DS&OR Lab
Fachgebiet Wirtschaftsinformatik
Fakultät Wirtschaftswissenschaften
Universität Paderborn

Metaheuristic

Metaheuristics Project: Container packing

by
Ahmad Hashemi
6785702
haschemi.ahmad@gmail.com

supervised by
Jun. Prof. Dr. Kevin Tierney

Summer Semester 2017

Contents

1 Weight-Coded GA

To solve the problem with metaheuristic, the codding method in [Rai99] is used. In this article, each solution candidate (chromosome) is codded with a technique: A solution to the optimization problem is represented by a vector of weights that modifies the original problem solution. The method in the mentioned article is presented to solve the container packing problem merely by consideration of the weight constraint and the capacity of homogeneous containers. Therefore, it is upgraded according to the case of this project as below:
To consider the box orientation in a container, dimensions of a box are labeled with numbers 1,2 and 3 defined as an array, which introduce smallest, middle and largest dimension of the box, respectively. In another word, these weights and orientation arrays introduce implicitly a candidate solution. Then it will be converted to the final solution using a statistical method and packing algorithm.
Each candidate solution for the packing container problem in weight encoded GA is expressed as a vector:

$$\vec{WD} = \{w_1d_1, w_2d_2, ..., w_jd_j\},$$

where every weight w_j and dimension d_j is related to the item j. To calculate the weight a so called *uniform additive biasing* method is used. At first, the *relative value* of each box is calculated with this formula:

$$r_j = \frac{p_j}{v_j * m_j}$$

where:
p_j represent the absolute box profit
v_j volume of box
w_j mass of box
During the generation of a GA's initial population, weights w_j get uniformly distributed random values in a certain range $[-\gamma\bar{r}, \gamma\bar{r}]$ with $\bar{r} = \frac{1}{n}\Sigma_{j=1}^{n} r_j$ being the average relative item value. The parameter γ controls how strong item weights can be biased and is therefore called *biasing strength*.
Care must be taken to guarantee always positive biased item values, which means :

$$\gamma \leq \frac{min_{j=1,...,n} r_j}{\bar{r}}$$

As mentioned before, the array d_j as a combination of numbers 1, 2, 3 represents the box orientation by packing in the container.
Finally an initial random population with a symmetrical range of values for the GA

is generated whose median is matched with the original problem.
At the end of each generation, vector $\vec{WD}$ is converted to a permutation of the solution sorted by the order of boxes in order to be packed in containers. In another word, each chromosome expresses the packing priority and placement of each box in a container. Then the weight is updated with this formula:

$$r_j^{\text{update}} = r_j + wj.$$

Then the orientation array d_j will be replaced with the real dimension of the box. The boxes are packed in the containers using the packing algorithm and the heuristic as bellow:

Algorithm 1 procedure Heuristic

```
for all containers i do
    Occupied size of containers $\longleftarrow 0$
    Occupied weight of containers $\longleftarrow 0$
    for all unpacked items j sorted according to decreasing $r_j^{\text{update}}$ do
        if $\text{box}_j, \text{d}_j$ can be packed in container i then
            pack $\text{box}_j$ in to container i
            update occupied size
            update occupied weight
            update fitness
            remove $\text{box}_j$ from $r_j^{\text{update}}$
        end if
    end for
end for
```

2 Genetic Algorithm

This chapter discusses the configuration of genetic Algorithm.

2.1 Applied GA operators

2.1.1 Elite

This operator allows a percentage of the best solutions with in the population to pass through to the next generation without being modified. For example, a percentage of 5% with a population of 100, means that the top individuals form part of the next generation. This means that any future generation is at least as good as the current generation. This helps to protect good individuals.

2.1.2 Corssover double point

two chromosomes are selected as parents. Two points are selected to determine a center section in each parent, this is swapped between them.

2.1.3 SwapMutate

The Swap Mutate operator, traverses each gene in the population and, based on the specified probability swaps one gene in the chromosome with another. The aim of this operator is to provide mutation without changing any gene values.

2.1.4 Customized Operator –Gene Mutate Operator

As previously mentioned, each gene represents implicitly a box that contains a weight and an orientation array. Using previous operators, dimension arrays will not be evolved along execution of the GA. To solve this problem, a new mutation operator is developed, by which the combination of arrays will change according to the mutation percentage for each gene in a chromosome.

2.1.5 The Constraint Termination

Two different ways are implemented to terminate the GA execution. first, the GA is adjusted to evolved for 400 generations. Whereas, In the second method this number of iteration is crossed if the solution quality was not satisfactory, in which the termination constraint is defined based on obtaining a better solution compared with the first method.

3 Packing Algorithm

Heuristic algorithms proposed in different literatures have mostly a high implementation complexity due to not using a metaheuristic. Moreover, they donâ€™t contain the constraints used in this project. Therefore, the proposed method is extracted from the so called Block-and-layer method in [MBO08]. Based on this algorithm, boxes are packed starting from the origin along the X-axis till there is no more space for a new box. Then, the next block is arranged in the same way as the previous one but taking the box with the largest width as the block border. It goes on till there is no more space for a new block in that layer. At this point a layer is completed and the first block of the next layer will be packed according to the box with the largest height in the previous layer. This process goes on till there is no more space to pack a new layer.

4 Literature Review

Different heuristics and metaheuristics are already proposed to solve the container packing problem. However, the problem generality is decreased in most of them. Meanwhile those using direct encoding [RK98], have chromosomes containing genes, which indicate which box should be packed in which container. In this case, infeasible solutions must be handled by introducing a repair algorithm or adding a penalty term. On the other hand, some algorithms benefit the order based encoding, in which each chromosome is a permutation of all boxes. The explicit solution is obtained using a first- or next-fit algorithm, such that the items are packed in containers till the capacity or dimension constraint is violated. Recombination and mutation operators such as crossover and swap mutation must be used to generate new chromosome that contain valid permutation again [Rai99].

5 How to compile and run the code

Our approach is implemented in C# using Microsoft Visual Studio Enterprise 2015 under Windows. To compile and run the code, simply open the solution file Meta.CPacking.sln and run the project using F5 (debug) or Shift + F5 (no debug). A copy of Microsoft Visual Studio Enterprise 2015 can be downloaded at Dreamspark UPB. Another method to run the code, is to open a command line window and enter the path to the compiled orbdar.exe with the additional parameters. Microsoft .NET Framework v4.5.2 must be installed as a requirement. In this project, The Genetic Algorithm Framework (GAF) is use to implement a GA. GAF can be installed through NuGet , and is free to use. Many popular genetic operators are provided within the GAF, in addition, external operators can be defined and added to the GA process pipeline as required by the consumer. The developed GA operator inherited from the operator main class of GAF.

6 Experimental Investigation

Firstly, it should be pointed out that the generated solutions could not have been uploaded to the leaderboard due to an error, even though the results are correct in format and meet the constraints. Unfortunately, no resolution was found even after contacting Mr. Kuske.
There are different parameters studied in this project, whose initializations influence the performance, explained later. In order to increase diversity as much as possible, the biasing strength value is chosen as great as possible, i.e. $\gamma = \frac{min_{j=1,\ldots,n} r_j}{\bar{r}}$

The population size in each instance is obtained by the product of the number of boxes and containers. The elitism is set to 5 percent, by which the best result is reserved in each generation for smallest instances out of 10 boxes and 2 containers. The percentage of crossover, the mutation and the gene mutate are set to 80, 20 and 50 percent respectively.
The program runtime depends on different factors such as the quality of initial population, definition of the termination function and the probability of jumping to the Basin of attraction of global extremum. However, it can be claimed that the algorithm is fast enough and the result of first generation appears in less than one minute even for large dataset of 500 and 1000 boxes.
The results in comparison with other results on leaderboard turn out to be in an acceptable range. It is recommended to improve the tuning of parameters in future works.

Bibliography

[MBO08] Wissam F Maarouf, Aziz M Barbar, and Michel J Owayjan. A new heuristic algorithm for the 3d bin packing problem. *Innovations and Advanced Techniques in Systems, Computing Sciences and Software Engineering*, pages 342–345, 2008.

[Rai99] Günther R. Raidl. The multiple container packing problem: A genetic algorithm approach with weighted codings. *SIGAPP Appl. Comput. Rev.*, 7(2):22–31, March 1999.

[RK98] Günther Raidl and Gabriele Kodydek. Genetic algorithms for the multiple container packing problem. In *Parallel Problem Solving from Natureâ€"PPSN V*, pages 875–884. Springer, 1998.