An Analogy between Bin Packing Problem and Permutation Problem: A New Encoding Scheme

Michel Gourgand, Nathalie Grangeon, Nathalie Klement

An analogy between bin packing problem and permutation problem: a new encoding scheme

Michel Gourgand, Nathalie Grangeon, and Nathalie Klement

LIMOS CNRS UMR 6158, Université Blaise Pascal,
Complexe scientifique des Cézeaux, 63173 Aubière Cedex, France
gourgand@isima.fr, grangeon@isima.fr, klement@isima.fr

Abstract. The bin packing problem aims to pack a set of items in a minimum number of bins, with respect to the size of the items and capacity of the bins. This is an NP-hard problem. Several approach methods have been developed to solve this problem. In this paper, we propose a new encoding scheme which is used in a hybrid resolution: a metaheuristic is matched with a list algorithm (Next Fit, First Fit, Best Fit) to solve the bin packing problem. Any metaheuristic can be used but in this paper, our proposition is implemented on a single solution based metaheuristic (stochastic descent, simulated annealing, kangaroo algorithm). This hybrid method is tested on literature instances to ensure its good results.

1 Introduction

The bin packing problem has been introduced by [1]. It considers a set of N items, each item with a given size w_i, and several bins with a same capacity C. The aim of this problem is to pack all of the items in a minimum number of bins. The sum of the size of the items packed in a bin has to be smaller than the capacity of the bin. Each item has to be packed in one bin.

This problem can be met in industrial application or computer network design and memory allocation [2]. [3] makes a state-of-the-art review about the container loading problem: the bin packing problem can be used to pack a set of cargo into a minimum number of containers. It can also be used to solve assembly line balancing problems [4] or multiprocessor scheduling problems [5].

In [6] we use the bin packing problem to model the problem of activities planning and resources assignment, called the HCT problem. The problem considers a system composed of resources and of a set of activities to assign. Each activity needs a resource to be treated and a time slot, a period when it will be done. Each activity has a known process time. Each activity starts in one period and finishes in the same period. Resources have a planning defining their available time: the resource opentime. Each resource cannot treat all the activities, there is a list of incompatibilities between activities and resources.

The aim is to assign each activity to one resource and one period. Activities have to be done as soon as possible, so the assigned periods have to be the smallest possible. The aim of the HCT problem is not to make a precise schedule of the activities but to assign a period to each activity.

The HCT problem has to respect some constraints: resources have to be able to process their assigned activities according to the incompatibilities; activities have to be assigned to a resource during the opentime of this resource; each activity has to be assigned to one resource and one period. Figure 1 gives an example of an assignment in such an application. A couple (resource, period) can be seen as a bin. Activities are assigned to resources during periods. All the resources have the same opentime.

Table 1 summarizes the analogies between the bin packing problem and this application. A new constraint in the HCT application is the incompatibility constraint between resources and activities. The HCT problem can be seen as an extension of the bin packing problem with incompatibility constraints.

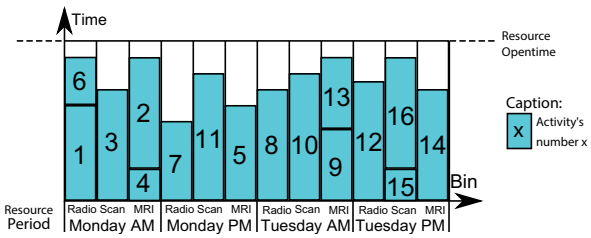


Fig. 1. Example of the HCT application

Table 1. Analogies between the bin packing problem and the HCT problem.

	Bin packing problem	**Problem of activities planning and resources assignment**
Data	Item	Activity
	Bin	Couple (resource, period)
	Size of an item	Process time of an activity
Objective	To assign items in bins	To assign activities to periods and resources
Constraints	Capacity of the bins	Resources opentime
	-	Incompatibility constraint
Criterion	To minimize the number of used bins	To minimize the number of used couples (resource, period)

In Section 2, a brief state of the art about approximate algorithms used to solve the bin packing problem is made. In this paper, we propose an encoding scheme, which is used in the hybridization between a metaheuristic and a list algorithm. The used method is described in Section 3, as its theoretical justification. The experimental results on literature instances are presented in Section 4. This paper is ended by a conclusion and some further works.

2 State of the art

Given the size of the real problems that can be solved thanks to the bin packing problem, exact methods quickly reach their limit about computational time. Moreover, [7] shows that it is an NP-Complete decision problem. Approximate algorithms seem to be a good way to solve this problem.

The most popular list algorithms have been developed by [1]. Items are considered one by one according to a list. **Next Fit** (NF) is the most intuitive method for the bin packing problem. The maximum number of items is packed into the current bin. If the available space in the current bin is smaller than the size of the considered item, this bin is closed and a new bin is opened and becomes the new current bin. **First Fit** (FF) is different from NF in that none of the bins is closed. Each item is packed into the first bin which can contain it. Once an item cannot fit in any bin, a new bin is opened. **Best Fit** (BF) consists in packing each item into the best bin which is the one with the smallest available space after packing the considered item into it.

The use of metaheuristics to solve the bin packing problem starts in the 1990s. In most published works, a classical encoding scheme S is used where $S(i)$ is the index of the bin where item i is placed. [8] proposes a hybrid metaheuristic based on tabu search. [9] proposes a hybrid grouping genetic algorithm which uses the heuristic FFD [1] and the dominance criterion from [10]. To a better use of grouping genetic algorithm, it uses the classical encoding, augmented with a group part which defines the used bins. [11] applies a new approach method: the weight annealing. It uses the classical encoding scheme. [12] uses a hybrid ant colony optimization, inspired by Falkenauer's works. It does not individualize the items, they are designated by their size and not their index: $S = (w_i)_{i \in N}$ with w_i the size of item i. In these papers, the hybrid part consists in matching a known metaheuristic (like genetic algorithm, tabu search method, ant colony optimization) to an additional improvement method such a local search.

Another encoding scheme has been exploited. [13] uses a permutation coding to hybridize the genetic algorithm. [14] uses a permutation with separators representation: it is a list of n items and l separators of bins, integers from range $\{1, ..., n\}$ represent the item, integers $\{n+1, ..., n+l\}$ represent the separators. A simplified version of the genetic algorithm is developed: a simple evolutionary based heuristic.

Most of the works on the bin packing problem uses population based metaheuristics.

3 Proposed hybrid method

The method proposed in this paper uses a hybridization between a metaheuristic and a list algorithm (Next Fit, First Fit, Best Fit). This method is performing in two search spaces: Ω where a solution represents a list of items and S where a solution represents the assignment of each item to a bin. The aim of this purpose is to reduce the size of the set of solutions where the metaheuristic is performing.

3.1 Encoding of the set of solutions

The used encoding is inspired by the one used in the permutation problems. A solution $(X_i)_{i \in N} \in \Omega$ is a list of items as in a permutation problem with X_i an item index. A list algorithm L, such as NF, FF or BF, is applied to the list of items X to determine the assignment of the items to the bins: $(Y_i)_{i \in N} \in S$ as the classical scheme, with Y_i the bin index assigned to item i. Then a cost function H is applied to the assignment Y. The general scheme of the encoding is the following:

$$X \in \Omega \underset{Heuristic\ L}{\longrightarrow} L(X) = Y \in S \underset{Criteria\ H}{\longrightarrow} H(Y)$$

Figure 2 summarizes the considered sets of solutions.

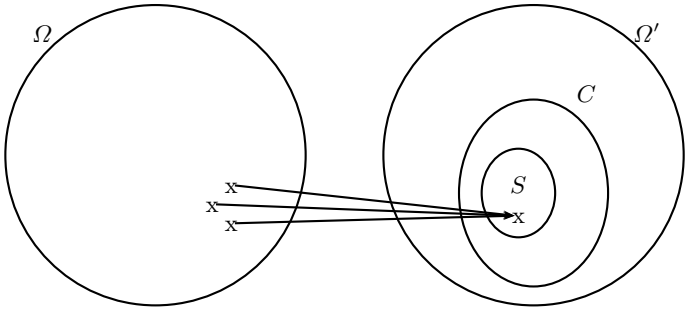
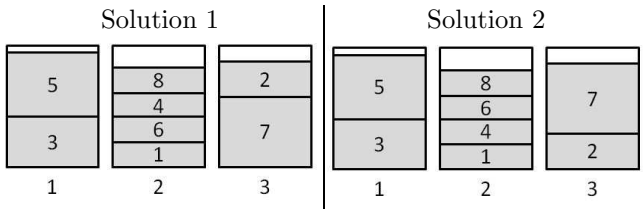


Fig. 2. Sets of solutions

- Ω is the set of all the lists of items. card(Ω)= $N!$
- Ω' is the set of all the possible assignment of items to bins, without checking the capacity constraint. card(Ω')= N^N
- C is the set of the admissible solutions: a solution is admissible if the capacity constraint is respected. $C \subseteq \Omega'$.
- S is the set of all the solutions built by the application of a list algorithm on a list of items. $S \subseteq \Omega'$ and $S \subseteq C$.

Proof that S contains the set of optimal solutions: Let be Y^* any optimal solution. $Y^* = (Y_1^*, ..., Y_N^*)$ with Y_i^* the bin index assigned to item i. A list of items assigned in each bin is deduced. By concatenation of these lists, an ordered list of all the items is determined: $X^* \in \Omega$. By applying the heuristic FF or NF to X^*, Y^* is found, so $Y^* \in S$. As a consequence the set of the optimal solutions written C^* is included in S.

The following example considers the assignment of eight items. Two different solutions in Ω give the same element in S. In total, $2 \times 4! \times 2 = 96$ elements in Ω give the same element in S.



$X \in \Omega =$ $(3, 5, 1, 6, 4, 8, 7, 2)$ $(3, 5, 1, 4, 6, 8, 2, 7)$
$Y \in S =$ $(2, 3, 1, 2, 1, 2, 3, 2)$ $(2, 3, 1, 2, 1, 2, 3, 2)$

Let be S_B the set of solutions $\in \Omega'$ built by the heuristic BF, S_F the set built by FF and S_N the set built by NF. We have the following assumption: $C^* \subseteq S_B \subseteq S_F \subseteq S_N \subseteq C \subseteq \Omega'$. By exploiting this encoding scheme, the search space is reduced and it is assured that the optimal solution can be reached.

This proposition presents some advantages:

- Admissibility of the solutions is ensured by the list algorithms.
- Many metaheuristics already exist to solve the permutation problems, with their neighborhood systems for single solution based metaheuristics and their crossover operator and/or mutation for population based metaheuristics.
- The metaheuristic is performed on a smaller set S than the original one Ω'.
- The set S contains all the optimal solutions.
- S conforms with the property of accessibility.
- This method is a combination of simple existing methods easy to implement.

3.2 General framework

The general framework can be used with any metaheuristic. Algorithm 1 shows the principle of hybridization between a local search and a list algorithm where $V : \Omega \longrightarrow \Omega$ denotes the neighborhood system and $H : S \longrightarrow \mathbb{N}$ the cost function.

Algorithm 1: Principle algorithm of the hybridization

```
1 Let be X ∈ Ω the initial solution
2 Y = L(X): a list algorithm is applied to the list X
3 while necessary do
4     Choose uniformly and randomly X' ∈ V(X)
5     Y' = L(X')
6     if H(Y') ≤ H(Y) then
7         X = X'
8         Y = Y'
```

Neighborhood system $V(X)$: Several classical neighborhood systems for permutation problems can be used; $P_{i,j}$: the item at position i permutes with the one which is at position j. $|P_{i,j}| = \frac{N.(N-1)}{2}$; $I_{i,j}$: the item at position i is inserted at position j. $|I_{i,j}| = N.(N-1)$. In both cases, V satisfies the accessibility and reversibility properties.

Cost function $H(Y)$: The most intuitive cost function is the number of used bins. But many solutions have the same value of cost function. To avoid it, [15] proposes a new cost function H which characterizes the average bin utilization, defined by Equation (1).

$$H(Y) = -\frac{\sum_{j \in N}(F_j(Y)/C)^k}{M(Y)} \tag{1}$$

Where $F_j(Y) = \sum_{i=1}^{N} w_i.\delta_{j,Y_i}, \forall j \in \{1, N\}$ is the sum of the sizes of the items packed in bin j (we use the Dirac function δ, $\delta_{a,x} = 1$ if $a = x$, 0 otherwise), $M(Y)$ is the number of bins used by the solution Y, $k > 1$ is a constant which defines how much a solution with equally filled bins is preferred over one in which some bins are rather full and other rather empty. A good value is $k = 2$ [9].

4 Experimentation

Hybridization has been tested with both metaheuristics: simulated annealing or kangaroo algorithm (iterated local search).

Originally, the inhomogeneous simulated annealing was used by [16] to simulate the physical annealing in metallurgy. Unfavorable transitions are accepted with a probability $e^{-\frac{H(Y')-H(Y)}{T}}$. Simulated annealing converges in probability to the set of optimal solutions if neighborhood system V satisfies the accessibility and reversibility properties [17]. The initial temperature T_0 is chosen such as all the transitions are authorized at the beginning, i.e. $e^{-\frac{H(Y')-H(Y)}{T_0}} \simeq 1, \forall (Y, Y')$ according to the algorithm proposed by [17]. The decreasing factor α is computed by $\alpha = \sqrt[IterMax]{(\frac{T_a}{T_0})}$. T_a is the latest temperature close to 0.

Kangaroo algorithm is an iterated local search. This algorithm consists in a stochastic descent, but if there is no improvement of the current solution after a number of iterations $A \geq |V|.ln(2)$ [18], a jump is made. To make this jump, a solution is chosen in a neighborhood system W different from V. Kangaroo algorithm converges in probability to the set of optimal solutions if neighborhood system W satisfies the accessibility property. W consists in applying several times neighborhood system V, it satisfies the accessibility property.

We compared all the possible combinations: choice of the metaheuristic (kangaroo algorithm or simulated annealing), choice of the list algorithm (NF, FF or BF) and choice of the neighborhood system ($P_{i,j}$ or $I_{i,j}$).

All of the referenced papers in the state of the art used the same instances, except [13]. In the instances, the bin capacity is equal to 150 and sizes of items between 20 and 100. Four sizes of problems are used: 120 items, 250, 500 and 1000. For each size, twenty different instances have been created. "u120" to "u1000" define the size of the instance. The results are about the sum of the differences between the optimal solution and the solution found by the considered method for all the instances of the same size. For example, for the twenty instances of size "u120", [9] finds the optimal solutions + 2 bins in total.

Table 2 compares our two best methods SABFP (Simulated Annealing + Best Fit + $P_{i,j}$) and KABFP (Kangaroo Algorithm + Best Fit + $P_{i,j}$) to the bibliographic ones. SABFP and KABFP use the Best Fit heuristic and the neighborhood system $P_{i,j}$. At the opposite, our hybridization does not work very well with the list algorithm Next Fit or with the neighborhood system $I_{i,j}$ (which disturbs too much the current solution).

This method gives good results. Our strength is the easy implementation of our proposition. It can be easily used on a lot of applications (resource constrained scheduling, assembly line balancing, multiprocessor scheduling).

Table 2. Results of the different methods to solve the bin packing problem

Prob	[9]	[12]	[8]	[11]	SABFP	KABFP
u120	+2	0	0	0	0	0
u250	+3	+2	0	0	+1	+2
u500	0	0	0	0	+2	+2
u1000	0	0	0	0	+3	+4

5 Conclusion

In this paper, we propose a hybridization between a metaheuristic and a list algorithm to solve the bin packing problem. Some methods already exist to solve it and perform very well. The main advantage of our proposition is its intelligibility because it combines two simple well-known methods. This method allows us to work on two different search spaces. This eases the used method. Several reasons justify this encoding scheme: it is not useful to check the admissibility of the solutions, many methods already exist to solve permutation problems, the set used for the metaheuristic is reduced and contains all the optimal solutions.

This resolution method used on the bin packing problem is currently applied on real applications such a problem of activities planning and resources assignment. We can also solve more sophisticated models thanks to this encoding scheme. Indeed, more constraints can be added to the list algorithms: incompatibilities between items, between items and bins, precedence.

Hybridization between a metaheuristic and a list algorithm can be used with any metaheuristic. In further work, hybridization will be used with population

based metaheuristic. The use of particle swarm optimization is the next step of our approach.

References

1. Johnson, D.S.: Near-optimal bin packing algorithms. PhD thesis, Massachusetts Institute of Technology (1973)
2. Chandra, A., Hirschberg, D., Wong, C.: Bin packing with geometric constraints in computer network design. Operations Research **26** (1978) 760–772
3. Bortfeldt, A., Wäscher, G.: Container loading problems: A state-of-the-art review. Univ., Faculty of Economics and Management (2012)
4. Wee, T., Magazine, M.J.: Assembly line balancing as generalized bin packing. Operations Research Letters **1** (1982) 56–58
5. Coffman Jr, E.G., Garey, M.R., Johnson, D.S.: An application of bin-packing to multiprocessor scheduling. SIAM Journal on Computing **7** (1978) 1–17
6. Gourgand, M., Grangeon, N., Klement, N.: Activities planning and resource assignment on multi-place hospital system: Exact and approach methods adapted from the bin packing problem. In: 7th International Conference on Health Informatics, Angers, France. (2014) 117–124
7. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-completeness. WH Freeman and Company, New York (1979)
8. Alvim, A.C., Ribeiro, C.C., Glover, F., Aloise, D.J.: A hybrid improvement heuristic for the one-dimensional bin packing problem. Journal of Heuristics **10** (2004) 205–229
9. Falkenauer, E.: A hybrid grouping genetic algorithm for bin packing. Journal of heuristics **2** (1996) 5–30
10. Martello, S., Toth, P.: Lower bounds and reduction procedures for the bin packing problem. Discrete Applied Mathematics **28** (1990) 59–70
11. Loh, K.H., Golden, B., Wasil, E.: Solving the one-dimensional bin packing problem with a weight annealing heuristic. Computers & Operations Research **35** (2008) 2283–2291
12. Levine, J., Ducatelle, F.: Ant colony optimization and local search for bin packing and cutting stock problems. Journal of the Operational Research Society **55** (2004) 705–716
13. Reeves, C.: Hybrid genetic algorithms for bin-packing and related problems. Annals of Operations Research **63** (1996) 371–396
14. Stawowy, A.: Evolutionary based heuristic for bin packing problem. Computers & Industrial Engineering **55** (2008) 465–474
15. Falkenauer, E., Delchambre, A.: A genetic algorithm for bin packing and line balancing. In: International Conference on Robotics and Automation, IEEE (1992) 1186–1192
16. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E.: Equation of state calculations by fast computing machines. The Journal of Chemical Physics **21** (1953) 1087–1092
17. Aarts, E.H., van Laarhoven, P.J.: Simulated Annealing: Theory and Applications. Kluwer Academic Publishers (1987)
18. Fleury, G.: Méthodes stochastiques et déterministes pour les problèmes NP-difficiles. Ph.D. thesis, Université Blaise Pascal, Clermont-Ferrand II (1993)