Production Planning & Control

ISSN: 0953-7287 (Print) 1366-5871 (Online) Journal homepage: www.tandfonline.com/journals/tppc20

A new heuristic algorithm for the one-dimensional bin-packing problem

Jatinder N. D. Gupta & Johnny C. Ho

To cite this article: Jatinder N. D. Gupta & Johnny C. Ho (1999) A new heuristic algorithm for the one-dimensional bin-packing problem, Production Planning & Control, 10:6, 598-603, DOI: 10.1080/095372899232894

To link to this article: https://doi.org/10.1080/095372899232894

Published online: 15 Nov 2010.

PRODUCTION PLANNING & CONTROL, 1999, VOL. 10, NO. 6, 598–603

A new heuristic algorithm for the one-dimensional bin-packing problem

JATINDER N. D. GUPTA and JOHNNY C. HO

Keywords heuristic algorithm, bin-packing, optimization, parallel machine scheduling, empirical results

Abstract. We describe a new heuristic algorithm to solve the one-dimensional bin-packing problem. The proposed algorithm is optimal if the sum of requirements of items is less than or equal to twice the bin capacity. Our computational results show that effectiveness of the proposed algorithm in finding optimal or near-optimal solutions is superior to that of the FFD and BFD algorithms, specifically for those so called 'difficult' problems that require an optimal solution to fill most of the bins, if not all, exactly to capacity.

1. Introduction

We consider the well-known one-dimensional bin-packing problem consisting of packing a set of $N = \{1, 2, \ldots, n\}$ items into identical bins, each with a capacity C such that the capacity constraints are not violated. Associated with each item $i \in N$ is the capacity requirement t_i which must be supplied continuously only by one of the bins. The capacity requirements of items are additive. Our objective is to minimize the number of bins required to pack all the n items.

The one-dimensional bin-packing problem is important as it has numerous applications in the industry. Eilon and Christofides (1971) discuss its application to the loading of vehicles (or other containers) with consignments. Johnson *et al.* (1974) discuss three practical applications in computer science, which are table formatting, prepaging and file allocation. Brown (1971) provides an extensive discussion on industrial applications of one-dimensional bin-packing and other related problems. The solution procedures for the one-dimensional bin-packing problem have been used to develop heuristic sol-

Authors: J. N. D. Gupta, Department of Management, Ball State University, Muncie, IN 47306-0350, USA, e-mail: jgupta@bsu.edu, and J. C. H. Ho, Abbott Turner College of Business, Columbus State University, Columbus, GA 31907-5645, USA.

JATINDER N. D. GUPTA is a Professor of Management, Information and Communication Sciences, and Industry and Technology at the Ball State University, Muncie, Indiana, USA. He holds a PhD in Industrial Engineering (with specialization in production management and information systems) from Texas Tech. University. His current research interests include scheduling, planning and control, information systems education, technology and knowledge management, and organizational effectiveness. He is co-author of a textbook in Operations Research, and Dr Gupta serves on the editorial boards of several national and international journals. Recipient of an Outstanding Researcher award from Ball State University, he has published numerous research and technical papers in such journals as *Annals of Operations Research, Journal of Operational Research Society, International Journal of Information, Journal of Management Information Systems, Operations Research, IIE Transactions, Naval Research Logistics,* and *European Journal of Operational Research.*

JOHNNY C. HO is an Associate Professor of Operations Management and Operations Research at Columbus State University. He received a PhD in Management from DuPree College of Management of Georgia Institute of Technology in 1991. Recipient of a Faculty and Research Scholarship Award from Columbus State University in 1997, Dr Ho has published in *Annals of Operations Research, Naval Research of Logistics, European Journal of Operational Research,* and *Computers and Operations Research*, etc. His current research interests include scheduling, planning and control, and technology justification. He holds the Certified Quality Engineer title given by the American Society for Quality.

Production Planning & Control ISSN 0953–7287 print/ISSN 1366–5871 online
http://www.tandf.co.uk/JNLS/ppc.htm
http://www.taylorandfrancis.com/JNLS/ppc.htm

ution procedures for the identical parallel-machine scheduling problems (Parker 1995).

The problem of finding an optimum packing is known to be NP-hard (Coffman *et al.* 1978). Hence, heuristics are generally used to find near-optimal solutions. A comprehensive review of various heuristic algorithms is provided in a recent survey by Coffman *et al.* (1997). The most well-known heuristics are the first fit decreasing (FFD) algorithm (Eilon and Christofides 1971, Johnson *et al.* 1974) and the best fit decreasing (BFD) algorithm (Johnson *et al.* 1974). The FFD algorithm works as follows: it places items in non-increasing order of capacity requirements, then a packing is built by treating each item in succession, and adding it to the lowest-indexed bin in which it will fit without violating the capacity constraint. Eilon and Christofides (1971) find that the FFD performs reasonably well in their computational study. The BFD algorithm is similar to the FFD algorithm except that it selects the largest load bin that will fit without violating the capacity constraint. Coffman *et al.* (1978) show that the worst case for both FFD and BFD is $(11/9)k + 4$ bins, where k is the optimal number of bins.

The major weakness of the FFD and BFD is that its performance deteriorates for problems for which an optimal solution requires that most of the bins, if not all, be exactly filled. These problems are called 'difficult' problems. Hence, at optimality, a difficult problem has less total slack (sum of idle capacity of all bins) than a non-difficult problem. The FFDs and BFDs solutions often require more bins than that of an optimal solution for these problems (Coffman *et al.* 1978). This motivates us to develop a new heuristic which is more effective in dealing with these difficult problems.

In this paper, we develop a heuristic algorithm to solve the one-dimensional bin-packing problem and show that the proposed algorithm is optimal when the bin capacity C is at least half the total capacity required to pack all items, i. e. $2C \geq \sum_{i=1}^{n} t_i$.

The rest of the paper is organized as follows. Section 2 discusses the proposed algorithm. In Section 3, we use five difficult problems from Coffman *et al.* (1978) and Eilon and Christofides's (1971) papers to illustrate the effectiveness of the proposed algorithm. Section 4 discusses the computational results of our simulation study. Finally, section 5 concludes the paper and provides some fruitful directions for future research.

2. The proposed algorithm

It is easy to see that minimizing the number of bins is equivalent to minimizing the total slack remaining on the bins. Hence, we propose an algorithm which aims at minimizing the slack or idle time at each bin one at a time. In contrast to the FFD and BFD algorithms which find the bin where a particular job should be placed, the proposed algorithm, called MBS (minimum bin slack), is a bin-focus algorithm, i.e. it attempts to find a set of items that fits the bin capacity as much as possible. We start by describing an optimization algorithm to minimize the slack capacity of a given bin.

2.1. *Minimizing slack in one bin*

Utilizing the concept of lexicographic search and specific properties of the bin packing problem, we describe an optimization algorithm to select the items to be loaded in a single bin to minimize the slack capacity of the bin. In this algorithm, an item is placed in the bin in the decreasing order of its capacity requirement. If, at any time, an item is not able to be placed in a bin due to capacity limitation, lexicographically that item is placed in the bin which does not violate the capacity constraint. If no such item is found, backtracking is used to remove an item with the least capacity requirement (i.e., the highest numbered item among those already placed in the bin). This process is continued until either the bin capacity is completely used or there are no more items that can be placed in (or unplaced from) the given bin.

Let $\pi = (\pi(1), \ldots, \pi(j))$ be the j items already loaded in bin k where $t_{\pi(i)} \geq t_{\pi(i+1)}$ for $1 \leq i \leq j-1$. Further, let $P_\pi = \sum_{i=1}^{j} t_{\pi(i)}$. Then, the algorithm to minimize the slack in a single bin k is described as follows.

Algorithm L: lexicographic search optimization procedure

Input: t_i for $i = 1, \ldots, s$; C; k; $\sigma = (\sigma(1), \ldots, \sigma(s))$ such that $t_{\sigma(1)} \geq t_{\sigma(2)} \geq \ldots \geq t_{\sigma(s)}$; $\alpha = 0$; $j = 1$; and $\pi = (\pi(1), \ldots, \pi(j)) = (\sigma(1), \ldots, \sigma(j))$.

Step 1. If $P_\pi = C$, let $S_k = \pi$, and go to step 6; otherwise enter step 2.

Step 2. Find q such that $\sigma(q) = \pi(j)$. If $P_\pi < C$, set $j = j + 1$, enter step 3; otherwise enter step 4.

Step 3. If $P_\pi > \alpha$, set $\alpha = P_\pi$, and $S_k = \pi$ Enter step 4.

Step 4. If $q < s$, set $\pi(j) = \sigma(q+1)$ and return to step 1, otherwise enter step 5.

Step 5. If $j = 1$, enter step 6; otherwise, set $j = j - 1$, find q such that $\sigma(q) = \pi(j)$ and return to step 4.

Step 6. Items in S_k assigned to bin k is an optimal solution with minimum slack capacity of $C - \alpha$.

Even though the computational complexity of algorithm L is $O(2^s)$, our computational results show that the algorithm is quite efficient in solving problem instances with a large number of items and various values of bin capacity C.

2.2. *Finding minimum slack assignment*

We now describe a heuristic algorithm that uses algorithm L iteratively to find the minimum slack assignment for all items. To do so, we start with a single bin, $k = 1$ and use algorithm L to assign S_k items to the first bin with minimum slack. We then set $k = k + 1$ and apply algorithm L with the remaining number of items to minimize the slack for bin k. We continue this process until all items are assigned to a bin. The steps of this algorithm are as follows.

Algorithm MBS: minimum bin slack procedure

Input: t_i for $i = 1, \ldots, n$; C; $k = 1$; and $s = n$.

Step 1. Use algorithm L to find the assignment of items S_k to bin k. Let $\sigma = (\sigma(1), \ldots, \sigma(s))$ be the set of items not yet assigned to a bin and enter step 2.

Step 2. If $\sigma = \phi$, go to step 3; otherwise set $k = k + 1$, and return to step 1.

Step 3. The allocation of items $S = (S_1, S_2, \ldots, S_k)$ assigned to bin $(1, 2, \ldots, k)$ respectively, is the heuristic solution with minimum slack capacity of $kC - \sum_{i=1}^{n} t_i$.

The computational complexity of algorithm MBS is exponential as algorithm L requires $O(2^s)$ and must be used k times. However, our computational results show that the MBS algorithm is quite efficient in solving problem instances with a large number of items and various values of bin capacity C.

2.3. *Optimality for the special case*

The following theorem shows that MBS determines an optimal solution if $2C \geq \sum_{i=1}^{n} p_i$.

Theorem 1. *MBS finds an optimal solution if* $2C \geq \sum_{i=1}^{n} p_i$.

Proof. Because the application of algorithm L in MBS procedure minimizes the slack capacity of the first bin, it will pack all items in two bins if it is feasible. However, if the algorithm cannot pack the items in two bins, then three bins must be required to pack all items. Clearly, the algorithm will find this solution, as at each iteration, it optimizes the slack capacity of the bin being considered. □

3. Examples of difficult problems

In order to illustrate the effectiveness of the proposed algorithm, five difficult problems used in bin-packing literature (Eilon and Christofides 1971, Coffman *et al.* 1978) were solved using the proposed MBS algorithm, and compared to the FFD and BFD algorithms. In an optimal solution, the first four problems require all bins to be completely filled; whereas the last one requires, on average, the bins to be over 98% filled. The first problem is taken from Eilon and Christofides (1971: 268), while the remaining four problems are taken from Coffman *et al.* (1978: 5–6). Table 1 shows the number of bins used for each of the five problems by each of the three algorithms, FFD, BFD and MBS. As shown in table 1, the MBS algorithm finds optimal solutions in all five problems; while the FFD and BFD algorithms fail to generate an optimal solution for any one of the five problems.

4. Computational results

This section describes the computational tests which are used to evaluate the effectiveness of the FFD, BFD and the proposed MBS algorithms in finding good quality schedules. All algorithms were coded in FORTRAN language, and the tests were performed on an IBM compatible pentium 166 personal computer.

4.1. *Test problems*

Three classes of test problems were generated. In each class, item requirements are random integers from a uniform distribution [1, 100]. The number of items in a problem varied from 8 to 70. Problem hardness is likely to

Table 1. Solution of difficult problems.

Problem #	Capacity requirements	Bin capacity	No. of bins required		
			MBS	FFD	BFD
1	60, 50, 30, 20, 20, 20	100	2	3	3
2	3, 3, 2, 2, 2, 2	7	2	3	3
3	7, 5, 4, 4, 4, 3, 3, 3, 3, 3	13	3	4	4
4	17, 9, 7, 6, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4	17	5	6	6
5	44, 24, 24, 22, 21, 17, 8, 8, 6, 6	61	3	4	4

depend on the relationship between bin capacities and the total capacity requirements. For this reason, the different values of bin capacities are used in the computational experiments. To describe these bin capacities, let $P = \sum_{i=1}^{n} t_i$ and LB be the smallest integer greater than or equal to P/C, where C is the bin capacity. Then, the bin capacities considered are as follows.

- Class 1: $C = 0.501P$, $0.505P$ and $0.510P$.
- Class 2: $C = 300$ and a problem is generated such that $C * LB \geq P/(1-x)$, where $x = 0.005$, 0.01 and 0.015.
- Class 3: $200 \leq C \leq 400$ in increments of 25.

Class 1 deals with problems that meet conditions of the special case considered in theorem 1. In this class, an optimal solution of most problems would use only two bins, while a few problems may require the use of three bins. Because larger slack capacity and larger number of items are likely to produce optimal solutions by each algorithm, the number of items in these problems varied between 8 and 30. Class 2 problems test the MBS's performance on those problems that have x% or less average slack at optimality (considered difficult problems), where x is set at three levels: 0.5, 1 and 1.5. These problems were generated holding the bin capacity constant at $C = 300$. The number of items in class 2 problems varied from 10 to 70, which indicates that the number of bins required to solve a given problem also varied.

Class 3 problems evaluate the effectiveness of the three heuristic algorithms for varying values of bin capacity, regardless of relationships in class 1 and class 2 problems. The number of items for this class was kept at a constant $n = 35$.

For each parameter combination, 100 replications were created and solved by the FFD, BFD and MBS algorithms.

4.2. *Computational results*

For class 1 problems, table 2 shows the mean number of bins used, percentage of suboptimal solutions obtained, and the average CPU time (in s) required to solve a set of problems by each of the three algorithms. From table 2, it is seen that the percentage of sub-optimal problems for FFD and BFD algorithms range between 0–65% and 0–64%, respectively, with average values of 22.88% and 22.46%, respectively. The average number of bins used by the FFD, BFD and MBS algorithms is 2.25, 2.24 and

Table 2. Comparative evaluation of heuristics for class 1 problems.

		FFD			BFD			MBS		
n	Ratio	Mean	% Sub-optimal	CPU	Mean	% Sub-optimal	CPU	Mean	% Sub-optimal	CPU
8	0.501	2.78	56	0.00	2.75	53	0.00	2.22	0	0.05
	0.505	2.43	38	0.00	2.40	35	0.00	2.05	0	0.00
	0.510	2.23	19	0.05	2.20	16	0.05	2.04	0	0.05
9	0.501	2.71	61	0.15	2.71	61	0.10	2.10	0	0.10
	0.505	2.41	41	0.05	2.40	40	0.05	2.00	0	0.10
	0.510	2.19	19	0.05	2.19	19	0.00	2.00	0	0.05
10	0.501	2.69	63	0.05	2.71	65	0.05	2.06	0	0.10
	0.505	2.36	36	0.00	2.37	37	0.10	2.00	0	0.05
	0.510	2.12	12	0.15	2.12	12	0.10	2.00	0	0.10
11	0.501	2.66	65	0.05	2.65	64	0.10	2.01	0	0.10
	0.505	2.28	28	0.10	2.29	29	0.05	2.00	0	0.05
	0.510	2.06	6	0.15	2.06	6	0.15	2.00	0	0.15
15	0.501	2.49	49	0.10	2.46	46	0.05	2.00	0	0.05
	0.505	2.06	6	0.00	2.05	5	0.05	2.00	0	0.00
	0.510	2.01	1	0.10	2.01	1	0.15	2.00	0	0.15
20	0.501	2.26	26	0.00	2.27	27	0.00	2.00	0	0.10
	0.505	2.00	0	0.05	2.00	0	0.05	2.00	0	0.10
	0.510	2.00	0	0.05	2.00	0	0.15	2.00	0	0.05
25	0.501	2.13	13	0.05	2.13	13	0.05	2.00	0	0.05
	0.505	2.01	1	0.15	2.01	1	0.15	2.00	0	0.15
	0.510	2.00	0	0.10	2.00	0	0.10	2.00	0	0.10
30	0.501	2.09	9	0.05	2.09	9	0.05	2.00	0	0.05
	0.505	2.00	0	0.15	2.00	0	0.10	2.00	0	0.10
	0.510	2.00	0	0.05	2.00	0	0.05	2.00	0	0.05
Average		2.25	22.88	0.07	2.24	22.46	0.07	2.02	0	0.08

2.02, respectively. As expected, the MBS algorithm always finds optimal solutions for class 1 problems. The average CPU times for FFD, BFD and MBS are 0.07, 0.07 and 0.08 s, respectively. Hence, on average, the MBS takes about 14% longer to solve a problem than the other two algorithms.

For class 2 (difficult) problems, table 3 depicts the mean number of bins used by each algorithm, percentage of problems for which MBS solution is better than the FFD and BFD algorithms, and the average CPU time (in s) required to solve a set of problems by each of the three algorithms. On average, the solutions obtained by the proposed MBS algorithm are better than FFD and BFD algorithms in 44.92% and 43.50% cases, respectively. More difficult problems (those with the smallest slack) generally produce larger bound deviations. The number of bins used by FFD, BFD and MBS algorithms is 6.73, 6.71 and 6.28, respectively.

The mean CPU time for FFD, BFD, and MBS is 0.17, 0.18 and 0.36, respectively, indicating that on average, the MBS algorithm takes twice the CPU time to solve a problem than the FFD or BFD algorithms. Results in table 3, therefore, clearly demonstrate that the proposed MBS algorithm is relatively more effective in solving difficult problems than the FFD and BFD algorithms.

For class 3 problems, table 4 depicts the mean number of bins used by each algorithm, percentage of deviation of the heuristic solution from its lower bound (LB), and the average CPU time (in s) required to solve a set of problems by each of the three algorithms.

From the results in table 4, it is seen that the average bound deviations for FFD, BFD and MBS are 0.58%, 0.56% and 0.00%, respectively. Hence, the MBS proposed algorithm optimally solves every problem. As the number of jobs increases, the bound deviation decreases. The average number of bins used by FFD, BFD and MBS algorithms is 6.71, 6.70 and 6.66, respectively. The average CPU time for FFD, BFD and MBS is 0.16, 0.14 and 0.35 s, respectively.

5. Conclusions

In this paper, we proposed a new heuristic algorithm, called MBS, to solve the one-dimensional bin-packing problem. The MBS algorithm is particularly useful for solving two types of problems.

- Problems where the sum of requirements of items is less than or equal to twice the bin capacity. The

Table 3. Comparative evaluation of heuristics for class 2 problems.

		FFD			BFD			MBS	
n	% Slack	Mean	% MBS Better	CPU	Mean	% MBS Better	CPU	Mean	CPU
10	1.5	2.51	57	0.05	2.50	56	0.10	1.94	0.15
	1.0	2.56	62	0.00	2.56	62	0.05	1.94	0.00
	0.5	2.54	64	0.05	2.54	64	0.10	1.90	0.05
15	1.5	2.77	39	0.10	2.76	38	0.10	2.38	0.15
	1.0	2.90	49	0.10	2.89	48	0.05	2.41	0.15
	0.5	3.07	60	0.05	3.06	59	0.05	2.47	0.10
20	1.5	2.82	39	0.10	3.80	37	0.10	3.4	0.15
	1.0	3.88	43	0.10	3.84	39	0.05	3.45	0.20
	0.5	4.29	71	0.00	4.25	67	0.00	3.58	0.10
30	1.5	5.40	32	0.25	5.41	33	0.30	5.08	0.30
	1.0	5.55	43	0.25	5.56	44	0.20	5.12	0.35
	0.5	5.78	68	0.10	5.77	67	0.10	5.10	0.20
40	1.5	7.13	27	0.20	7.11	25	0.20	6.86	0.40
	1.0	7.17	39	0.15	7.14	36	0.25	6.78	0.25
	0.5	7.30	60	0.05	7.26	56	0.05	6.70	0.40
50	1.5	8.80	27	0.20	8.78	25	0.25	8.53	0.30
	1.0	8.83	36	0.30	8.81	34	0.20	8.47	0.35
	0.5	9.08	61	0.20	9.05	58	0.25	8.47	0.60
60	1.5	10.36	24	0.30	10.36	24	0.25	10.12	0.70
	1.0	10.44	34	0.20	10.44	34	0.25	10.10	0.55
	0.5	10.73	58	0.30	10.71	56	0.40	10.15	0.60
70	1.5	12.18	18	0.35	12.17	17	0.45	12.00	0.90
	1.0	12.04	23	0.20	12.03	22	0.20	11.81	0.80
	0.5	12.28	44	0.45	12.27	43	0.40	11.84	0.90
Average		6.73	44.92	0.17	6.71	43.50	0.18	6.28	0.36

Table 4. Comparative evaluation of heuristics for class 3 problems.

		FFD			BFD			MBS		
n	*C*	Mean	% Bound deviation	CPU	Mean	% Bound deviation	CPU	Mean	% Bound deviation	CPU
35	200	9.43	1.62	0.15	9.42	1.51	0.20	9.28	0.00	0.50
	225	8.36	0.84	0.10	8.35	0.72	0.10	8.29	0.00	0.40
	250	7.60	0.53	0.05	7.60	0.53	0.05	7.56	0.00	0.30
	275	6.93	0.14	0.10	6.93	0.14	0.05	6.92	0.00	0.30
	300	6.37	0.63	0.20	6.37	0.63	0.20	6.33	0.00	0.40
	325	5.95	0.34	0.25	5.95	0.34	0.20	5.93	0.00	0.40
	350	5.57	0.36	0.25	5.57	0.36	0.15	5.55	0.00	0.20
	375	5.18	0.39	0.05	5.18	0.39	0.10	5.16	0.00	0.30
	400	4.96	0.40	0.25	4.96	0.40	0.25	4.94	0.00	0.35
Average		6.71	0.58	0.16	6.70	0.56	0.14	6.66	0.00	0.35

MBS algorithm guarantees optimal solutions for this type of problem.

- Problems for which an optimal solution requires most of the bins, if not all, to be exactly filled.

Results of our computational experiments designed to test the effectiveness of the proposed MBS algorithm against FFD and BFD algorithms show that the MBS algorithm outperforms the FFD and BFD algorithms, especially with respect to the two types of problems mentioned above.

Several issues are worthy of future investigations. First, the development of the worst-case performance bounds for the MBS algorithm will be useful. Second, extension of results to the case of non-identical bins will be interesting and desirable. Third, the development of algorithms for multiple criteria optimization will enhance the utility of bin-packing algorithms to several production planning and control problems. Finally, extension of our results to more complex bin-packing environments, e.g. the inclusion precedence constraints between items, is important for application in industry.

References

Brown, A. R., 1971, *Optimum Packing and Depletion*. (New York, NY: American Elsevier).

Coffman, E. G., Garey, M. R., and Johnson, D. S., 1997, Approximation algorithms for bin packing, in D. S. Hochbaum (ed.) *Approximation Algorithms for NP-hard Problems* (Boston, MA: PWS Publishing), pp. 46–93.

Coffman, E. G., Garey, M. R., and Johnson, D. S., 1978, An application of bin-packing to multimachine scheduling. *SIAM Journal of Computing*, **7**, 1–17.

Eilon, S., and Christofides, N., 1971, The loading problem. *Management Science*, **17**, 259–268.

Johnson, D. S., Demers, A., Ullman, J. D., Garey, M. R, and Graham, R. L., 1974, Worst-case performance bounds for simple one-dimensional packing algorithm. *SIAM Journal of Computing* **3**, 299–326.

Parker, R. G., 1995, *Deterministic Scheduling Theory* (London, UK: Chapman and Hall).