October 29, 1999
RT0321
Computer Science 17 pages

Research Report

Local Search Algorithms for the Bin Packing Problem and Their Relationships to Various Construction Heuristics

T. Osogami, H. Okano

IBM Research, Tokyo Research Laboratory
IBM Japan, Ltd.
1623-14 Shimotsuruma, Yamato
Kanagawa 242-8502, Japan

Local Search Algorithms for the Bin Packing Problem and Their Relationships to Various Construction Heuristics

Takayuki Osogami[*] Hiroyuki Okano[†]

IBM Japan, Tokyo Research Laboratory[‡]

Abstract. The tradeoff between the speed and quality of the solutions obtained by various construction and local search algorithms for the elementary bin packing problem (BPP) are analyzed to obtain useful information for designing algorithms for real-world problems that can be modeled as the BPP. On the basis of intensive computational experiments, we observe that the framework of a solution (i.e., a part of a solution consisting of large items or items with tight constraints) should be constructed in the early stages of a local search. New local search algorithms are proposed as an existential support for the observation.
Keywords: elementary bin packing problem, bin packing problem with conflicts, local search, construction heuristic, real-world problem, prioritized improvement

1 Introduction

Variants of the bin packing problem (BPP) can be seen various types of real-world problems, such as vehicle routing problems (VRPs) in logistics, and 2D or 3D packing problems in manufacturing. Typically, they are NP-hard combinatorial optimization problems, and it is difficult to obtain good greedy-type (construction) heuristics; instead, dedicated construction heuristics and local search algorithms are often used. In contrast, simple construction heuristics are known to be good for the one-dimensional BPP. The odds are that by using some information on the BPP about why some heuristics perform well, we can design good heuristics for real-world BPPs. In this paper, through intensive computational experiments, we first present such useful information on the BPP, and then verify its applicability to a real-world BPP. On the basis of the experiments, we propose new construction and local search algorithms for the BPP, which perform better than any previously known heuristics for the BPP.

1.1 The Bin Packing Problem

The BPP is known as an NP-hard combinatorial optimization problem, and is defined as follows (Garey and Johnson (1979)): given a finite set $U = \{ u_1, u_2, \dots , u_n\}$ of items and a rational size $s(u) \in [0,1]$ for each item $u \in U$, find a partition of U into disjoint subsets $U_1, U_2, . . . , U_k$ such that the sum of the item sizes in each U_i is no more than 1 and k is as small as possible. Without loss of generality, the BPP is considered to be the problem of packing a finite set of items of integer sizes $s(u_i) \in [0, C]$ into bins of capacity C to minimize the number of bins.

Many of the previous studies of the BPP have concerned the development of construction heuristics and theoretical analyses of their performances (Coffman et al. (1997)). Well-known construction heuristics include first fit (FF) and first fit decreasing (FFD). FF is an online algorithm that places each item

[*]E-mail: `osogami@jp.ibm.com`
[†]E-mail: `okanoh@jp.ibm.com`
[‡]1623-14, Shimotsuruma, Yamato-shi, Kanagawa-ken 242-8502, Japan

in a given order into the first (lowest indexed) bin into which it will fit. FFD is an offline algorithm that places each item in decreasing order of size into the first bin into which it will fit. Usually the asymptotic performance ratio R_A^∞ is used for analysis of the worst-case performance of the approximation algorithms for the BPP (Coffman et al. (1997)). Given a list of items L and an algorithm A, and letting $A(L)$ be the number of bins when A is applied to L and $OPT(L)$ be the optimal number of bins for L, the asymptotic performance ratio R_A^∞ is defined as follows:

$$R_A^\infty \equiv \inf\{r \geq 1 : \text{ for some } N > 0, R_A(L) \leq r \text{ for all } L \text{ with } OPT(L) \geq N\},$$

where $R_A(L) \equiv A(L)/OPT(L)$. The asymptotic performance ratio of FF and that of FFD are known to be $R_{FF}^\infty = \frac{17}{10}$ and $R_{FFD}^\infty = \frac{11}{9}$ (Johnson (1973)), and furthermore a polynomial-time offline algorithm with $R_A^\infty = 1$ (Karmarker and Karp (1982)) is known.

1.2 Real-World BPPs

As mentioned above, approximation algorithms with good performances for the BPP have been proposed. However, our goal is to develop efficient approximation algorithms for real-world problems that can be modeled as the BPP. As one approach, we are studying a method based on heuristics for the BPP. In the following, real-world problems that can be modeled as the BPP are called real-world BPPs, and the original BPP is called the elementary BPP. Falkenauer (1994) takes a similar approach and categorizes the elementary BPP as a grouping problem. He proposed a genetic algorithm (GA) that can be applied to general grouping problems, and applied it to the elementary BPP.

Two examples of real-world BPPs include VRPs and 2D or 3D packing problems. In a certain kind of VRP, minimization of the number of trucks is required when assigning delivery points to trucks and generating routes that start at a depot, visit delivery points, and return to the original depot. If the number of trucks is the same, solutions with shorter total route lengths are preferred. This type of problem can be modeled as the BPP by considering trucks and visiting points as bins and items, respectively. In this case, the objective function includes the total length of routes as well as the number of trucks, and it is necessary to satisfy various constraints, such as combinations of items that can be loaded onto the same truck. The routes of trucks can be obtained by an approximation algorithm for the traveling salesman problem (TSP).

In certain kinds of 2D or 3D packing problems (Okano (1998), Osogami (1998)), various real-world constraints, including the directions of the items and the clearances between items, have to be considered. In this case, it is sometimes practical to determine groups of items that should be placed in the same bins before obtaining an optimal packing pattern for each bin, which requires a huge computation time.

1.3 Local Search Algorithms for the BPP

Because it is typically difficult to develop good construction heuristics for these real-world BPPs, unlike the elementary BPP, it is practical to improve constructed solutions by using local search algorithms. Previous local search algorithms for the elementary BPP include the simulated annealing (SA) algorithm by Kämpke (1988), the genetic algorithm (GA) by Falkenauer (1994), and the GA by Reeves (1996). Kämpke's method fixes the number of bins to the target and balances the contents of bins by moving items from one bin to another or by swapping items in different bins. Falkenauer's GA performs genetic operations on the sequences of bins in which items are packed and performs further local search algorithm in crossovers and mutations. Reeves's GA performs genetic operations on the sequences of items and obtains a packing pattern from the sequence of items by means of construction heuristics, including FF and FFD.

Feasibility checks and calculations of the objective function in real-world BPPs are often more complex than those in the elementary BPP. For example, in the abovementioned VRP, the lengths of trucks' routes are obtained by an approximation algorithm for the TSP. Although efficient approximation algorithms, which experimentally run in $O(N \log N)$ time on uniform data sets, where N is the number of the cities, are known for the TSP with L_2 metric (Bentley (1992)), the time complexity cannot be ignored when they are used to evaluate neighborhoods iteratively in local search algorithms. Therefore, the tradeoff between the speed and quality of the solutions must be considered in local search algorithms for real-world BPPs. Note that studies on the same tradeoff focusing on a single local search, such as 2-opting for the TSP (Okano et al. (1999)), aim to select the best construction heuristic. On the other hand, this paper aims to obtain useful information for designing both construction and local search algorithms for real-world BPPs through simple experiments on the elementary BPP.

1.4 Framework of This Paper

In this paper, comparing various combinations of construction and local search algorithms, we first identify the conditions for local search algorithms to find good solutions in the elementary BPP, then show combinations of construction and local search algorithms that have such conditions and also have a good tradeoff between the speed and quality of the solutions. Various algorithms are also applied to a real-world BPP to examine whether the conditions for the elementary BPP hold true also in real-world BPPs. We use the BPP with conflicts (Jansen (1998)) as an example of real-world BPPs, where a graph $G = (V, E)$ shows constraints and adjacent items $(i, j) \in E$ have to be assigned to different bins.

Section 2 illustrates various construction and local search algorithms for the elementary BPP that will be used in the computational experiments in Section 3, where the qualities of solutions and the numbers of searches performed by various local search algorithms are compared. We also discuss the conditions for the construction and local search algorithms to reach good solutions and give local search algorithms that find good solutions and have a good tradeoff between the speed and quality of the solutions. In Section 4, the performances of various local search algorithms are compared for the BPP with conflicts. Section 5 concludes the paper.

2 Local Search Algorithms for the BPP

This section illustrates various construction and local search algorithms that are used in the later computational experiments. Local search is a type of algorithm that starts at an initial solution constructed by a certain construction heuristic, iteratively moves to a better solution among the defined neighborhood solutions, and obtains a locally optimal solution. A local search algorithm can be defined by the following: a solution space, a neighborhood, an objective function, and a search method. These will be defined in Subsections 2.1–2.5. The complexity of optimality checking is mentioned in Subsection 2.6, and various construction heuristics are given in Subsection 2.7.

2.1 Solution Space

A solution space can be defined to be the set of feasible solutions or the set of all solutions. Falkenauer (1994) and Reeves (1996) adopted the former and Kämpke (1988) the latter. If the solution space of the set of feasible solutions is so small that it is difficult to define an appropriate neighborhood, it is better to include some infeasible solutions in the solution space. In this case, a certain method is required to guarantee that the final solution will be feasible. In this paper, the solution space is defined to be the set of feasible solutions.

2.2 Objective Function

It is natural to define the objective function for the BPP to be the number of bins required to pack all items, but it is difficult to define an appropriate neighborhood with this objective function. That is, in most cases, there is no neighborhood operation that reduces the number of bins, and local search algorithms do not work well. Therefore, Kämpke's objective function is to balance the contents of bins, fixing the number of bins to the target. This method is good when the solution space includes infeasible solutions, but it does not work when the solution space is the set of feasible solutions. Accordingly, Falkenauer and Delchambre (1992) have proposed the following objective function to be maximized:

$$f = \frac{\sum_{i=1}^{N} F_i^k}{N}, \tag{1}$$

where N is the number of bins used and F_i is the total size of items in bin i. When $k = 1$, the right hand side of the equation is the total size of all items, which is constant, divided by N, so the objective function is equivalent to the minimization of the number of bins. When $k > 1$, the greater the variance of the total size of the items in each bin, the larger the value of the function f, if the number of bins used is constant. *Nearly filled* bins and *nearly empty* bins become more preferred as k increases, where nearly filled and nearly empty mean that the level of occupancy of a bin is nearly 100% and nearly 0%, respectively. When $k > 2$, a solution with a larger number of bins can be better than a solution with a smaller number of bins (Falkenauer (1998b)). For this reason, Falkenauer concluded that $k = 2$ is the best. Reeves also used this objective function with $k = 2$.

A drawback of Equation (1) is that it does not guarantee that a solution with a smaller number of bins is always better than a solution with a larger number of bins when the solution contains more than two bins that are not occupied to half of their capacity (Falkenauer (1998b)). In this paper, the objective function is generalized to:

$$f = \frac{\sum_{i=1}^{N} F_i^k}{N^\alpha}. \tag{2}$$

Clearly, when $\alpha = 1$, Equation (2) is equivalent to Equation (1). In this paper, a sufficiently large α is selected so that solutions with larger numbers of bins always produce smaller values of the objective function. We use $k = 2$, since different values of k did not affect the performances of algorithms throughout the experiments if $k > 1$.

2.3 Neighborhood

Kämpke's neighborhood operation randomly selects two bins, including the most occupied bin, selects one item from each bin at random, and exchanges the two selected items. If the *floor* of the item at the top of a bin is located higher than the *ceiling* of the item at the top of another bin, the former item is moved into the latter bin to make the two bins balanced. On the other hand, the local search that is applied in the crossovers and the mutations of Falkenauer's GA exchanges items that are not packed and those in a certain bin. For example, the mutation randomly selects a few bins, unpacks all items in the selected bins, and exchanges up to three items that are unpacked and up to two items in a certain bin. The exchanges of items are repeated until no exchange of items can improve the objective function, and finally the items that are not packed are placed by FFD.

In this paper, *p-q-exchange*, which is a modification of Kämpke's neighborhood operation, is used. In a p-q-exchange, up to p items and up to q items are exchanged between two arbitrary bins. Racer (1994) and Yagiura et al. (1997) have used 1-1-exchange for the generalized assignment problem, and Chiang (1998) has used p-p-exchange for the assembly line balancing problem.

2.4 Search Method

A search method determines which of the better solutions in the neighborhood a local search algorithm should move to. The two most frequently used search methods are the best improvement (BI) and the first improvement (FI).

The BI evaluates all the solutions in the neighborhood and moves to the best one. The local search algorithms performed in the crossovers and the mutations in Falkenauer's GA uses the BI. On the other hand, the FI moves to the first solution that is found to be better than the current solution. The order of the search has to be defined in the FI, and a random order is often used, as in Kämpke's SA.

In this paper, the BI and some variants of the FI are compared. The BI applies the p-q-exchange that best improves the objective function among all the combinations of items, while the FI with random searching order, which is called simply the FI in the following, applies the p-q-exchange between the two bins first found to be improvable that best improves the objective function. In Subsection 2.5, we propose another search method.

2.5 A New Search Method

We propose a search method that searches in a predefined searching order and moves to the first solution that is found to be better than the current solution. This method is called the prioritized improvement (PI) in the following. In particular, two kinds of PIs are considered for the elementary BPP. One prioritized improvement method (PI_1) searches bins with smaller total sizes of items earlier. The intuitive concept behind the PI_1 is that bins with smaller total sizes of items should easily become empty and they should be more able to accept other items. In this sense, the PI_1 performs similarly to the BI. The other prioritized improvement (PI_2) searches bins with greater average sizes of items earlier. Because items with greater sizes are difficult to pack, the PI_2 first tries to make nearly filled bins using items with greater sizes; then, it improves other bins composed of items with smaller sizes. Such nearly filled bins should be relatively more difficult to improve than others; that is, the PI_2 tries to make nearly filled bins using items that are difficult to pack in the early stages of the algorithm.

To be more precise, a PI sorts bins in a prioritized order and searches as follows:

Search order of a PI

```
Step. 1  for $n = 3$ to $N \times 2 - 1$ do          ($N$: number of bins)
Step. 2    for $i = \max\{n - N,\ 1\}$ to $\lceil n/2 - 1 \rceil$ do
Step. 3      $j = n - i$
Step. 4      search all the neighborhoods between bins $i$ and $j$.
```

2.6 Complexity of Optimality Checking

The size of the neighborhood affects the complexity of the search, which is the complexity of determining whether the current solution is locally optimal. The complexity of the optimality checking of the local search algorithm defined above is $O(n^2)$ with 1-1-exchange, $O(\frac{n^3}{N})$ with 1-2-exchange, $O(\frac{n^4}{N^2})$ with 2-2-exchange, and $O(\frac{n^{p+q}}{N^{p+q-2}})$ with p-q-exchange, where n is the number of items and N is the number of bins used in the current solution.

However, local search algorithms with the BI need not search all the combinations of bins once the neighborhood is searched. A neighborhood operation is considered to be an operation between two bins, and the neighborhood operations among the other bins are independent of the neighborhood operations between the two. Accordingly, the neighborhood operations between the bins that are independent of the bins between which the previous neighborhood operations are performed need not be searched again, since they never contain any improving neighborhood. Consequently, the BI's complexity for searching a neighborhood is $O(\frac{n^2}{N})$ with 1-1-exchange, $O(\frac{n^3}{N^2})$ with 1-2-exchange, $O(\frac{n^4}{N^3})$ with 2-2-exchange, and $O(\frac{n^{p+q}}{N^{p+q-1}})$ with p-q-exchange. Similarly, in the FI and PIs, it is not necessary to re-search the neighborhood between the bins that had been searched previously if these bins are independent of the bins between which the previous neighborhood operations were performed.

2.7 Construction Heuristics

Next fit (NF), as well as FF and FFD, is used as a construction heuristic for initial solutions. In NF, the last used bin is considered to be *open* and the others are *closed*. Each item is placed in a given order into the open bin if possible; otherwise, the open bin is closed and the item is placed into a new open bin.

Furthermore, two variants of those three construction heuristics are considered. One variant restricts the number of items in each bin, which is called $\mathrm{C}_k CH$, where k is the upper bound of the number of items in each bin and CH is the name of the original construction heuristic. For example, C_3FFD places items in the same way as FFD, except that it does not place items into a bin that already contains three items. When $k = 1$, $\mathrm{C}_k CH$ is independent of the original construction heuristic CH, since it produces the solution in which all bins contain exactly one item, so $\mathrm{C}_1 CH$ is called C_1 in the following. The other type of modification reduces the capacity of all bins, which is called $\mathrm{R}_d CH$, where d is the reduced capacity and CH is the name of the original construction heuristic. For example, R_5FFD places items in the same way as FFD, except that the capacity of bins is reduced by 5 (we will use 150 for the capacity of bins in experiments). Clearly, $\mathrm{C}_k CH$ and $\mathrm{R}_d CH$ are equivalent to the original construction heuristic CH when k is large enough or $d = 0$, respectively.

3 Application of Local Search Algorithms to the Elementary BPP

In this section, the local search algorithms described in Section 2 are applied to the elementary BPP, and the qualities of the obtained solutions and the numbers of the performed searches are compared. The settings of the experiments are illustrated in Subsection 3.1. In Subsection 3.2, the results of the various computational experiments are shown and the conditions for local search algorithms to find good solutions are discussed.

3.1 Settings of Experiments

The computational experiments in this section use instances called the uniform class, which are a part of the test instances of OR-Library[†]. OR-Library has other instances called the triplet class, but there

[†]http://mscmga.ms.ic.ac.uk/info.html

is little difference among the solutions obtained by various local search algorithms for this class, and therefore the uniform class is used here. The instances of the uniform class consist of items of integer sizes that are uniformly distributed between 20 and 100 and bins of capacity 150. There are four classes, corresponding to the numbers of items, which are 120, 250, 500, and 1000. Each class has 20 instances, and the average results for these 20 instances are used in the following discussion. Because these instances are claimed to be easy (Gent (1998, 1999), Falkenauer (1998a)), we use other two classes that have 5000 or 10000 items, which were generated in the same way as the uniform class of the OR-Library. These two classes also have 20 instances each. In the following, instances with 120, 250, 500, 1000, 5000, and 10000 items are called u120, u250, u500, u1000, u5000, and u1000, respectively.

In this section, the quality of the solution is represented by the difference of the number of bins from the number of bins in the globally optimal solution or from the lower bound of the number of bins. For the instances from OR-Library, the difference of the number of bins from that in the globally optimal solution is used as the quality index of solutions, since the globally optimal solution has been obtained by Carvalho (1999). For u5000 and u10000, the difference between the number of bins and the lower bound of the number of bins is used. The lower bound of the number of bins N_{LB} is obtained as follows:

$$N_{LB} = \left\lceil \frac{\sum_{i=1}^{n} s_i}{C} \right\rceil, \tag{3}$$

where n is the number of items, s_i is the size of item i, and C is the capacity of a bin.

In the following discussion, the number of searches, which is the number of evaluations needed to check whether a neighborhood operation improves the objective function, is used as the speed index instead of the CPU time. This is because the CPU times for the evaluation of the objective function and the feasibility check vary widely among real-world BPPs, while the number of searches depends mainly on the characteristics of the local search algorithms. The CPU time is also affected by the performance of computers and their implementations, which are irrelevant to the essence of the algorithm.

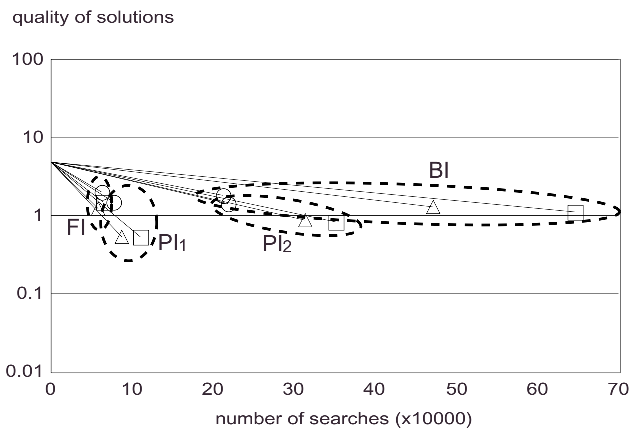
Note that *filled* bins, that is, bins occupied to their full capacity, are not searched in our implementations because, once filled, they do not contribute to any improvement of the objective function. Most of the bins become filled as local search algorithms proceed, so the number of searches is greatly decreased by not searching them.

3.2 Computational Experiments

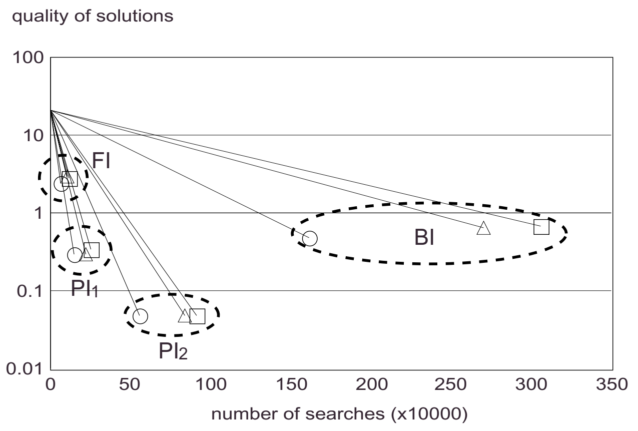
3.2.1 A Condition for Local Search Algorithms to Obtain Good Solutions

First of all, only well-known construction heuristics, FFD, FF, and NF, are used, and the performances of various local search algorithms are compared. Figure 1 shows (a) the qualities of solutions obtained by various local search algorithms, and (b) the numbers of searches performed in the local search algorithms to reach locally optimal solutions. Segments are drawn between the results of construction heuristics and the results after application of the local search algorithms. Figure 1 shows the results only for u1000, but the characteristics of construction and local search algorithms were the same in other instances.

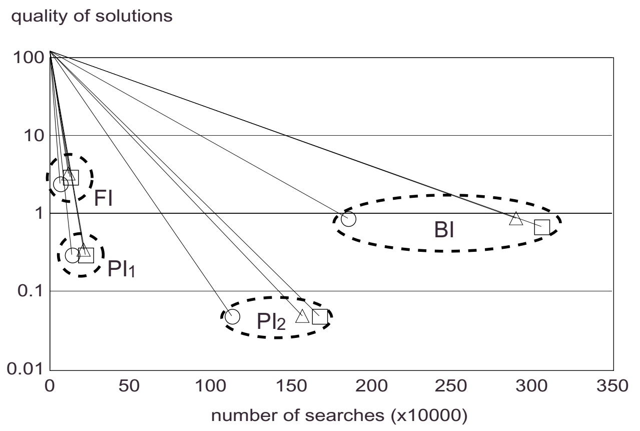
The best-quality solutions were obtained by the PI_2 after FF or NF, where the quality of solutions was 0.05, which means that the algorithm found the solutions that had the same number of bins as that of the globally optimal solution in 19 out of 20 instances. In the PI_2, the greater the average size of items in a bin, the earlier the bin is searched. Therefore, in the early stages of the PI_2, items with greater sizes are packed to form nearly-filled bins. If these items cannot compose nearly-filled bins until the final stages, bins containing these items have little possibility of being improved. In other words, the PI_2 first constructs a rough framework of the solution with items with greater sizes, which are more difficult to



(a) FFD



(b) FF



(c) NF

○ :1-1-exchange △ :1-2-exchange □ :2-2-exchange

Figure 1: Performance of various local search algorithms with various construction heuristics

pack, and then constructs the details of the solution with smaller items. The grand-tour method (Misono and Iwano (1996)), a construction heuristic for the TSP that generates a shorter tour and runs faster than the multiple-fragment method, is based on a similar idea. The following observation summarizes the discussion:

Observation 1. *The framework of a solution (i.e., a part of a solution consisting of large items) should be constructed in the early stages of a local search.*

3.2.2 Dependence on Initial Solutions

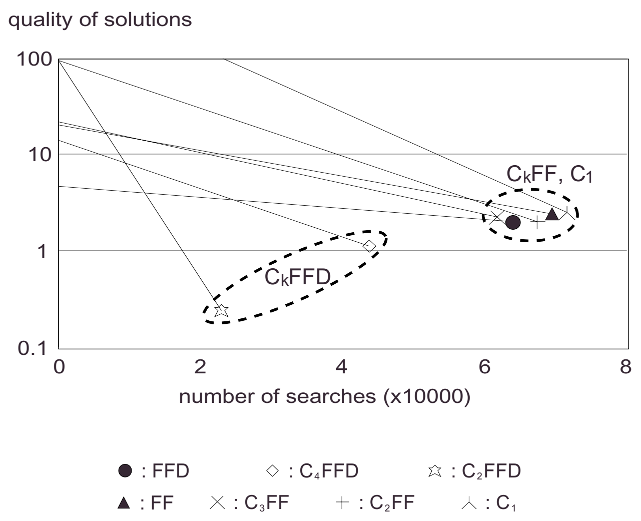
From Figure 1, we can see that the quality of the solutions obtained by local search algorithms depended on the initial solutions and search methods rather than the neighborhood operations, while the number of searches monotonically increases as the size of the neighborhood increases. We can also see that good initial solutions were not necessarily improved to good solutions by local searches. Locally optimal solutions obtained by the BI, PI_1, and PI_2 starting from initial solutions constructed by FFD were all worse than those starting from initial solutions constructed by FF or NF, though those obtained by the FI after FFD were slightly better than those after FF or NF. In the later discussion, we apply 1-1-exchange as the neighborhood operation, since there is little difference in the quality of the solutions among various p-q-exchanges ($p \geq 1$, $q \geq 1$), while the complexity increases monotonically as p or q increases.

Figure 2 shows the performance of the FI with various construction heuristics, including $\mathrm{C}_k CH$s and $\mathrm{R}_d CH$s. A remarkable result is that when initial solutions were constructed by C_kFFD, the qualities of solutions were better and the numbers of searches were smaller than those after FFD; and the smaller k was, the better the qualities of solutions and the smaller the numbers of searches were. This characteristic also held true for the PI_1. For the BI and PI_2, the qualities of solutions became better as k decreased, but the number of searches increased.

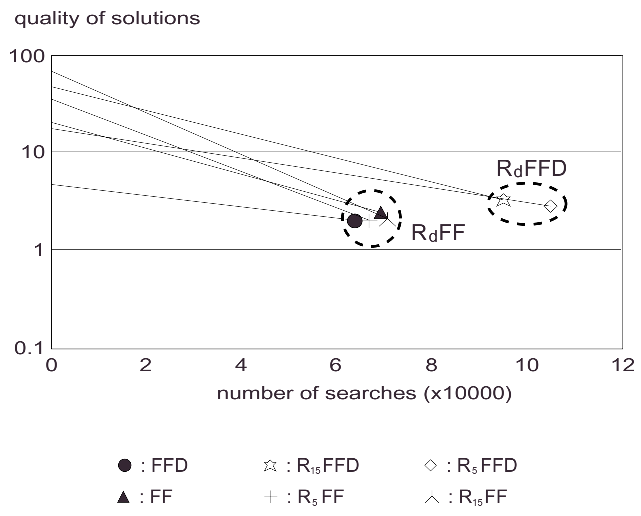
On the other hand, when initial solutions were constructed by C_kFF or R_dFF, the performance, which is the number of searches and the quality of solutions obtained by a local search applied after them, was about the same as that after FF. This was also true for other search methods, except that the BI after R_dFF yielded worse solutions and required a larger number of searches than the BI after FF. The number of searches after initial solutions constructed by R_dFFD was larger than that after FFD, while the quality of solutions was about the same. This was also true for other search methods.

Accordingly, we conclude that whether initial solutions are good or bad does not very much affect the performance of local search algorithms for the BPP. To clarify the conditions of initial solutions from which local search algorithms obtain good solutions, we focus on the characteristics of C_2FFD, since the combination of C_2FFD and the FI has a good tradeoff between the speed and quality of the solutions. In the instances used in this section, the average number of items in a bin of optimal solutions is about 2.5, since the capacity of a bin is 150 and the sizes of items are uniformly distributed between 20 and 100. Because C_kFFD places items in decreasing order of their size, in the solutions obtained by C_2FFD, there exist both nearly filled bins composed of the pairs of relatively large items and nearly empty bins composed of pairs of small items. Since it takes more neighborhood operations to improve bins with small items, the bins with large items are naturally nearly filled in the early stages of the algorithm. This may be a reason for C_2FFD's good performance. More formally, the local optimality of the bins with large items is guaranteed by the following theorem and corollary:

Theorem 1. *The solution obtained by C_2FFD is optimal with respect to the swap operation, where Equation (2) is used as the objective function.*



(a) FI after C$_k$$CH$s



(b) FI after R$_d$$CH$s

Figure 2: Performance of the first improvement (FI) after various C$_k$$CH$s and R$_d$$CH$s

The proof of this theorem is given in the appendix. Clearly, between the bins with items whose sizes are all greater than C/3, there is no move operation that produces a feasible solution. Accordingly, we obtain the following corollary.

Corollary 1. *If a subset of the bins in a solution obtained by C_2FFD consists solely of bins with items whose sizes are greater than C/3, then the subset is optimal with respect to 1-1-exchange, where Equation (2) is used as the objective function.*

In our experience, when the capacities of bins are increased without changing the sizes of items, the problem become easier, and local search algorithms almost always find solutions that have the same number of bins as in the globally optimal solutions. This observation may support C_2FFD's efficiency for relatively difficult instances.

3.2.3 Other Search Methods

The above discussion leads to the following local search algorithm (algorithm A), which provides support for the correctness of our discussion:

```
Algorithm A
Step. 1    construct an initial solution by $C_2$FFD
Step. 2    for $k = 3$ to $N$ do          ($N$: maximum number of item in a bin)
Step. 2.1    apply a local search algorithm under the constraint that no bin has more than $k$ items
Step. 2.2    $k := k + 1$
```

Clearly, algorithm A forces items with greater sizes to be placed into nearly filled bins in the early stages. If Observation 1 is correct, algorithm A with any search method finds good solutions. Algorithm A with the BI, FI, or PI_1 is applied to u1000, and the results are shown in Figure 3.

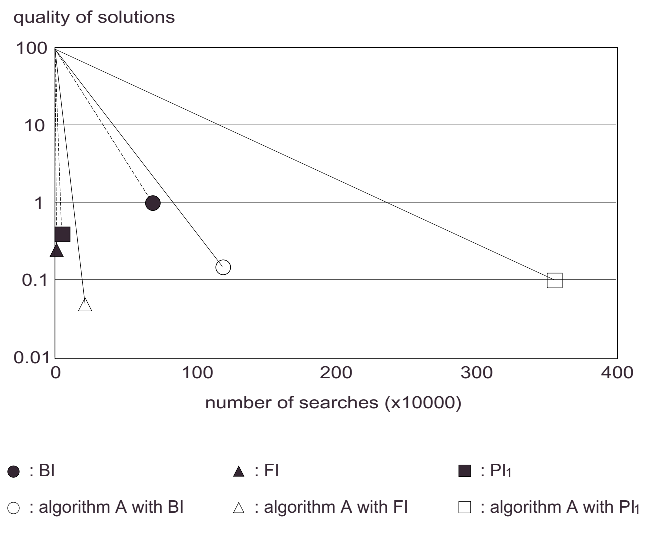
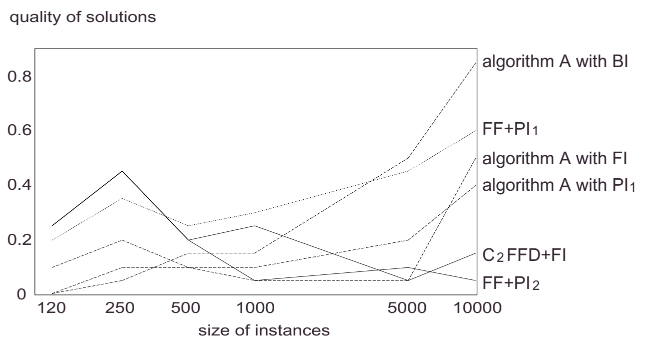


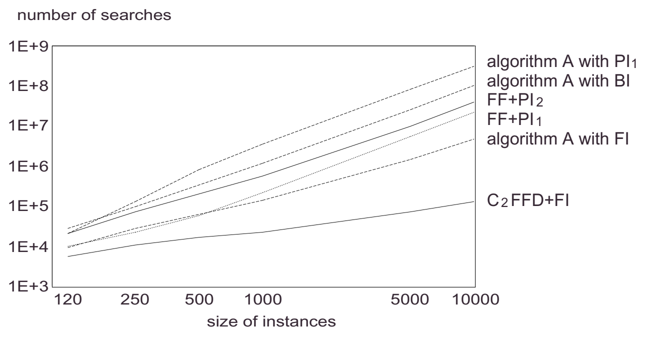
Figure 3: Performances of algorithm A and various local search algorithms after C_2FFD

Figure 3 shows that algorithm A was hardly affected by search methods and found solutions that were comparable to the solutions obtained by the PI_2. This result supports Observation 1 that a framework of a solution should be constructed in the early stages of a local search. The number of searches performed in algorithm A was relatively large. This is mainly because the algorithm begins searching from scratch after the value of k is incremented in our current implementations.

3.3 Scalability of Various Local Search Algorithms



(a) Quality of solutions



(b) Numbers of searches

Figure 4: Scalability of various local search algorithms

Finally, the performances of various local search algorithms when they were applied to instances of different sizes were evaluated to clarify the best combinations of heuristics for the elementary BPP. Figure 4 (a) shows the qualities of solutions obtained by various combinations of construction and local search algorithms. Note that Figure 4 shows only results in which the number of bins in excess of the lower bound was less than 1 on the average for u10000. Algorithm A performed well for small instances. In particular, algorithm A with the BI or PI_1 obtained solutions with optimal numbers of bins in all the 20 instances in u120. In contrast, the PI_2 after FF and the FI after C_2FFD performed well for large instances. Both combinations obtained solutions with the same numbers of bins as the lower bound in at least 17 out of 20 instances for u5000 or u10000.

Figure 4 (b) shows the number of searches performed by the combinations of construction and local search algorithms that appeared in Figure 4 (a). We can say that the FI after C_2FFD and algorithm A after the FI have good tradeoffs for large and small instances, respectively.

4 Application of Local Search Algorithms to a Real-World BPP

In Section 3, we showed that Observation 1 holds true for the elementary BPP. In this section, we will discuss whether the same observation, that the framework of a solution should be constructed in the early stages of a local search, also holds true for real-world BPPs.

4.1 BPP with conflicts

In this section, we consider the bin packing problem with conflicts (BPP with conflicts), which is defined as follows (Jansen (1998)): given an undirected graph $G = (U, E)$, a finite set $U = \{ u_1, u_2, \ldots , u_n\}$ of items and a rational size $s(u) \in [0,1]$ for each item $u \in U$, find a conflict-free partition of U into disjoint subsets $U_1, U_2, \ldots , U_k$ such that the sum of the sizes of the items in each U_i is no more than 1 and k is as small as possible. If E is an empty set, it is the elementary BPP. Furthermore, if $\Sigma_{j \in U} s_j \leq 1$, it is the problem of computing the chromatic number of the conflict graph G, which is also known to be NP-hard (Garey and Johnson (1979)). Without loss of generality, the BPP with conflicts is considered to be the problem of finding a conflict-free packing of a finite set of items of integer sizes $s(u_i) \in [0, C]$ into bins of capacity C to minimize the number of bins.

4.2 A PI for the BPP with conflicts

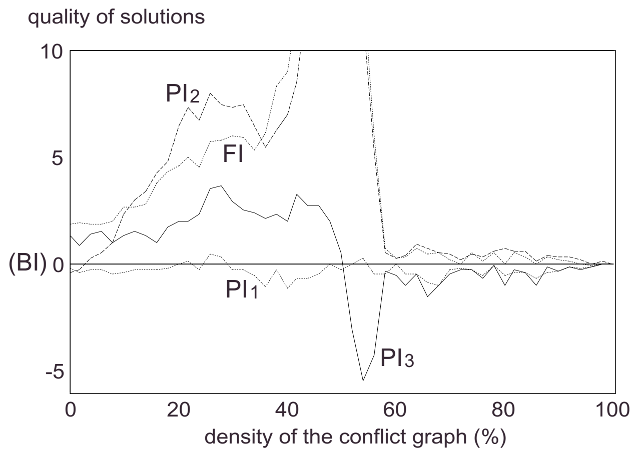
In Section 3, we showed that Observation 1 holds true for the elementary BPP. In the BPP with conflicts, items that are difficult to pack, which should compose a framework of a solution in the early stages of local search, are not only items with large sizes but also items with large degrees, that is, items that conflict with many other items. Accordingly, in this section, we consider another PI, which is called the PI$_3$ in the following. The PI$_3$ searches bins with greater average degrees of items earlier. If the observation also holds true in the BPP with conflicts, then the PI$_2$ should perform well when conflict graphs are sparse, and the PI$_3$ should perform well when conflict graphs are dense.

4.3 Settings of Experiments

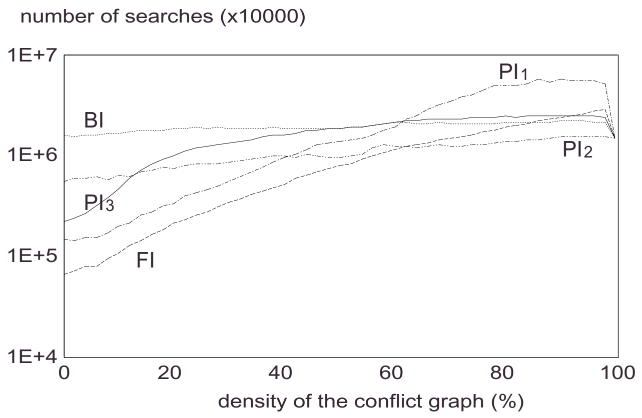
The instances used in this section are u1000 with undirected graphs where edges are randomly generated. The construction heuristics for the elementary BPP were modified for the BPP with conflicts so that the items that conflict with any item in a bin are not placed in that bin. The neighborhood operations for the elementary BPP were also modified so that conflicts of edges are considered in feasibility checking. Maximization of Equation (2) is used as the objective function. 1-1-exchange is used as the neighborhood operation, as in the elementary BPP, since there was little difference in the qualities of solutions among various p-q-exchanges ($p \geq 1$, $q \geq 1$).

4.4 Computational Experiments

We show how the performances of various local search algorithms vary as the density of the conflict graph increases, where the density of the graph is the ratio of the number of edges to the number of edges of its corresponding complete graph. Here, initial solutions are constructed by FFD in terms of the degree of the item (FFD$_{\text{degree}}$), by placing each item in decreasing order of degree into the first bin into which it will fit. FFD$_{\text{degree}}$ produced the best initial solution among all the construction heuristics considered in this paper when the conflict graph was dense; to be more precise, when the density of the graph was more than about 25%. The higher the quality of initial solution, the better the performance of the local search algorithm when the conflict graph was dense, especially when the density was more than 60%. When there is no edge in the conflict graph, $\text{FFD}_{\text{degree}}$ is equivalent to FF, which was fairly good when combined with local search algorithms except the FI. Consequently, in many cases of the BPP with conflicts, $\text{FFD}_{\text{degree}}$ is good as a construction heuristic for initial solutions in terms of the total performance when combined with local search algorithms.



(a) Quality of solutions



(b) Numbers of searches

Figure 5: Performance of various local search algorithms in the BPP with conflicts

Figure 5 (a) shows the quality of solutions obtained by various local search algorithms, where the quality of a solution is shown as the number of bins in excess of that obtained by the BI. In terms of the quality of solutions, the PI_1 was similar to the BI, though the PI_1 produced slightly better solutions in most cases. When there were few edges in the conflict graph, the PI_2 had the best performance. However, its performance gradually declined as the number of edges increased, and it never produced solutions better than that by the BI on average when the density was more than 4%. The dependence of its performance on the density of the conflict graph was similar to that of the FI. The quality of solutions obtained by the PI_3 was never worse than that of solutions obtained by the BI when the density of the conflict graph was more than 50%. In particular, when the density was around 54%, the quality of the solutions obtained by the PI_3 was outstanding. From the above results, we conclude that Observation 1 also holds true in the BPP with conflicts.

Figure 5 (b) shows the number of searches performed by each local search algorithm. In terms of the tradeoff between the speed and quality of the solutions, the PI_1 had a good performance when the density of the conflict graph was relatively low, and the PI_3 and BI were both good when the density was relatively high. The PI_2, which produced quite good solutions in the elementary BPP, performed well only when there were few edges in the conflict graph. We conjecture that some PI that prioritizes both the sizes and degrees of items may produce good solutions where neither the PI_2 nor PI_3 performs worse than the BI or PI_1.

5 Conclusion

In this paper, we clarified that the following statement holds true both for the elementary BPP and real-world BPPs: the framework of a solution (i.e., a part of a solution consisting of large items or items with tight constraints) should be constructed in the early stages of a local search. That is, items that are difficult to pack should be placed in nearly filled bins in the early stages of local search algorithms. New local search algorithms that have such characteristics were proposed, and their performances were evaluated through various computational experiments.

In terms of the tradeoff between the speed and quality of the solutions, the combination of algorithm A, which forces large items to be placed in nearly filled bins in the early stages, and the first improvement was good for small instances of the uniform class of the elementary BPP, while the combination of the first improvement and C_2FFD was good for large ones. These construction and local search algorithms were also applied to the BPP with conflicts. In the BPP with conflicts, one prioritized improvement (PI_1), which prioritizes bins with small total sizes of items, had a good performance when the density of the conflict graph was relatively low; and the best improvement and another prioritized improvement (PI_3), which prioritizes items that conflict with more items, had good performances when the density was relatively high.

The authors hope that the observation presented in this study will provide useful information for designing algorithms for real-world problems that can be modeled as the BPP. Successful design of new algorithms for the BPP with conflicts based on the observation may provide existential support for real-world BPPs for which efficient approximation algorithms can be found by our approach.

Appendix: Proof of Theorem 1

It is sufficient to show that when arbitrary two bins i, j $(i < j)$ are selected from the solution obtained by C_2FFD, there is no swap operation that improves the objective function between the two bins. Clearly, there is no swap operation that reduces the number of bins, because only a swap of two items is considered, so the objective function is equivalent to $\sum_{i=1}^{N} F_i^k$. Four cases are possible, corresponding to the number of items in each bin.

Case 1: Both bin i and bin j have one item each.

Clearly, the swap operation does not change the solution, and there is no swap operation that improves the objective function between the two bins.

Case 2: Bin i has one item and bin j has two.

It has previously confirmed that items in bin j cannot be placed in bin i, since bin i has a lower index than bin j. Therefore, an item in bin j and the item in bin i cannot be in the same bin, but an item in bin j and the item in bin i must be placed in the same bin by any swap operation. Consequently there is no swap operation between the two bins that produces any feasible solution.

Case 3: Bin i and bin j have two items each.

Let a and b ($a \geq b$) be the sizes of the two items in bin i, and c and d be the sizes of the items in bin j. For simplicity, let a also mean the item with size a and so on. a is the largest among the four items, since bin i has a lower index than bin j. It is sufficient to consider the swap of b and an item in bin j, since the swap of a and c is equivalent to the swap of b and d and so on. If $c > b$ and $d > b$, then c and d have been packed before b. The fact that neither c nor d are packed into bin i indicates that neither c nor d can be in a bin with a. Therefore swapping b and any item in bin j produces an infeasible solution. If at least one item in bin j (let it be c) is not greater than b ($b \geq c$), then it is possible to swap b and c, but in this case the improvement of the objective function is

$$(a+c)^2 + (b+d)^2 - [(a+b)^2 + (c+d)^2] = -2(a-d)(b-c) \leq 0,$$

since $a \geq d$, $b \geq c$. Therefore, there is no swap operation that improves the objective function between the two bins.

Case 4: Bin i has two item and bin j has one.

The discussion in case 3 still holds when $d = 0$, so there is no swap operation that improves the objective function between the two bins.

It is concluded that there is no swap operation that improves the objective function between the two bins in any case, and thus the theorem is proved.

References

J. J. Bentley. (1992). "Fast Algorithms for Geometric Traveling Salesman Problems," *ORSA Journal of Computing* 4, 387–411.

V. Carvalho. (1999). "Exact Solution of Bin-Packing Problems Using Column Generation and Branch-and-Bound," *Annals of Operations Research* 86, 629–659.

W.-C. Chiang. (1998). "The Application of a Tabu Search Metaheuristic to the Assembly Line Balancing Problem," *Annals of Operations Research* 77, 209–227.

E. G. Coffman, M. R. Garey, and D. S. Johnson. (1997). "Approximation Algorithms for Bin Packing: a Survey," In D. S. Hochbaum (ed.), *Approximation Algorithms for NP-Hard Problems.* PWS Publishing Company.

E. Falkenauer and A. Delchambre. (1992). "A Genetic Algorithm for Bin Packing and Line Balancing," In *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1186–1192.

E. Falkenauer. (1994). "A Hybrid Grouping Genetic Algorithm for Bin Packing," *Journal of Heuristics* 2, 5-30.

E. Falkenauer. (1998a). "On Method Overfitting," *Journal of Heuristics* 4, 281–287.

E. Falkenauer. (1998b). *Genetic Algorithms and Grouping Problems.* John Wiley & Sons Ltd.

M. R. Garey and D. S. Johnson. (1979). *Computers and Intractability — A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company.

I. P. Gent. (1998). "Heuristic Solution of Open Bin Packing Problems," *Journal of Heuristics* 3, 299–304.

I. P. Gent. (1999). "A Response to "On Method Overfitting," *Journal of Heuristics* 5, 109–111.

K. Jansen. (1998). "An Approximation Scheme for Bin Packing with Conflicts," In *Proceedings of the 6th Scandinavian Workshop on Algorithm Theory*, pp. 35–46.

D. S. Johnson. (1973). "Near-Optimal Bin Packing Algorithms," *Ph.D. thesis*, Massachusetts Institute of Technology, Department of Mathematics, Cambridge.

T. Kämpke. (1988). "Simulated Annealing: Use of a New Tool in Bin Packing," *Annals of Operations Research* 16, 327–332.

N. Karmarker and R. M. Karp. (1982). "An Efficient Approximation Scheme for the One-Dimensional Bin Packing Problem," In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, pp. 312–320.

S. Misono and K. Iwano. (1996). "Experiments on TSP Real Instances," *IBM Research Report*, RT0153.

H. Okano. (1998). "An Approximation Algorithm for the 2D Free-Form Bin Packing Problem," *IBM Research Report*, RT0286.

H. Okano, S. Misono, and K. Iwano. (1999). "New TSP Construction Heuristics and Their Relationships to the 2-Opt," *Journal of Heuristics* 5, 71–88.

T. Osogami. (1998). "Approaches to 3D Free-Form Cutting and Packing Problems and Their Applications: A Survey," *IBM Research Report*, RT0285.

M. Racer. (1994). "A Robust Heuristic for the Generalized Assignment Problem," *Annals of Operations Research* 50, 487–503.

C. R. Reeves. (1996). "Hybrid Genetic Algorithms for Bin-Packing and Related Problems," *Annals of Operations Research* 63, 371–396.

M. Yagiura, T. Yamaguchi, and T. Ibaraki. (1999). "A Variable Depth Search Algorithm for the Generalized Assignment Problem," In S. Voss (ed.), *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization.* Kluwer Academic Publishers.