Heuristics methods for solving the block packing problem

Viktor M. Kureychik
Computer sciences and information safety institute
Southern Federal University
Taganrog, Russia
kur@tgn.sfedu.ru

Vladimir Vl. Kureychik
Computer sciences and information safety institute
Southern Federal University
Taganrog, Russia
kureichik@yandex.ru

Roman Potarusov
Southern Federal University
Taganrog, Russia
potarusov.roman@gmail.com

Liliya Kureychik
Computer sciences and information safety institute
Southern Federal University
Taganrog, Russia
kur@tgn.sfedu.ru

***Abstract* – In the given paper one-dimensional Bin Packing Problem which plays an important role for the optimization of transportations and production activities is considered. The Hybrid Genetic Algorithm for one-dimensional Bin Packing Problem is proposed. For this purpose two evolution models (de Vries' evolution model and Lamarck's evolution model) have been adapted. Besides, new problem-oriented genetic operators are developed. The main advantage of the suggested approach is that it never decreases the quality of solution so it allows obtaining valid Bin Packing Problem solutions. Two effective local search algorithms allowing to improve of Bin Packing Problem solutions by getting quasi-optimal and optimal packings are proposed. Computational experiments show that a new hybrid approach based on genetic algorithm intended for solving one-dimensional BPP provides approximation and optimal solutions for all benchmarks in-stances in a tolerable computational time as well as demonstrate the robustness of the proposed approach.**

Keyword – optimization problem, bin packing, genetic algorithms, local search

I. INTRODUCTION

Bin Packing Problem (BPP) plays an important role for the optimization of transportations and production activities as well as the development of new information and computer technologies, etc. The problem of BPP is a complex combinatorial logic and NP-complete task. The goal of this task is to arrange the elements in blocks with a given dimension (capacity), in such a way as to maximize the filling of blocks and to minimize the total number of blocks. Bin Packing Problem refers to a common scientific and industrial field. Despite on the existence of a large number of different approaches, methods and algorithms of packing, the optimal solutions are not found. Currently, there is no universal algorithm be capable for solving all test tasks with similar efficiency. Therefore, the development of new effective heuristic approaches, methods and algorithms is an urgent and important issue.

To our extend there are some heuristic approaches applied to the problem of Bin Packing Problem. Gupta and Ho [1] proposed the bin-oriented heuristic, which they called Minimum Bin Slack (MBS). This heuristic is different from item-oriented heuristics in that the packing process is carried out bin-by-bin rather than item-by-item, using a procedure called MBS-One-Packing. At each iteration, instead of packing the items one by one based on certain rules, the MBS-One-Packing procedure searches for a group of items (from all the unpacked items) that could fill a bin with minimal slack (i.e. the smallest residual capacity) and packs this group of items into a new bin. The MBS heuristic repeatedly calls MBS-One-Packing procedure until all items are packed. A recursive version of MBS-One-Packing was illustrated in [2]. The procedure stops when either a combination with zero slack (i.e. no residual capacity left) is found or all item combinations have been explored.

MBS' heuristic. Fleszar and Hindi [2] presented a variant of this procedure (denoted by MBS'), which always packs the first item of the vector into the current packing. The modified algorithm gives similar solution quality but shorter computation time in most instances. The MBS heuristic has shown to be superior to FFD, BFD, B2F and FFD-B2F in terms of the solution quality and is able to solve the problem to optimality where the optimal solution is two bins. It is especially efficient when the optimal solution requires the majority of bins to be fully filled. The worst time complexity of this algorithm is 2n, where n is the number of items. However, the experimental results have shown it to be very efficient for most problem instances ([1], [2]).

One more approach for solving BPP is based on variable neighborhood search algorithm. The main components of this approach are moves. By "move" we define the transition of an item from its current bin to another one or the swap of a pair of items across their respective current bins. Moves that do not violate the bin capacity constraint are called "valid moves". Further only valid moves are considered [2]. The essential stage of any neighborhood search process is the choice of the

objective function. According to our problem, we have to choose such an objective function that allows minimizing the number of bins. The objective function is rather pointless since because of a lot of different configurations, in terms of the assignment of items to bins, corresponding to the same number of bins. Considering that a good solution will always have nearly full bins leads to an objective function that seeks configurations having this feature. Let's consider the following objective function

$$\max f(x) = \sum_{\alpha=1}^{m} (l(\alpha))^2 \quad (1)$$

where m is a number of bins in x and l(α) is a sum of sizes of items Aα assigned to bin α, i.e.,

$$l(\alpha) = \sum_{i \in A_\alpha} t_i \quad (2)$$

Besides the maximization of bin loads, the objective function tries to reduce the number of bins. So, it is obvious that the value of the function will not change if an empty bin is added or removed [2].

An approach of Hybrid Ant Colony Optimization algorithm for BPP, proposed by John Levine and Frederick Ducatelle [4] combines the ACO meta-heuristic with a simple but effective iterated local search algorithm based on the Dominance Criterion of Martello and Toth [5]. Starting with a set of empty bins, fill each bin in turn by repeatedly placing the largest item from those remaining which will still fit into the bin. If no items left are small enough to fit into the bin, a new bin is started. This procedure results in the FFD solution, but is more useful for purposes of ACO: it shows that the heuristic favourability of an item is directly related to its size [6].

Building the solution. Every ant starts with the set of all items to be placed and an empty bin. It will add the items one by one to its bin, until none of the items left are light enough to fit in the bin. Then the bin is closed, and a new one is started.

Iterated local search. In every ant's solution, the n least full bins are opened and their contents are made free. Items in the remaining bins are replaced by larger free items. This gives fuller bins with larger items and smaller free items to reinsert. The free items are reinserted via FFD. The procedure is repeated until no further improvement is possible. Only the global best ant increases the pheromone trail.

International studies in the field of computer science and genetics allow obtaining new information technology related to the use of nature evolution models. It is therefore proposed to solve BPP based on bioinspired methods including evolutionary, genetic and swarm approaches. The main feature of genetic algorithms is their possibility of working not with one solution, but a set of alternative solutions. But in the case of too large search space bioinspired algorithms have drawbacks related with pure convergence in the global optimum with an adequate precision. Local search techniques follow a different strategy, being able to find any local optimum with great precision, using information from the neighboring candidate solutions. Advantages and drawbacks of genetics algorithms and local optimization approaches are converse. The synergy between them leads to appearance of new hybrid methods, simultaneously global and precise. The direct implementation of this idea is to apply a genetic search and a local technique in two consecutive steps. Actually, the genetic algorithm explores the domain and finds a good set of initial estimates. On the second step in order to locate the nearest and best solution, the obtained estimates are refined by the local technique. In this paper we propose a new hybrid approach based on genetic algorithm intended for solving one-dimensional BPP providing approximation and optimal solutions for all benchmarks instances in a tolerable computational time [7].

II. SIMULATED ANNEALING HYPER-HEURISTICS FOR A MAXIMISATION PROBLEM

Simulated annealing is a local search method inspired by Metropolis et al.'s algorithm to simulate the physical cooling process [8]. Since its introduction as an optimization tool [3], SA has been intensively studied both in theory and application. The theoretical analysis of SA have been concerned with its convergence criteria, based on the fact that simulated annealing can be treated as a series of homogeneous Markov chains or a single nonhomogeneous Markov chain. Research has proven that SA is able to asymptotically converge to an optimal solution if certain conditions are satisfied [3]. However, these theories are not very useful in practice because guaranteeing an optimal solution often requires more iterations than an exhaustive search.

The procedure for simulated annealing is fairly simple. For a maximization problem with objective function f and neighborhood structure N, SA starts from an initial solution and repeatedly generates and transfers to a neighbor of the current solution. During this process, SA has the possibility of visiting worse neighbors in order to escape from local optima. Specifically, a parameter, called temperature t, is used to control the possibility of moving to worse neighbor solutions. The algorithm, starting from a high temperature, repeatedly decreases the temperature in a strategic manner (usually referred to as a cooling schedule) until the temperature is low enough, or some other stopping criteria are satisfied. In each iteration, the algorithm accepts all uphill (a move which increases the objective value for a maximization problem) moves and some of the downhill (a decrease in the objective value for a maximization problem) moves according to the Metropolis probability, defined by exp(δ/t) where δ is the difference in the objective function between the new candidate solution and the current solution. A general simulated annealing algorithm for maximization problem can be described by the following [1]:

```
Repeat
    Repeat
            Randomly select s  N (s0);
            δ = f(s) − f(s0);
            If δ > 0 s0 = s;
            Else if exp(δ/t) > random(0,1) s0 = s;
            Endif
            If f(s0) > f(sbest) sbest = s0;
            Endif
    Until iteration_count = nrep
```

Set t = $\phi(t)$;
Until the stopping conditions are met
Output sbest as the best solution found.

Two important factors have to be carefully considered before implementing this general simulated annealing algorithm. These are the definition of neighborhood structure N and the cooling schedule which is determined by

1) a starting temperature ts;

2) temperature reduction function α(t);

3) the number of iterations at each temperature nrep and

4) stopping condition(s).

The starting temperature. The initial temperature should be high enough to allow "free moves" at the initial state such that the final solution is not dependent on the initial state [3]. However, if one wants SA to start from a good quality solution created by some sophisticated heuristics, the initial temperature should not be too high. This is due to the fact that, when the temperature is too high, the algorithm accepts almost all of downhill moves (without specification, it is assumed that we are trying to maximize the objective function). In this case, the search, in fact, starts from a random initial solution. The effort of obtaining a high quality initial solution is, therefore, irrelevant.

A lot of research has been carried out in order to identify an optimal initial temperature or a method by which an initial temperature can be determined.

In [9] suggested an initial temperature $t_0 = \delta max$ where δmax is the maximal difference in the objective value between two neighboring solutions. Another more intuitive method is setting an initial temperature value such that the ratio of accepted downhill moves to all neighborhood moves is equal to a predefined value. Besides the authors suggested using the average cost difference of a set of sample neighboring solutions to approximate the initial temperature of a given acceptance ratio of downhill moves. Suppose δ represents the average cost difference of a set of sampled neighboring solutions and r0 is the given acceptance ratio allowed at the beginning of the search, the initial temperature can be calculated by $t0 = -\delta avg/\ln(r0)$.

Cooling schedule. Considerable research has been carried out in the pursuit of a good cooling strategy. Two of the most popular methods are geometric cooling $\phi(t) = \alpha t$ ($\alpha < 1$) and a non-linear cooling function $\phi(t) = t/(1 + \beta t)$ (where β is a very small positive value) [10]. In the geometric cooling function, the temperature reduction rate is a constant and usually takes value in the range of [0.8, 0.99]. However, in the Lundy and Mees' cooling function, the temperature drops very quickly when the temperature is high and relatively slower when the temperature is low. At each temperature only one iteration is executed. Both cooling schedules are monotonic de-creasing functions. However, an optimal cooling schedule may be not monotonic and be dependent on different problems [11]. Therefore, several other cooling strategies have also been proposed which take into account the history of the search and allow temperature increases during the search.

When the temperature becomes very low, SA degenerates into a hill climbing algorithm and most of the time is being wasted in generating and rejecting inferior solutions.

Instead, the temperature could be held constant throughout the search. He tested the idea on quadratic assignment problems and concluded that there exists a fixed temperature at which the performance is optimized. However, this optimal temperature might be different from problem to problem and is very difficult to obtain.

Stopping condition(s). The conventional simulated annealing algorithm stops when the temperature reaches zero or a value small enough such that the algorithm converges to a local optimum. Choosing an appropriate value of the stopping temperature can be based on experiments or one can monitor the acceptance ratio of downhill moves and the algorithm stops when the ratio decreases below a given very small value (0.01 for example). Other stopping conditions were also used, such as the allowed computation time, the number of consecutive non-improvement moves, etc.

Figure 1 shows a general framework for the simulated annealing hyper-heuristic proposed by Ruibin Bai [11].

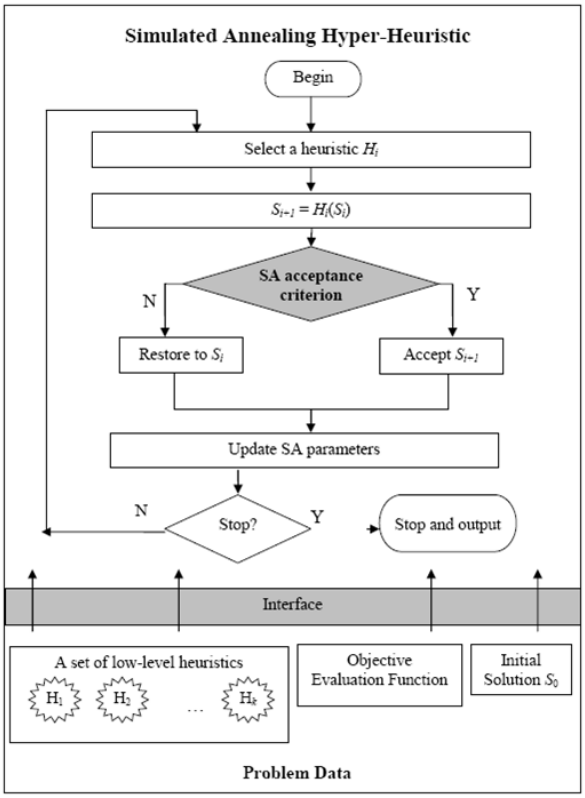

Fig. 1. The framework of simulated annealing hyper-heuristics for a maximization problem

The system is very similar to other forms of hyper-heuristics except that a simulated annealing algorithm is used as an acceptance criterion. At each iteration, the algorithm selects a heuristic from the set of low-level heuristics available

and applies it to the current solution. If the solution generated by this heuristic is better than the current solution, it is accepted. Otherwise, it is accepted according to a Metropolis probability. The temperature of the simulated annealing is then modified. When the stopping conditions are met, the system terminates and outputs the best solution found so far. Note that the proposed hyper-heuristic does not conflict with the existing local search hyper-heuristics. The selection of the heuristics could be in a random way or by utilizing some intelligence that has been proposed in other hyper-heuristic frameworks. In his thesis, he requires all of the solutions generated by the low level heuristics to be feasible, i.e. the low level heuristics searches in the feasible solution space [11].

Firstly, the simulated annealing hyper-heuristic biases the exploration of the neighborhood by heuristically sampling the candidate solutions (using different low-level heuristics) rather than sampling them uniformly from the given neighborhood as does a traditional simulated annealing algorithm with multi-neighborhoods. Secondly, a simulated annealing algorithm requires every state to be reachable (i.e. any solution can be reached by any other solution after a finite number of iterations of moves in the defined neighborhood) [11]. However, in the simulated annealing hyper-heuristic, the low-level heuristics do not necessarily satisfy this requirement as long as there is a combination of these heuristics that can make each solution reachable. This is very useful when we have several possible heuristics or operators that can transfer the state of the current solution but these operators alone are not able to generate a neighborhood that satisfies the reachability condition. For example, when dealing with the bin packing problem, a neighbor solution can be created by "interchanging two (random) items of two (random) bins". However, using this heuristic alone cannot guarantee to reach every other solution (from the current solution) within a limited number of executions. Meanwhile, although the neighborhood defined by another operator "shifting a random item from one random bin to another random bin" satisfies the reachability condition, the local search algorithms using this operator alone generally perform very badly.

III. Applying SA hyper-heuristic for solving the BPP

The difficulty of applying SA to bin packing probably comes from the fact that the objective function of the bin packing problem (the number of the bins occupied) is not sensitive to the general neighborhood moves [12]. For instance, two general neighborhood moves, interchanging two items between two bins and shifting an item from one bin to another, are usually not able to change the objective function. Employing some elaborate moves could damage the reachability of the neighborhood and prematurely lead the search into a local optimum. In fact, improving the bin packing solution really needs the cooperation of several types of neighborhood moves rather than executing single type of moves. However, a general simulated annealing algorithm only allows a single neighborhood structure with the requirement of reachability, which handicaps the application of SA in problems such as bin packing. Therefore, in this application R. Bai propose to utilize several heuristics under the framework of the simulated annealing hyper-heuristics, rather than just using a single neighborhood.

The heuristics used to transfer the state of the solution do not necessarily satisfy the reachability requirement as required by a neighborhood definition. The heuristics used could have some intelligent elements such that the cur-rent solution is transferred to the promising directions rather than being purely random.

Low-level heuristics. The implementation of the simulated annealing hyper heuristic requires a set of problem-specific low-level heuristics. A total of five low-level heuristics are used, as follows [12]:

H1: Exchange largest Bin_largest Item. This heuristic selects the largest item from the bin with the largest residual capacity and exchanges this item with another smaller item (or several items whose capacity sum is smaller) from another randomly selected non-fully-filled bin. The idea behind this heuristic is to transfer smaller residual capacity from a random bin to a bin with the largest residual capacity such that this bin can be emptied by other heuristic(s).

H2: Exchange random Bin_largest Item. This heuristic is similar to H1 except that the exchange is carried out between two randomly selected non-fully filled bins.

H3: Shift. This heuristic selects each item from the bin with the largest residual capacity and tries to shift them to the rest of the bins using the BFD heuristic.

H4: Split. H1, H2 and H3 all operate on non-fully-filled bins. However, in some cases, a fully-filled bin may contain too many small items such that it is impossible to transfer to the optimal solution using H1, H2 and H3 because of the difficulty in packing large items. Hence this heuristic is designed to solve this problem. Once the number of the items in a bin is found to have exceeded the average number of items of other bins, this heuristic transfers half of the items, selected at random, into a new bin.

H5: Best Packing. This heuristic firstly selects the biggest item from a probabilistically selected bin. The TBR_MBS heuristic is then used to search a group of items (called one packing) that contains this item and considers all the other items (the sequence of these items in the vector Π is sorted by the residual capacity of the corresponding bins with tie broken arbitrarily). All the items that appeared in the packing found by TBR_MBS are then transferred into a new bin. The time limit is set to 0.2 second based on preliminary experiments. The probability of selecting a bin is calculated by

$$\psi = \text{resCapj} / \sum \text{resCapj}, \quad (3)$$

where resCapj is the residual capacity of the bin j. Hence the selection is in favour of the bins with the larger residual capacity. The bins with zero residual capacity will not be selected because they are already packed well [12].

Note that heuristics H1 and H2 are normally not able to change the objective, while heuristic H3 is an objective improving heuristic and heuristic H4 is objective non-improving. Heuristic H5 could undermine or improve the objective.

IV. Implementation of BPP based on SA hyper-heuristic

Let's demonstrate results of implementation of BPP based on SA hyper-heuristic. These data sets consist of two classes of problems: uniform and triplet. In the uniform class, the number of items is 120, 250, 500 and 1000 respectively and their sizes are uniformly distributed in the range of [20,100]. The bin capacity is 150. There are 20 instances for each problem size and hence 80 problem instances in total. In the triplet class, the bin capacity is 1000 and the item sizes are deliberately generated such that, in the optimal solution, every bin contains exactly three items (one "big" and two "small" items) without any residual capacity. The number of the items is 60, 120, 249 and 501 and each of them contains 20 instances. This class of data set is claimed to be more difficult because of the fact that no residual capacity is allowed in any bin in the optimal solution [12].

Table 1. Computational results of MBS based heuristics and hyper-heuristics

		MBS	TBR_MBS	SAHH
Data sets	#num	#opt	#opt	#opt
FAL_U120	20	11	17	20.0
FAL_U250	20	12	13	18.6
FAL_U500	20	11	11	19.0
FAL_U1000	20	7	5	20.0
FAL_T60	20	0	0	20.0
FAL_T120	20	0	0	20.0
FAL_T249	20	0	0	20.0
FAL_T501	20	0	0	20.0

Table 2. Computational results of MBS based heuristics and hyper-heuristics

		HGGA	VNS	SAHH
Data sets	#num	#opt	#opt	#opt
FAL_U120	20	18	20	20.0
FAL_U250	20	18	19	18.6
FAL_U500	20	20	20	19.0
FAL_U1000	20	20	20	20.0
FAL_T60	20	18	20	20.0
FAL_T120	20	20	20	20.0
FAL_T249	20	20	20	20.0
FAL_T501	20	20	20	20.0

In the tables:

- #num is the number of instances in the given data sets.
- #opt is the number of instances for which the given algorithm finds a solution.

The Hybrid Parallel Genetic Algorithm to solve the BPP has been presented. New specific genetic operators are developed with two effective replacement procedures. The realized computational experiments establish that the presented HGA never decrease the quality of existing solutions in the literature. Moreover, new search strategy has been designed. The main characteristic of this new strategy is parallelization of genetic search, hybridized by effective local search.

Two evolution models (de Vries' evolution model and Lamarck's evolution model) have been adapted to solve the BPP. New problem-oriented genetic operators have also been developed. They never decrease the quality of solution and allow obtaining valid BPP solutions. Two effective local search algorithms are proposed. They allow improving of BPP solutions to get quasi-optimal and optimal packings.

Computational experiments show that the presented algorithm gives quasi-optimal and optimal solutions for all benchmark instances in an acceptable amount of computing time, clearly showing the robustness of the proposed approach. In the case of quasi-optimal solutions the absolute deviation from reference solution is at most one bin. Efficiency of HGA is due to new hybridization mechanisms of genetic algorithms and local search procedures.

Future work could explore the possibility of designing more sophisticated architectures of genetic search with migration and applying the proposed approach to solve the Vehicle Routing Problem with Multiple Routes. BPP approach seems to be effective to distribute routes to vehicles.

Acknowledgment

This work is supported by the Grant of Russian Scientific Foundation (project №14-11-00242) in the Southern Federal University.

References

[1] Gupta, J. N. D. and Ho, J. C., 1999. A New Heuristic Algorithm for the Onedimensional Bin-packing Problem, Production Planning & Control. 10, 598-603.

[2] Fleszar, K. and Hindi, K. S., 2002. New Heuristics for One-dimensional Bin-packing, Computers & Operations Research. 29, 821-839.

[3] Ruibin Bai. An investigation of novel approaches for optimizing retail shelf space allocation. PhD Thesis. The University of Nottingham, Nottingham, UK, 2005.

[4] John Levine and Frederick Ducatelle. Ant Colony Optimization and Local Search for Bin Packing and Cutting Stock Problems. - Centre for Intelligent Systems and their Applications, School of Informatics, University of Edinburgh, 2003.

[5] Knapsack problems, algorithms and computer implementations : Silvano Martello and Paolo Toth Wiley, Chichester, England, 1990.

[6] Dorigo, M., Vittorio, M. and Alberto, C. Optimization by a Colony of Cooperating Agents, IEEE Transactions on Systems, Man and Cybernetics – Part B: Cybernetics. 26, 1996. 26-41 p.

[7] Potarusov, R., Kureychik, V., Goncalves, G. and Allaoui H. Solving the Bin Packing Problem with Algorithm of Genetic Search with Migration. Proceedings of International Conference on Artificial Intelligence Systems (AIS'07), Divnomorskoe, Russia, September 3-10, 2007. – 34-45 p.

[8] Petch, R.J. and Salhi, S. A multi-phase constructive heuristic for the vehicle routing problem with multiple trips. Discrete Applied Mathematics, 133, 2004. – 69-92 p.

[9] Alvim Adriana C.F., Glover Fred S., Ribeiro Celso C., and Aloise Dario J. Local search for the bin packing problem. Journal of heuristics, 10, 2004. – 205-229 p.

[10] Lourenco, H.R., Martin, O. C., and Stutzle, T. Iterated local search. In: Glover, F. and Kochenberg, G.A. (Eds.), Handbook of Metaheuristics, Kluwer, 2003. – 321-354 p.

[11] Blum, C. and Roli, A. Metaheuristic in Combinatorial Optimization: Overview and Conceptual Comparison, ACM Computing Surveys. 35, 2003. – 268-303 p.

[12] Kureychik, V.M., Potarusov, R.V., Goncalves, G. Bionicheskie metodi upakovri blokov. – Taganrog: Izd-vo TTI UFU, 2009. – 120p.

[13] Gladkov, L.A, Kureychik, V.V., Kureychik, V.M.: Genetic algorithms, 2nd. edn. Fizmatlit, Moscow (2006)