Archive of SID

Mathematical Sciences *Vol. 4, No. 3 (2010) 253-266*

Application of Genetic Algorithm for the Bin Packing Problem with a New Representation Scheme

N. Mohamadi[1]

Department of Mathematics, Islamic Azad University, Shirvan Branch, Shirvan, Iran.

Abstract

The Bin Packing Problem (BPP) is to find the minimum number of bins needed to pack a given set of objects of known sizes so that they do not exceed the capacity of each bin. This problem is known to be NP-Hard [5]; hence many heuristic procedures for its solution have been suggested. In this paper we propose a new representation scheme and solve the problem by a Genetic Algorithm. Limited computational results show the efficiency of this scheme.

Keywords: Bin Packing, Heuristics, Genetic Algorithm.

© 2010 Published by Islamic Azad University-Karaj Branch.

1 Introduction

In one dimensional bin packing problem the objects have one dimension. Object's value is weight, size, cost or time. The term "bin" here is in fact a generic name which could stand for a "container", as in the "transportation" context, "work stations " in industrial assembly lines (line balancing), "a space in time" in scheduling, or a "surface area", as in metal working, for example.

The one dimensional bin packing problem can be stated as follows: We are given a set of n objects each with a given weight (or size)($w_i > 0$) . We want to place these

[1]E-mail Address: Mohamadi_37@yahoo.com

www.SID.ir

Archive of SID

objects into bins of a given capacity C ($C > w_i$) so that the total number of bins needed is minimized. This problem has many practical applications: Trucks are to be loaded with different items. The aim is to use as few trucks as possible to carry the loads without exceeding the capacity of each truck. Another example is where tubes or cables are to be cut from quantities of standard length C. We want to use as few tubes or cables of standard length as possible to meet the demand. The same idea is used in metal working where steel sheets of different sizes must be cut from "master" sheets. Yet another example is in scheduling, where tasks of varying duration must be allocated using the least number of machines or processors. In fact applications of this problem are so pervasive and mathematically appealing that even American Mathematical Society (AMS) [15] and National Institute of Standards and Technology (NIST) [16] have internet sites devoted to it.

There are other variations and extensions of this problem. The most obvious is the two dimensional bin packing problem, where in addition to weight, the volume of the objects too, have to be taken into account. That is instead of one set of constraints, we have two sets of constraints [7],[12].

A variation on this two dimensional problem is the "strip packing" problem. The problem is to pack a set of n rectangles into an open ended bin of fixed width C so that the height of the bin is minimized. The rectangles must not overlap. There are also other extensions such as three dimensional and maximum value problems, for which the reader is referred to many good references on this subject such as [1], [4],[6],[9] and [11].

In this paper we concentrate only on the one dimensional bin packing problem. We first present a mathematical formulation of the problem as an integer program. Then we discuss some solution procedures and representation schemes so far suggested for the problem. The new representation scheme and a genetic algorithm utilizing this scheme are presented in sections 5 and 6. Finally, some computational results are given in section 5, followed by summary and conclusions.

www.SID.ir

Archive of SID

2 Mathematical formulation and solution procedures

Mathematically the one dimensional bin packing problem can be formulated as [6], [3]:

$$\min z = \sum_{j=1}^{n} y_j$$

$$\begin{aligned} s.t: \ & \sum_{i=1}^{n} w_i x_{ij} \le C y_j \ , \ \forall j \\ & \sum_{j=1}^{n} x_{ij} = 1 \qquad , \ \forall i \\ & x_{ij}, \ y_j = 0, 1 \qquad , \ \forall i, j \end{aligned}$$

where,

$$y_j = \begin{cases} 1 & \text{if bin j is used} \\ 0 & \text{otherwise} \end{cases}$$

$$x_{ij} = \begin{cases} 1 & \text{if object i is placed in bin j} \\ 0 & \text{otherwise} \end{cases}$$

In this model the objective minimizes the total number of bins used. Constraints (1) ensure that the weights of objects placed in bin j, do not exceed the capacity of the bin. Note that here we assumed that all the bins have the same capacity, C, but in general this need not be the case. Finally, constraints (2) ensure that each object is placed only in one bin.

The above program can be solved using standard procedures for the solution of integer programs, such as branch and bound; but since the number of cases to be considered by enumerating all possible combinations is , the computational efforts needed to solve this problem is prohibitive for large n's. In fact it is known [5] that the bin packing problem is NP-Hard. Hence, heuristic procedures such as *next fit*, *first fit*, *first fit decreasing*, *best fit*, *best fit decreasing*, to name a few, are proposed for this problem.

www.SID.ir

Archive of SID

Since these methods are easily implemented, they are usually embedded in a genetic algorithm to enhance its performance. We have adopted this practice here, too. In addition their optimal values provide upper bounds, and through proven inequalities lower bounds, for the problem. The packing used by *first fit* or *best fit* uses no more than $\frac{17}{10}OPT + 2$bins, while *first fit decreasing* or *best fit decreasing* uses at most $\frac{11}{9}OPT + 4$ bins [17]. A better bound for *first fit* decreasing, given in [13] is $\frac{11}{9}OPT + 1$. Sometimes it is assumed that a list L of objects to be packed is available in advance. Sometimes, however, items become available as they are being packed. The algorithms using this dynamic characteristic are usually referred to as *on-line* algorithms.

3 Genetic algorithm

GA is a *meta-heuristic* originally proposed by Holland [8]. Since then it has evolved into a powerful method for solving many hard combinatorial optimization problems a list of which can be found in many references, see e.g., [6]. The general steps of the algorithm can be outlined as follows:

```
Generate an initial population
While an stopping criteria is met, do
        - Choose pairs for mating
        - Perform cross-over to generate off-springs
        - Evaluate the fitness of new off-springs
        - Generate a new population
end while
```

A key element in a *GA* is the generation of a population whose components are called *chromosomes*, a term borrowed from *Genetics*. A chromosome is in fact a coded representation of a solution; an n-dimensional array each component of which is termed a

www.SID.ir

Archive of SID

gene.

4 representation scheme

For the bin packing problem three representation schemes have been proposed [6], bin-based representation, object-based representation, and group-based representation:

4.1 bin-based representation

In bin-based representation, a chromosome has a fixed length equal to n, the number of objects. Each bin is represented by a gene. The position of each gene in a chromosome, then indicates the object number placed in the bin represented by that gene. For example, the chromosome 2 3 4 1 5 indicates that object 1 is placed in bin 2, object 2 is placed in bin 3, object 3 is placed in bin 4, and so on, (figure 1).

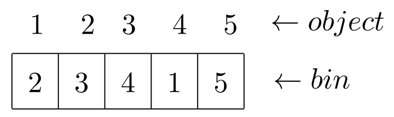

Figure 1: Bin-based representation of a chromosome

The advantage of this representation is that each chromosome has a fixed length which makes it suitable for application of genetic operators. A draw back of this method, as pointed out by Falkenauer [2], is that it creates many redundant solutions, which exponentially increases with the number of bins, which in turn decreases the efficiency of the GA. In addition, it might create infeasible solutions.

4.2 object-based representation

In the object-based representation a chromosome represents a permutation of objects. Then the chromosome is partitioned based on number of objects placed in bins. For example, if we have six objects, then a permutation of the numbers 1 to 6 is 1 2 3 4 5 6. A partition of this chromosome could be 1 2 | 3 4 5 | 6, which indicates that objects 1

www.SID.ir

Archive of SID

and 2 are placed in bin 1, objects 3, 4, and 5 are placed in bin 2, and object 6 is placed in bin 3. This representation has some draw backs too. For example, the following chromosomes with the indicated partitions all represent the same solution as above:
2 1 | 4 5 3 | 6 ,   3 4 5 | 2 1 | 6 ,   6 | 2 1 | 5 4 3, etc.

4.3 group-based representation

The representations mentioned so far are *item oriented*, hence, are not suitable for the bin packing problem. The group-based representation proposed by Falkenauer [2],[6] is in fact group oriented, and provides an appropriate means of representing this problem, since it is not important for us where the objects are placed, rather how they are put together. It consists of two parts. The first part is similar to the bin-based representation. The second part provides an encoding of the bins used in a one-to-one correspondence to genes. For example, the chromosome 4 1 2 5 2 3 means that object 1 is placed in bin 4, object 2 in bin 1, objects 3 and 5 in bin 2, object 4 in bin 5 and object 6 in bin 3. Using letters to represent the bins, then this chromosome can be represented as DABEBC, where $D = \{1\},\ A = \{2\},\ B = \{3, 5\},\ C = \{6\},\ and\ E = \{4\}$ (figure 2)

1	2	3, 5	6	4	← *object*
D	A	B	C	E	← *bin*

Figure 2: Group-based representation for the example

The advantage of this scheme is that genes represent both objects and groups. The group part is actually used in genetic operations, while the object part is merely used to identify the elements of a group. The disadvantage, however, is that the chromosomes have a variable length.

5 A new representation scheme

Now we describe a new representation scheme which does not have redundancies, has a fixed length, and has a compact representation, hence combines the nice features of

www.SID.ir

Archive of SID

all previous schemes while it does not have their draw backs.

we assume each bin has n cells, one cell per object. If object i is placed in bin j, then the weight of object i is placed in cell i of bin j. The number of bins needed for a particular assignment is placed in an additional cell at the beginning, cell zero.

To describe this representation, consider the following example. Suppose we have four objects with weights 2, 2, 4, and 4 respectively, assigned to three bins as shown below (figure 3.)

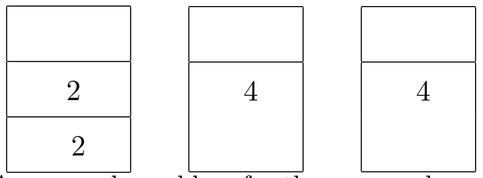

Figure 3: An example problem for the proposed representation

The corresponding chromosome can be defined as:

$$3 \mid 2\ 2\ 0\ 0 \mid 0\ 0\ 4\ 0 \mid 0\ 0\ 0\ 4 \mid 0\ 0\ 0\ 0$$

Note that:

1) the number in the first place indicates the number of bins needed for the assignment (3 bins).

2) The next 4 digits indicate that the objects with weights 2 and 2 are placed in the first bin.

3) The next 4 digits indicate that the third object with a weight of 4 is placed in bin 2.

4) The third set of 4 digits corresponds to the third bin and shows that the 4^{th} object with a weight of 4 is placed in that bin.

5) Finally, the last bin is empty

This representation has the nice features of the group-based representation, and the additional desirable property of having a fixed length. However, since in general, chromosomes have a length equal to $1+n^2$,then for large n we will need a large amount of memory to store them. Fortunately we can use a more compact way of representing

www.SID.ir

Archive of SID

this assignment which uses chromosomes of length $n+1$ instead of $1+n^2$. If we only keep track of the cell numbers in the above example, we would have the sequence 0 1 2 7 12 , which has the length $n+1$, with $n=4$ in this case. In the first cell, cell 0, the number of bins is stored. The next n numbers indicate the cell numbers in which the objects are placed.

Now if we denote the cell number as c, and the total number of objects as n, then we obtain a and b by $a = c\ mod(n)$; (or $c = bn + a$) i.e., a is the remainder in dividing the cell number by the total number of objects.
If the remainder is zero (c is divisible by n), then:

The object number is n and the bin number is b .

Otherwise:

The object number is a and the bin number is $1+b$.

In the above example, for cell 7 we have: $7 = 1 \times 4 + 3$,
the remainder is 3 and $b = 1$, so $b+1 = 2$, thus object 3 is placed in bin 2.

For cell 12, we have: $12 = 3 \times 4 + 0$,
the remainder is 0, and $b = 3$, thus object 4 is placed in bin 3, and so on.

Note that the above representation is indeed independent of the numbering and preserves the group property. To see this more clearly consider again the above example with the given 4 objects. Assume again that they are placed in 3 bins but in a different configuration, as shown below.

Object	1	2	3	4
Weight	2	4	4	2

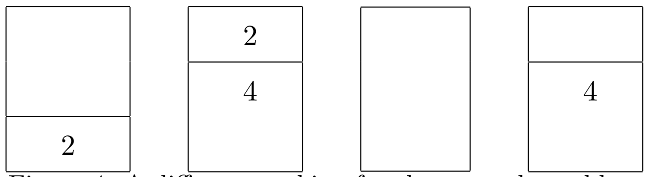

Figure 4: A different packing for the example problem

www.SID.ir

Archive of SID

The corresponding chromosome would be

$$3 \mid 2\ 0\ 0\ 0 \mid 0\ 4\ 0\ 2 \mid 0\ 0\ 0\ 0 \mid 0\ 0\ 4\ 0$$

with the compact representation 0 1 6 8 15. Using the proposed representation scheme, it is easily verified that the object numbers 1, 2, 4, and 3 are placed in bins 1, 2, 2, and 4, respectively as follows:

For cell 1 we have $1 = 0 \times 4 + 1$:

The remainder is 1 and $b = 0$, so $b + 1 = 1$, thus object 1 is placed in bin1.

For cell 6, we have $6 = 1 \times 4 + 2$:

The remainder is 2 and $b = 1$, so $b + 1 = 2$, thus object 2 is placed in bin2.

For cell 8, we have $8 = 2 \times 4 + 0$:

The remainder is 0, and $b = 2$, thus object 4 is placed in bin 2.

For cell 15, we have $15 = 3 \times 4 + 3$:

The remainder is 3, and $b = 3$, so $b + 1 = 4$, thus object 3 is placed in bin 4.

6 The algorithm

It has five components:

- Initialization
- Selection
- Solution Generation
- Evaluation
- Termination. [10]

The new proposed chromosome representation is suitable for the application of the genetic operators. Starting from an initial population, new off-springs (solutions) can be generated and their fitness evaluated. The procedure is repeated until a stopping criteria, such as the number of iterations, or convergence of the solutions, is met.

www.SID.ir

Archive of SID

In our implementation, the initial population is generated using the *first fit (FF)* heuristic, discussed in section 2. Solutions are randomly selected from the population with a *linear bias*, that is the probability of selecting a solution is linearly dependent on the position of that solution in the population, such that a better solution has a higher probability of being selected. The *bias* is usually a number between 1.5 and 3, and means that the probability of selecting the best solution is 1.5 - 3 times higher than the probability of selecting the worst solution [10].

We define the *neighborhood* of a given solution (chromosome) as those solutions (chromosomes) which differ only in the location of one object. For example we consider the following chromosomes to be neighbors, 2 | 0 0 1 | 1 3 0 | 0 0 0 and 2 | 1 0 1 | 0 3 0 | 0 0 0 . The neighbors of a given solution (new chromosomes) are generated using the *mutation* operator. Among these new solutions then the best one which is better than the current solution, is added to the population. In order to reduce the computational time needed for generating all neighbors of a solution in each iteration, we can only generate a predetermined number of neighbors as suggested in [10].

Evaluation of a solution is usually performed by considering either the number of bins used, or by the remaining capacities in the bins. In the first instance, the smaller the number of bins used, the better the solution is. In the latter, a solution with the smaller sum of unused capacities of bins is considered to be superior.

Finally, the algorithm terminates when all the solutions use the same number of bins.

7 Computational results

To test the efficiency of the proposed scheme and the algorithm, we tested the 100 sample problems, with known optimal solutions, listed in [14]. The algorithm was coded in $C^{++}$and run on Pentium III 600 with 128 MB RAM. The results for a few sample

www.SID.ir

Archive of SID

problems with 4, 9, 10, 20, 40, 50 and 120 objects are given in table 1, below. Note that in all problems listed GA achieves the optimal solution. This was actually the case for all the 100 problems tested. This good result is partially due to the fact that we start from a "good" initial solution, and that we stop only when the difference between the best and worst solutions is zero. Relaxing this rule will reduce the computation time but at the cost of reducing the accuracy of the solution. In our case, the time needed to solve the problem with 120 objects was less than a second, which justifies our stopping rule.

set	The number of objects	Capacity of bins	FF	GA	optimum
1	120	150	63	49	49
2	50	100	25	23	23
3	40	70	21	19	19
4	20	45	11	10	10
5	9	14	7	6	6
6	4	6	3	2	2
7	10	20	6	6	6

Table 1: The results for some test problems from [14]

In above table, after calculating correlation coefficient, the result is: $r \cong 1$.
If we put $X = FF, Y = GA$ and use:

$$r = \frac{n\sum_{i=1}^{n} x_i y_i - (\sum_{i=1}^{n} x_i)(\sum_{i=1}^{n} y_i)}{\sqrt{n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2}\sqrt{n\sum_{i=1}^{n} y_i^2 - (\sum_{i=1}^{n} y_i)^2}}$$

then we have:

$$\sum_{i=1}^{7} x_i = 136 \qquad \sum_{i=1}^{7} y_i = 115 \qquad \sum_{i=1}^{7} x_i y_i = 4255 \qquad \sum_{i=1}^{7} x_i^2 = 5250$$

$$\sum_{i=1}^{7} y_i^2 = 3467 \qquad (\sum_{i=1}^{7} x_i)^2 = 18496 \qquad (\sum_{i=1}^{7} y_i)^2 = 13225 \qquad r = 0.996 \cong 1$$

www.SID.ir

Archive of SID

8 Summary and conclusions

In this paper we considered the one dimensional bin packing problem. We briefly reviewed some heuristic procedures and representation schemes suggested for the problem. We then proposed a new scheme which did not have redundancies, had a fixed length, and had a compact representation, hence combined the nice features of all previous schemes while it did not have their draw backs. The new proposed chromosome representation was suitable for the application of the genetic operators. To test the efficiency of the proposed scheme and the algorithm, we tested the 100 sample problems. Our limited computational results attests the efficiency of the proposed algorithm.

9 Appendix

A. Weights for the 120 object problem

97; 57; 81; 62; 75; 81; 23; 43; 50; 38; 60; 58; 70; 88; 36; 90; 37; 45; 45; 39; 44; 53; 70; 24; 82; 81; 47; 97; 35; 65; 74; 68; 49; 55; 52; 94; 95; 29; 99; 20; 22; 25; 49; 46; 98; 59; 98; 60; 23; 72; 33; 98; 80; 95; 78; 57; 67; 53; 47; 53; 36; 38; 92; 30; 80; 32; 97; 39; 80; 72; 55; 41; 60; 67; 53; 65; 95; 20; 66; 78; 98; 47; 100; 85; 53; 53; 67; 27; 22; 61; 43; 52; 76;64; 61; 29; 30; 46; 79; 66; 27; 79; 98; 90; 22; 75; 57; 67; 36; 70; 99; 48; 43; 45; 71; 100; 88; 48; 27; 39;

B. Weights for the 50 object problem

12; 45; 90; 92; 30; 40; 42; 51; 17; 18; 12; 25; 32; 47; 41; 63; 60; 51; 81; 73; 51; 40; 15; 20; 11; 12; 9; 13; 91; 92; 75; 44; 71; 32; 35; 16; 29; 44; 39; 58; 16; 23; 9; 91; 26; 42; 84; 98; 50; 62

C.Weights for the 40 object problem

12; 13; 11; 40; 22; 60; 61; 63; 11; 10; 19; 31; 32; 37; 25; 14; 21; 38; 51; 59; 40; 45; 54; 62; 59; 40; 13; 31; 17; 20; 26; 36; 15; 12; 9; 10; 27; 31; 55; 40

D. Weights for the 20 object problem

17; 19; 12; 11; 17; 18; 17; 4; 5; 21; 10; 23; 37; 32; 29; 40; 41; 30; 21; 11

www.SID.ir

Archive of SID

E. Weights for the 9 object problem

5; 7; 3; 5; 12; 11; 10; 11; 9

F. Weights for the 4 object problem

2; 2; 4; 4

G. Weights for the 10 object problem

14; 15; 12; 2; 4; 8; 13; 19; 20; 7

Acknowledgment

I would like to thank the editor and the anonymous referees for their helpful comments.

References

[1] Coffman Jr., Garey M., Johnson D. (1984) "Approximation Algorithms for Bin-Packing, " An updated survey, in Algorithm Design for Computer Systems Design, Ausiello G., Lucertini M. , Serafini P. (eds.), Springer-Verlag, New York, 49-106.

[2] Falkenauer E. (1998) "Genetic algorithms and grouping problems, " J. Wiley & Sons, 25-103.

[3] Falkenauer E. (1995) "Tapping the full power of genetic algorithms through suitable representation and local optimization: Application to bin packing," in Evolutionary algorithms in management applications, Springer Verlag, Berlin,167-182.

[4] Falkenauer E., DelChambre A. (1992) "A genetic Algorithm for Bin packing and Line Balancing," in Proc. Of the IEEE 1993 International Conference on Robotics and Automation, V.2, Piscataway, NJ, 1186-1192.

[5] Gary M., Johnson D. (1979) "Computers and Intractability: A Guide to Theory of Np-completeness, " W.H Freeman, New York.

[6] Gen M., Cheng R. (2000) "Genetic algorithms and engineering optimization," J. Wiley & Sons, 61-69.

www.SID.ir

Archive of SID

[7] Hayek J. E., Moukrim A., Negre S. (2008) "New resolution algorithm and pretreatments for the two-dimensional bin-packing problem," Computers & Operations Research, 35, 10, 3184-3201.

[8] Holland J. (1975,1992) "Adaptation in natural and artificial systems, University of Michigan press," Ann Arbor, MI, MIT press, Cambridge, MA.

[9] Labb M., Laporte G., Martello S. (2003) "Upper bounds and algorithms for the maximum cardinality bin packing problem," European Journal of Operational Research, 149, 3, 490-498.

[10] Pargas R. P., Jain R. (1993) "A parallel stochastic optimization algorithm for solving 2D bin packing problems," ICNN, 18-25.

[11] Peeters M.,Degraeve Z. (2006) "Branch-and-price algorithms for the dual bin packing and maximum cardinality bin packing problem," European Journal of Operational Research, 170, 2, 416-439.

[12] Pisinger D., Sigurd M. (2005) "The two-dimensional bin packing problem with variable bin sizes and costs," Discrete Optimization, 2, 2, 154-167.

[13] Yue M. (1991) "A simple proof of the inequality FFD(L)$\leq$ (11/9)OPT(L) + 1, for all L, for the FFD bin-packing algorithm, " Acta. Math. Appl. Sinica 7, 321-331.

[14] http://people.brunel.ac.uk/ mastjjb/jeb/orlib/binpackinfo.html

[15] www.ams.org

[16] www.nist.gov

[17] www.cs.gsu.edu/ cscskp/Algorithms/Np/node11htm/

www.SID.ir