been reached). We assume that it is due to a solution encoding and mutation implemented in our algorithm.

The best results are achieved when λ is between 10 and 20, and Max_C is equal to 150. Fig. 2 summarizes the test results for $Max_C = 150$; the data present an average number of cost function evaluations to obtain *optimum* + 1 bin (in this case 84 bins). There is a visible improvement in the performance of ES while size reduction techniques are incorporated: especially effective are RRR and SR, which need four times less evaluations than heuristic with no reduction to obtain the same results for $\lambda = 10$, and twice less for $\lambda = 20$. It should be emphasized that only SR technique allows algorithm to reach optimum solutions, with probability strongly related to population size. For all instances of Falkenauer *t_249* set, the heuristic is able to find the optimum with probability 75.0%, 52.5%, and 27.5% for λ equal to 20, 50, and 100, respectively. If $\lambda = 10$ the optimum solutions cannot be reached although the algorithm with SR performs better than algorithm with MTRP or RRR.

It was decided to incorporate the separators removal approach in the final version of the algorithm.

Fig. 3 presents the typical progress of the solution (in term of max, min and average *FF* of population) over the iteration for the instances of Falkenauer *t_249* set. It can be observed that the average *FF* during succeeding generations is increasing, with direction and pattern corresponding to the fitness of the best individual in population. These results constitute a visible evidence that the algorithm explores intelligently the space of the feasible solutions. Optimization proceeds rapidly, because the distribution of new trials focuses the computational effort on promising regions of the search space.

3.11. Population size

Some further preliminary experiments with other Falkenauer sets convinced us that population size λ is not the constant value for all problems but it should increase when size and hardness of problem increase. We have tried to implement population resizing mechanisms (Eiben et al., 2004; Lobo and Lima, 2006) in our algorithm but none of the evaluated techniques has worked. We have not also managed to discover analytical formula for λ determining, so we can give only some general rules:

(1) population size should be not less than 5, and not grater than 40,
(2) the start value of λ should be set to 15 which performs very well with all problems,

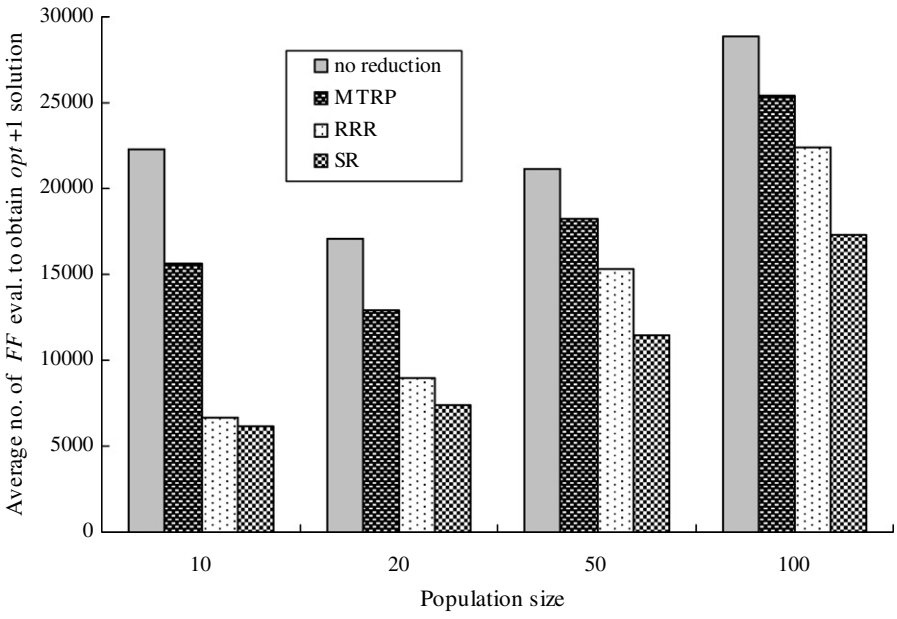

Fig. 2. The performance of size reduction techniques for Falkenauer *t_249* set ($Max_C = 150$).

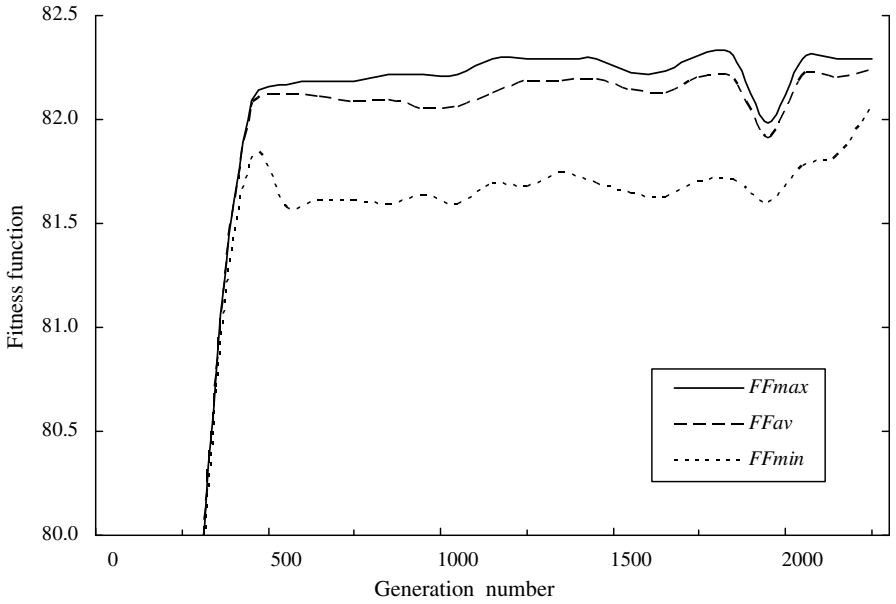

Fig. 3. The typical progress of the solution over the iteration for Falkenauer *t_249* set.

(3) if – after destabilization procedure – the solution comes back fast to local optima, the value of λ should be decreased,
(4) if generated solutions are much worse than the best discovered so far, the value of λ should be increased,
(5) once tuned, population size should be kept at the constant level during algorithm run for problem being considered.

4. Results

The proposed algorithm has been tested on common data sets found in the literature to study its performance as an optimization tool.

We consider five classes of test problems:

– two classes *uniform* and *triplets* introduced by Falkenauer (1994),
– three classes *set_1*, *set_2* and *set_3* from Scholl and Klein (2007).

Falkenauer classes consist of four sets with 20 problems each and can be found in the OR-Library (Beasley, 2007). In case of *uniform* class elements whose weights were drawn at random from the uniform distribution of the interval [20, 100] are to be placed in the bins of the size 150, the number of elements being 120, 250, 500, 1000. In case of *triplets* elements whose values are taken from the interval [0.25, 0.50] are to be placed in the bins of the size 1, the number of elements being 60, 120, 249, 501 – 1/3 of which has considerable weight, while the weigh of the remaining 2/3 is rather small (less than 1/3 of the bin size). Scholl and Klein classes are formed by 720, 480 and 10 instances, respectively, with various bin capacity and various number of elements. The optimal solutions of all these instances are known.

The ES parameters used in these experiments are defined in Section 3. Table 1 summarizes the results of 10 runs for each instance involved in this experiment. For each class we report the number of instances, population size λ, the percentage of runs for which the optimum was found, and mean value and standard deviations of parameters Gen_o and Gen_b. Gen_o is the average number of cost evaluations needed to reach the best solution if optimum was found and Gen_b – if optimum was not found. It is worth to notice that our algorithm always achieved exactly *optimum* + 1 bin in all runs when optimal solutions were not reached. Our heuristic

Table 1
Statistic summary of ES over classes being considered

Class	No. of instances	λ	% opt	Gen_o		Gen_b	
				Mean	*SD*	Mean	*SD*
u_120	20	8	100.0	8,677.0	17,259.7		
u_250	20	10	100.0	19,480.7	27,545.2		
u_500	20	15	93.0	33,850.2	33,669.4	24,489.0	7,674.0
u_1000	20	20	96.5	68,175.0	31,525.5	63,034.3	17,000.3
t_60	20	9	61.0	114,204.9	84,292.5	1,117.3	510.4
t_120	20	13	39.5	170,730.3	87,813.0	3,134.3	1,050.1
t_249	20	20	75.0	176,751.6	85,805.7	9,592.0	2,405.1
t_501	20	30	74.5	182,509.0	76,338.4	25,735.1	6,013.8
set_1	720	8	85.7	26,323.1	52,264.4	71,469.9	79,431.1
set_2	480	9	98.8	11,265.1	22,673.0	16,481.1	17,819.9
set_3	10	7	86.0	31,398.1	63,340.1	6,524.6	2,531.3

% opt, percentage of optimum solutions found in 10 runs for each instance.
Gen_o, the average no. of cost evaluations when optimum was found.
Gen_b, the average no. of cost evaluations to obtain the best solution when optimum was not found.

needs on average 16,658.2 (with standard deviation equal to 29,719.0) cost evaluations to find solutions near the optimal (i.e. one bin above) for all instances.

The proposed heuristic is able to find optimal solutions with a reasonable probability, even for hard classes (*triplets* and *set_3*) the probability of finding optima is still above 60% (except of the *t_120* problem where the probability is only 39.5%). In almost all cases our evolutionary based algorithm with separators removal has determined optimal solutions. The exceptions include 40 instances from *set_1*, and 2 instances from *set_2*. The probability of finding optimal solutions is almost the same for *set_1* and *set_3* (86%) but it is misinterpretation: *set_3* is formed by 10 hard, similar instances while *set_1* consists of 720 instances varying in structure and size. The results for *set_1* would be much better if we could match λ with similar groups in *set_1* class.

The results in Table 1 indicate that our variant of evolutionary strategy is a high quality algorithm that produces good and repetitive results on a suite of test problems of very different size and structure. This algorithm is fully comparable with more complicated approach. For example, Alvim et al. (2004) reported that the probability of finding optimal solutions for five classes being considered with their hybrid improvement heu-

Table 2
The probability of finding optimal solutions

Class	u_120	u_250	u_500	u_1000	t_60	t_120	t_249	t_501	set_1	set_2	set_3
max. iterations											
50,000	95.5	92.0	84.0	31.5	18.0	1.0	4.0	0.0	74.1	95.6	77.0
100,000	99.0	96.5	89.0	85.0	28.0	8.0	21.0	11.0	79.5	97.3	80.0
150,000	100.0	99.5	90.0	91.5	37.5	13.0	38.0	29.5	82.0	98.0	81.0
200,000		99.5	91.0	93.0	48.5	23.0	50.5	47.5	83.4	98.1	83.0
250,000		100.0	91.5	94.0	54.5	29.0	59.0	62.0	84.6	98.3	84.0
300,000			92.5	95.5	59.0	34.5	68.5	68.5	85.2	98.5	85.0
350,000			93.0	96.5	61.0	39.5	75.0	74.5	85.7	98.8	86.0
400,000			94.0	96.5	63.5	41.5	78.0	81.0	86.7	98.8	86.0
450,000			94.0	97.0	64.0	43.5	83.5	84.0	87.6	98.9	87.0
500,000			94.5	98.0	66.0	47.5	87.0	85.5	88.3	99.0	90.0
600,000			95.5	98.5	71.0	56.0	89.0	91.0	89.3	99.1	92.0
700,000			96.5	99.0	74.5	64.0	92.5	94.0	90.3	99.2	92.0
800,000			97.0	99.5	76.0	71.0	94.0	94.5	91.0	99.2	94.0
900,000			97.0	100.0	77.5	75.0	94.5	94.5	91.6	99.3	94.0
1,000,000			97.0		79.5	76.0	96.0	95.0	92.0	99.3	95.0
1,500,000			99.5		90.5	82.0	98.5	98.5	93.6	99.4	97.0

max. iterations, maximum number of evaluated solutions during single algorithm run.

ristic is 98.5% while in the case of our algorithm this percentage is 89.6%. If we allow up to 500,000 cost evaluations (which is still comparable to evolutionary approaches of Falkenauer (1994) and Khuri et al. (1996)), the probability increases to 91.7%.

Further, we have investigated how the values of *max. iterations* influence heuristic performance. Every 50,000 iterations we have noticed the above mentioned parameter; the results are summarized in Table 2. The probability of finding optimal solutions increases from 89.6% to 94.7% while maximum allowed iterations increases from 350,000 to 1,000,000, and to 96,0% if maximum number of iterations is equal to 1,500,000.

As a consequence of implemented representation and mutation, the proposed ES is a simple and effective tool for grouping problems.

5. Final conclusions

The objective of this paper was to present a simple evolutionary based heuristic for the bin packing problem. Our algorithm has several features: the simplicity, non-hybridized approach, effective size reduction and smart mutations. The proposed heuristic is not pure evolutionary strategy; it is rather similar to Nissen (1994) combinatorial variant of ES for quadratic assignment problem. Its probabilistic behaviour consists in:

- generating λ offsprings by randomly based mutations,
- assuming that existing parent do not compete with children when new parent is being selected,
- employing the destabilization procedure in which the worst child becomes the parent to escape the local optima.

Experimental tests described here have confirmed that the proposed evolutionary strategy can be well applied to the bin packing problem and gives only slightly less satisfactory solutions than much more complicated algorithms.

Our ES provides new features to evolutionary algorithms, such as new encoding/decoding mechanism for the *permutation with separators* representation, the concept of separators' movements during mutation, and separators' removal as size reduction procedure. The algorithm has been tested on a set of problems varying in size and structure. Numerical experiments have proved that ES is a fast and high quality optimizer.

The empirical results confirmed once more the power of the evolutionary algorithms which consists in the ability to generate very good solutions without going into the structure of the problem. In case of the grouping problem it means that even simple evolutionary algorithm can perform as well as specialized and/or hybridized heuristics.

Therefore, one will not agree with the opinions by Falkenauer (1994) and Khuri et al. (1996), who stated that the standard permutation representation and genetic operators are inadequate when dealing with grouping problems. It turns out that evolutionary algorithms are very flexible; a slight modification of the standard scheme (whereby for instance, group separators as well as elements will be subjected to operators) gives satisfactory results while tackling various optimization problems.

References

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

Beasley J.E. (2007). OR-Library. Bin packing – One-dimensional. http://people.brunel.ac.uk/~mastjjb/jeb/orlib/binpackinfo.html, last visited on October 2007.

Biethahn, J., & Nissen, V. (1995). *Evolutionary Algorithms in Management Applications*. Berlin, Heidelberg: Springer-Verlag.

Coffman, E. G., Garey, M. R., & Johnson, D. S. (1996). Approximation algorithms for bin-packing: A survey. In D. S. Hochbaum (Ed.), *Approximation Algorithms for NP-hard Problems* (pp. 46–93). Boston: PWS Publishing Company.

Eiben, A. E., Marchiori, E., & Valko, V. A. (2004). Evolutionary algorithms with on-the-fly population size adjustment. In X. Yao et al. (Eds.), *Parallel problem solving from nature* (pp. 41–50). Springer.

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

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

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

Jones, D. R., & Beltramo, M. A. (1991). Solving partitioning problems with genetic algorithms. In R. K. Belew & L. B. Booker (Eds.), *Proceedings of the fourth international conference on genetic algorithms* (pp. 442–449). Morgan Kaufmann.

Khuri, S., Schütz, M., & Heitkötter, J. (1996). Evolutionary heuristics for the bin packing problem. In D. W. Pearson et al. (Eds.), *Proceedings of the second international conference on artificial neural networks and genetic algorithms* (pp. 285–288). Wien: Springer.

Lobo, F. G., & Lima, C. F. (2006). Revisiting evolutionary algorithms with on-the-fly population size adjustment, In *Proceedings of the eighth annual conference on genetic and evolutionary computation* (pp. 1241–1248). New York: ACM Press.

Martello, S., & Toth, P. (1990). Lower bounds and reduction procedures for the bin packing problem. *Discrete Applied Mathematics, 28*, 59–70.

Nissen, V. (1994). Solving the quadratic assignment problem with clues from nature. *IEEE Transaction on Neural Networks, 1*, 66–72.

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

Scholl, A., & Klein, R. (2007). Bin packing. On line document at http://www.wiwi.uni-jena.de/Entscheidung/binpp/, last visited on October 2007.

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

Schwefel, H. P. (1995). *Evolution and optimum seeking*. New York: John Wiley & Sons.

Stawowy, A. (1997). The polynomial algorithms for the flowshop sequencing problem. Ph.D. thesis (in Polish). Krakow: AGH University of Science and Technology.

Stawowy, A. (1998). Evolutionary algorithm for bin packing problem. *Mechanics, 17*(4), 593–598, in Polish.

Stawowy, A. (2006). Evolutionary algorithm for manufacturing cell design. *Omega, International Journal of Management Science, 34*, 1–18.