2009 IEEE International Advance Computing Conference (IACC 2009)
Patiala, India, 6–7 March 2009

# Better-Fit Heuristic for One-Dimensional Bin-Packing Problem

A. K. Bhatia
National Bureau of Animal Genetic Resources
Karnal-132001 (India)
Email: avnish@lycos.com

M. Hazra, S. K. Basu*
Dept. of Computer Science, Banaras Hindu University
Varanasi-221005 (INDIA)
Email*: swapankb@bhu.ac.in

***Abstract*— This paper reports a study on *better-fit* heuristic for classical bin-packing problem, proposed in [1]. *Better-fit* replaces an existing object from a bin with the next object in the list, if it can fill the bin better than the object replaced. It takes $O(n^2m)$ time where $n$ is the number of objects and $m$ is the number of distinct object sizes in the list. It behaves as off-line as well as on-line heuristic with the condition of permanent assignment of objects to a bin removed. Experiments have been conducted on representative problem instances in terms of expected waste rates. It outperforms off-line best-fit-decreasing heuristic on most of the instances. It always performs better than the on-line best-fit heuristic.**

**Keywords:** Bin Packing Problem, Combinatorial Optimization, Heuristics.

## I. Introduction

Given a list of objects and their sizes, one-dimensional bin-packing problem (BPP) consists of finding a packing of the objects using a minimum number of bins of pre-specified capacity. Formally, given a list L of objects $o_i$ with sizes $s_i,\ i = 1, n$ and sufficient number of bins $B_1, B_2, B_3, \cdots$, each of capacity C, we want a minimum number ($z$) of bins to pack all the $n$ items.

Bin packing models several applications such as storage allocation of computer networks, assigning commercial breaks on television, copying a collection of files to magnetic tapes and floppy disks, etc. [2].

BPP is an NP-hard problem. Various heuristics are reported in the literature for solving on-line as well as off-line version of bin packing problem [3], [2].

On-line algorithms permanently assign the objects to a bin in the sequence of arrival. Three most popular on-line algorithms are described below. All the on-line heuristics have the initial condition that the first object is already packed in a bin.

*Next-Fit (NF)*: NF packs the next object into an active bin if it fits in the bin; otherwise the heuristic uses a new bin for the object, which is marked as the active bin.

*First-Fit (FF)*: FF packs the next object in the first bin in which it fits; otherwise FF packs the object in a new bin.

*Best-Fit (BF)*: BF packs the next object into a bin which is filled to the maximum extent but having adequate vacant space to pack the object; otherwise BF packs the object in a new bin.

Time complexity of NF is $O(n)$ while those of FF and BF is $O(n \log n)$. NF produces a worst packing of $2 * optimum$. FF and BF produce a worst packing of $1.7 * optimum$.

Off-line algorithms have all the objects available before the packing starts. Two common off-line algorithms are described below.

*First-Fit Decreasing (FFD)*: After sorting the list of objects in non-increasing order of sizes, the FFD packs objects according to the FF heuristic.

*Best-Fit Decreasing (BFD)*: After sorting the list of objects in non-increasing order of sizes, the BFD packs objects according to the BF heuristic.

Time complexity of FFD as well as BFD is $O(n \log n)$. Worst case performance of both the algorithms is $(11/9) * optimum$.

In this paper, we present *better-fit* heuristic for the one-dimensional bin packing problem initially proposed in [1]. Experiments have been conducted on benchmark problem instances from the literature [4]. The results obtained with *better-fit* heuristic have been compared with those obtained using best-fit-decreasing and best-fit heuristics representing off-line and on-line heuristics, respectively.

We organize the paper as follows: section II introduces the better-fit heuristic, section III describes experiments and the results obtained, and section IV contains our concluding remarks.

## II. Better-Fit Heuristic

A new heuristic for the one-dimensional bin packing called **better-fit** has been introduced in [1]. It has been derived from dominance criterion [5], [6] with importance given to a single object.

Better-fit heuristic packs next object from the list in the first bin that it can fill better than any of the existing objects in that bin. The object replaces one existing object from the bin. The replaced object is again packed with better-fit heuristic, starting with the first bin in the current solution. The process continues till a replaced object cannot perform better-fit in any of the bins. The last replaced object is then packed with the best-fit heuristic.

Each object in the given list is packed using better-fit heuristic. Algorithmic description of better-fit heuristic is provided below.

978-1-4244-2928-8/09/$25.00 

```
procedure betterfit(nextobject)
theobject ← nextobject;
extraobject ← false;
while (not extraobject) do
      for j=1 to z do
/* each bin B_j of the z bins in the current solution */
          replaced ← false;
          for k = 1 to |B_j| do
/* |B_j| is the number of objects in the bin B_j */
              if (theobject fills B_j better
              than object o_k) then
                 replace object o_k with theobject in B_j;
                 theobject ← o_k;
                 replaced ← true;
                 break;
              end if
          end for
      if (replaced) break;
      end for
if (all the bins have been tried) extraobject = true;
end while
bestfit(theobject)
/* pack 'theobject' with the best-fit heuristic */
end betterfit
```

While fitting the $i$th object using better-fit, it can replace at most $m$ objects in the bins, where $m$ is the number of distinct object sizes in the list. Summing over all possible values of $i$, gives the total maximum number of replacements as $mn\frac{(n-1)}{2}$. Hence, better-fit requires $O(n^2m)$ time.

With the objects arranged in decreasing order of sizes, no object can replace other objects and the performance of better-fit is equivalent to that of best-fit-decreasing. When the objects are arranged in increasing order of sizes, better-fit fills the initial bins with small objects that can't be replaced by larger objects. Thus the large objects require separate bins and it provides poor result. Better-fit provides the best performance in the average case when the object sizes are independent, identically distributed random variables.

## III. Experiments and Results

Experiments have been conducted on various problem instances of discrete uniform and bounded probability sampled distributions [4].

Instances of the discrete uniform distributions are denoted by $U\{h, j, k\}$, $1 \leq h \leq j < k$ and constructed with bin size $C = k$, uniform random integer object sizes $h \leq s_i \leq j$, $i = 1, n$, where $n$ is the number of objects. Instances of the bounded probability sampled distributions are denoted as $BS\{h, j, k, m\}$, $1 \leq h \leq j < k$ and constructed with bin size $C = k$ and $m \leq (j - h + 1)$ distinct sizes $h \leq s \leq j$ with non-uniform probability. The frequency of a distinct size $s$ in the list of $n$ objects is proportional to a randomly assigned weight $w(s) \in [0.1, 0.9]$.

Performance of an algorithm $A$ is measured by expected waste rate $(EW)$ [4], [7].

$$EW_A^n = E[A(L_n) - s(L_n)]$$

where $L_n$ is $n$ object list with object sizes chosen independently according to distribution $F$, $A(L_n)$ is the solution provided by the algorithm $A$ on the list $L_n$, $s(L_n) = [\sum_{a \in L_n} s(a)]/C$, C is the bin size, and $s(a)$ is the size of object $a$. $s(L_n)$ is a lower bound $L_1$ for the one-dimensional bin packing problem [5].

Performance of better-fit has been compared with off-line best-fit-decreasing heuristic. Better-fit also behaves as on-line bin-packing heuristic with removal of the condition of permanent assignment of arriving objects to a bin [8]. Therefore, it has also been compared with on-line best-fit heuristic.

Tables I to VIII show the results as expected waste rates of the three heuristics on various problem instances. Number of experiments for a heuristic on a class of problem instances has been fixed at twenty for the number of objects $(n)$ equal to 100 and 1000, ten for $n$ equal to 10000, and three for $n$ equal to 40000 and 100000.

Better-fit and BFD perform equally in the problem instances of type $U\{1, 35, 100\}$ with $n$ falling in the range 100 to 100000. Both the heuristics can reach the lower bound $L_1$ in these instances. BFD performs better than the better-fit heuristic in the instances $U\{1, 65, 100\}$ and $U\{1, 95, 100\}$. Better-fit performs poorly in the instances $U\{1, 95, 100\}$. Better-fit provides average waste of 152.5 in the instance of size 100000 while the result for BFD is 21.0. In these instances, the large objects cannot replace smaller objects using better-fit and occupy separate bins, thus providing poor results.

Better-fit and BFD perform equally in the problem instances of type $U\{1, 200h, 500h\}$ with $h$ =1, 2, 4, 8, 16, 32 (table IV). These instances mimic continuous uniform distribution $U(0, .4]$ with unit-size bin [4]. Both the heuristics can reach the lower bound $L_1$ except in one or two instances.

Best-fit heuristic performs poorly compared to better-fit heuristic in all the instances from uniform discrete distribution.

TABLE I
EXPECTED WASTE RATES IN PROBLEM INSTANCES $U\{1, 35, 100\}$

| Number of objects (n) | Better-Fit | Best-Fit-Decreasing | Best-Fit |
|---|---|---|---|
| 100 | 0 | 0 | 0.10 |
| 1000 | 0 | 0 | 0.50 |
| 10000 | 0 | 0 | 2.50 |
| 100000 | 0 | 0 | 19.33 |

TABLE II
EXPECTED WASTE RATES IN PROBLEM INSTANCES $U\{1, 65, 100\}$

| Number of objects (n) | Better-Fit | Best-Fit-Decreasing | Best-Fit |
|---|---|---|---|
| 100 | 0.45 | 0.20 | 1.60 |
| 1000 | 0.30 | 0.05 | 4.80 |
| 10000 | 0.10 | 0 | 27.20 |
| 100000 | 0 | 0 | 206.67 |

TABLE III

EXPECTED WASTE RATES IN PROBLEM INSTANCES $U\{1, 95, 100\}$

| Number of objects (n) | Better-Fit | Best-Fit-Decreasing | Best-Fit |
|---|---|---|---|
| 100 | 3.25 | 1.65 | 4.50 |
| 1000 | 11.00 | 4.90 | 18.45 |
| 10000 | 49.20 | 26.20 | 72.70 |
| 100000 | 152.50 | 21.00 | 211.00 |

TABLE IV

EXPECTED WASTE RATES IN PROBLEM INSTANCES $U\{1, 200h, 500h\}$, NUMBER OF OBJECTS = 10000

| h | Better-Fit | Best-Fit-Decreasing | Best-Fit |
|---|---|---|---|
| 1 | 0.1 | 0 | 7.8 |
| 2 | 0 | 0 | 8.1 |
| 4 | 0 | 0 | 8.3 |
| 8 | 0 | 0 | 8.7 |
| 16 | 0 | 0 | 8.5 |
| 32 | 0.3 | 0.2 | 8.9 |

In the problem instances from bounded probability sampled distributions (BS), better-fit provides superior result compared to both BFD and best-fit heuristics in all the instances. In the instance $BS\{1, 6000, 10000, 400\}$, $n = 100000$, better-fit results in average waste of 9 while the value for BFD is 58.66. Best-fit provides average waste equal to 255.66. Average waste rates using all the heuristics remain constant with the increase in the number of distinct object sizes, except in the problem instances $BS\{1, 4999, 10000, m\}$, $n = 10000$ (table VII).

TABLE V

EXPECTED WASTE RATES IN PROBLEM INSTANCES $BS\{1, 6000, 10000, 400\}$

| Number of objects (n) | Better-Fit | Best-Fit-Decreasing | Best-Fit |
|---|---|---|---|
| 100 | 0.55 | 0.85 | 5.50 |
| 1000 | 1.30 | 6.20 | 34.20 |
| 10000 | 4.00 | 29.33 | 156.67 |
| 100000 | 9.00 | 58.66 | 255.66 |

TABLE VI

EXPECTED WASTE RATES IN PROBLEM INSTANCES $BS\{\lfloor 625k/4 \rfloor + 1, \lceil 625k/2 \rceil - 1, 625k, 100k\}$, NUMBER OF OBJECTS = 10000

| $k$ | Better-Fit | Best-Fit-Decreasing | Best-Fit |
|---|---|---|---|
| 1 | 152.0 | 404.9 | 487.1 |
| 2 | 151.0 | 416.2 | 485.3 |
| 4 | 139.8 | 416.6 | 474.9 |
| 8 | 136.9 | 420.3 | 477.5 |
| 16 | 145.3 | 413.0 | 477.9 |

Figures 1, 2 and 3 show the computational time (seconds) taken by better-fit, BFD and best-fit, respectively against the number of objects from 1000 to 10000 with a step of 1000 on the instances $U\{1, 95, 100\}$. The time is average of 10 experiments on a personal computer with Pentium IV 3.2 GHz and Windows XP. A power curve has been fitted to the

TABLE VII

EXPECTED WASTE RATES IN PROBLEM INSTANCES $BS\{1, 4999, 10000, m\}$, NUMBER OF OBJECTS = 10000.

| $m$ | Better-Fit | Best-Fit-Decreasing | Best-Fit |
|---|---|---|---|
| 100 | 2.1 | 13.4 | 31.6 |
| 200 | 1.0 | 6.6 | 26.8 |
| 400 | 0.5 | 3.4 | 21.6 |
| 800 | 0.6 | 2.0 | 17.5 |
| 1600 | 0.4 | 0.9 | 19.4 |
| 3200 | 0.1 | 0.3 | 17.7 |

TABLE VIII

EXPECTED WASTE RATES IN PROBLEM INSTANCES $BS\{5001, 9999, 20000, m\}$, NUMBER OF OBJECTS = 10000.

| $m$ | Better-Fit | Best-Fit-Decreasing | Best-Fit |
|---|---|---|---|
| 100 | 130.2 | 429.9 | 475.5 |
| 200 | 129.2 | 411.0 | 471.1 |
| 400 | 138.9 | 425.8 | 480.9 |
| 800 | 133.7 | 422.2 | 476.0 |
| 1600 | 146.2 | 421.1 | 480.0 |
| 3200 | 136.7 | 416.1 | 475.3 |

scatter plot, empirically showing computational complexity of the heuristics. All the heuristics use $O(n^2)$ time. The time used by better-fit is higher by an order of magnitude compared with the time used by BFD and best-fit. Computational time for BFD includes the time required for sorting the objects according to decreasing order of their sizes using quick-sort algorithm.

Computational complexity of algorithms depends on the data structure used for representing packing [3], [9]. We have used a two-way linked-list to represent a bin-packing solution. Verticle list stores bins and horizontal list stores indices of objects in a bin. With this data structure, better-fit empirically shows a time complexity equivalent to BFD and best-fit heuristics.

Figure 4 shows the computational time in seconds consumed by better-fit heuristic against the number of distinct object sizes ($m$) for the instances $BS\{5001, 999, 20000, m\}$, $n = 10000$. A power curve of order $n^{0.7}$ fits in the scatter plot. It demonstrates that the rate of increase of computational time of better-fit decreases with the increase in the number of distinct object sizes.

## IV. CONCLUSION

We have explored the performance of better-fit heuristic for classical bin-packing problem. It can be used as off-line as well as on-line heuristic with removal of the condition of permanent assignment of objects to bins. Experiments on representative problem instances show supremacy of the heuristic over the off-line best-fit-decreasing heuristic on most of the instances. Better-fit always performs better than the on-line best-fit heuristic. The three heuristics empirically show equivalent time complexity. Better-fit can also be used in fully dynamic bin-packing problem [10], a version of dynamic bin-packing [3], [11] with migration of objects allowed.

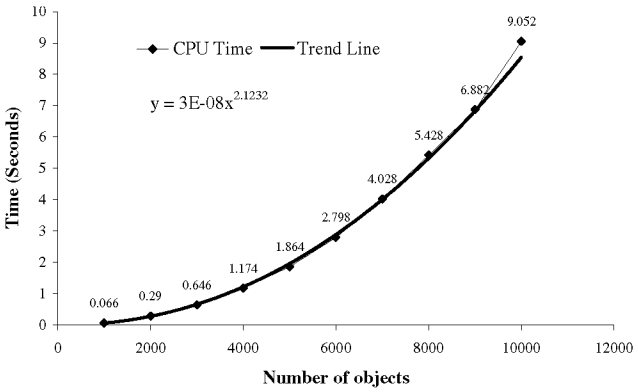


Fig. 1. Computational time (seconds) for better-fit on the problem instances $U\{1, 95, 100\}$

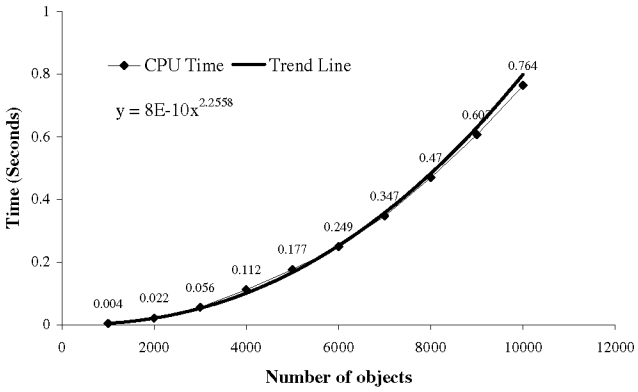


Fig. 2. Computational time (seconds) for best-fit-decreasing on the problem instances $U\{1, 95, 100\}$

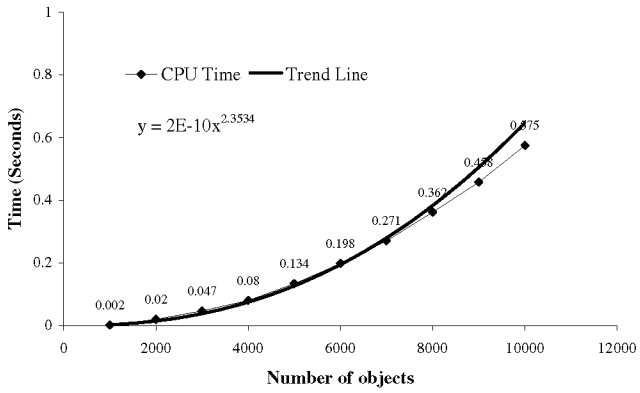


Fig. 3. Computational time (seconds) for best-fit on the problem instances $U\{1, 95, 100\}$

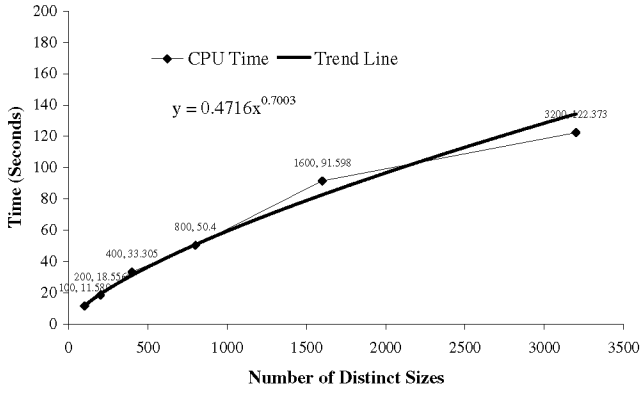


Fig. 4. Computational time (seconds) for better-fit on the problem instance $BS\{5001, 9999, 20000, m\}$, $n = 10000$ with varying number of distinct object sizes ($m$).

## REFERENCES


[1] A. K. Bhatia and S. K. Basu, "Packing bins using multi-chromosomal genetic representation and better-fit heuristic," in *Proc. of the 11th International Conference on Neural Information Processing (ICONIP 2004), Calcutta, India, LNCS 3316*, N. R. Pal et al., Ed., 2004, pp. 181–186.

[2] E. G. Coffman,Jr, J. Csirik, and G. J. Woeginger, "Approximate solutions to bin packing problem," URL=citeseer.nj.nec.com/coffman99approximate.html, 1999.

[3] E. G. Coffman,Jr, G. Galambos, S. Martello, and D. Vigo, "Bin packing approximation algorithms: combinatorial analysis," in *Handbook of Combinatorial Optimization*, D.-Z. Du and P. Pardalos, Eds. Kluwer Academic Publishers, 1998.

[4] D. L. Applegate, L. S. Buriol, B. L. Dillard, D. S. Johnson, and P. W. Shor, "The cutting-stock approach to bin packing: theory and experiments," in *Proceedings of the 5th Workshop on Algorithm Engineering and Experiments (ALENEX03), Maryland*, R. E. Ladner, Ed. SIAM, 2003, pp. 1–15.

[5] S. Martello and P. Toth, "Lower bounds and reduction procedures for the bin packing problem," *Discrete Applied Mathematics*, vol. 28, pp. 59–70, 1990.

[6] E. Falkenauer, *Genetic Algorithms and Grouping Problems*. John Wiley & Sons, 1998.

[7] J. Csirik, D. S. Johnson, C. Kenyon, P. W. Shor, and R. R. Weber, "A self organizing bin packing heuristic," in *Proc. of 1999 Workshop on Algorithm Engineering and Experimentation, Berlin, LNCS 1619*, M. Goodrich and C. C. McGeoch, Eds. Springer-Verlag, 1999, pp. 246–265.

[8] L. Epstein and A. Levin, "A robust APTAS for the classical bin packing problem," in *Proc. of the 33rd International Colloquium on Automata, Languages and Programming (ICALP2006)*, 2006, pp. 214–225.

[9] M. L. Fredman, D. S. Johnson, L. A. McGeoch, and G. Ostheimer, "Data structures for traveling salesmen," *Journal of Algorithms*, vol. 18, pp. 432–479, 1995.

[10] Z. Ivković and E. L. Lloyd, "Fully dynamic algorithms for bin packing: being (mostly) myopic helps," *SIAM J. Comput.*, vol. 28, no. 2, pp. 574–611, 1998.

[11] W.-T. Chan, T.-W. Lam, and P. W. H. Wong, "Dynamic bin packing of unit fractions items," in *Proc. of ICALP2005, LNCS 3580*, L. Caires et al., Ed. Springer-Verlag, 2005, pp. 614–626.