Bin Packing Problem (January 2018)

Vaasu Gupta, Siddharth Shailendra

***Abstract*— In the bin packing problem, objects of different volumes must be packed into a finite number of bins or containers** each of volume V in a way that minimizes the number of bins used. **In computational complexity theory, it is a combinatorial NP-hard problem. The decision problem (deciding if objects will fit into a specified number of bins) is NP-complete.**

There are many variations of this problem, such as 2D packing, linear packing, packing by weight, packing by cost, and so on. They have many applications, such as filling up containers, loading trucks with weight capacity constraints, creating file backups in media and technology mapping in Field-programmable gate array semiconductor chip design.

The bin packing problem can also be seen as a special case of the cutting stock problem. When the number of bins is restricted to 1 and each item is characterized by both a volume and a value, the problem of maximizing the value of items that can fit in the bin is known as the knapsack problem.

I. Introduction

Bin Packing is one of the very classical combinatorial optimization problems studied in computer science. Packing problems are optimisation problems that are concerned with finding a good arrangement of multiple items in larger containing regions (objects). The usual objective of the allocation process is to maximise the material utilisation and hence to minimise the “wasted” area. The objective of packing problems is to increase the usability of main object and obtain the layout pattern with the minimum loss of value.

The trim loss is the unused area in the main object.

This is of particular interest to industries involved with mass-production as small improvements in the layout can result in savings of material and a considerable reduction in production costs. The development of an algorithm to solve an industrial packing problem clearly must consider the complexity of the problem, determined by the geometry of the objects and the constraints imposed. In addition, the algorithm must be easy to adapt to the present competitive market with frequent product introductions, changing product designs and "shorter time to market" strategy. The flexibility achieved by manual packing is no longer a competitive solution due to high labour and liability costs.

Conventional automated packing methods do not offer this flexibility, since they are mostly tailored to a particular packing task. This calls for a new approach to packing problems, which implements automation, but also maintains the flexibility which is offered by manual composition of packing layouts.

In Cloud, improving the system performance and optimizing the organizational resources has caught the interest of researchers and users of all computing systems. This management leads to minimizing energy through virtual machine placement and translates into saving a significant amount of money and natural resources.

The two-dimensional packing problem (2D-BPP) frequently encountered in the wood-, glass- and paper industry.

The three-Dimensional Bin Packing Problem (3D-BPP) has a high practical relevance in modern industrial processes such as plane cargo management, warehouse management, pallet loading, and container ship loading.

II. Literary Survey

Bin Packing studies date back at least to the 1950’s and it appeared as one of the prototypical NP-hard problems in the book of Garey and Johnson. In the year 1965, Gilmore proposed stack building approach for cutting problem optimization which opens a way for research in the field of packing problems Mingozzi, and Toth in 1979 for solving the liquid loading problem. This algorithm used the dominance criterion and assumed that those liquids will not mix with each other.

In year 1990, Gehring et al developed heuristics by considering practical constraints like stack restrictions, box orientations and container the stability
along with container loading

In the year 1995, Chen et al. provide a mixed integer programming formulation to solve the 3D-BPP without orientation restriction. In later of the year 1995, Bischoff et al. propose a pallet loading heuristic for non-identical cartons where the stability of the pallet is considered.

In 2002 where strips and layers are created so that the 3D-BPP can be decomposed into smaller sub-problems.

In the year 2003, Faroe et al. provide a heuristic for packing cartons into a minimum number of identical containers based on guided local search, however, no carton rotation is allowed.

Lesh et al. (2004) gives a method to find strip packing solutions using an exact branch and bounds exhaustive search. a new heuristic algorithm using rules to mimic human intelligence to solve the 3D-BPP.

In the year 2007, Puchinger and Raid consider the two-dimensional BPP, where a new integer linear programming formulation is proposed and solved using CPLEX.

In the year 2007, Kevin J. Binkley, proposed a four corners heuristic which is specifically designed to increase the search efficiency for application in evolutionary algorithms (EAs) applied to two-dimensional packing problems using two different EA algorithms

It reviews 2D & 3D packing problems and their solution using genetic algorithm. It deals with bin packing problem classifications, genetic operators, and other strategies
Most recently, set of disjunctive box tower using greedy algorithm were generated and arranging the box tower on the floor of the container Traditional mathematical techniques take more computational time to solve than Meta heuristic algorithms so three intelligent algorithms compared for simple problem and complex problem with same condition maintained in all three algorithms. The objective is increases the value of box packing. Each box is assigned a weight value (finical value) and fitness function will select the best fit based on these financial weights.
Borgfeldt and Gehring proposed a hybrid GA based on the layer concept for solving the container loading problem with several practical constraints.
Bin Packing is also a good case study to demonstrate the development of techniques in approximation algorithms. The earliest ones are simple greedy algorithms such as the First Fit algorithm, analysed by Johnson which requires at most $1.7 \cdot OPT+1$ bins and First Fit Decreasing which yields a solution with at most 11/9OPT+4 bins. Later, Fernandez de la Vega and Luecker developed an asymptotic PTAS by introducing an item grouping technique that reduces the number of different item types and has been reused in numerous papers for related problems. De la Vegaand Luecker were able to find a solution of cost at most$(1+\varepsilon)OPT+O(1/\varepsilon^2)$ for Bin Packing and the running time is either of the form $O(n^f(\varepsilon))$ if one uses dynamic programming or of the form $O(n \cdot f(\varepsilon))$ if one applies linear programming techniques. A big leap forward in approximating bin packing was done by Karmarkar and Karp in 1982, who provided an iterative rounding approach for the mentioned linear programming formulation which produces a solution with at most $OPT+O(\log^2 OPT)$ bins in polynomial time, corresponding to an asymptotic FPTAS.
Rao (1990) defined the optimization as the act of obtaining the best solution among the available ones by satisfying the packing constraint and in the given circumstance.
Optimization of bin packing problem is not a new area. Optimization techniques have been proposed and analyzed since 1960's. The various names given by the researchers for this bin packing problems are container loading problem, cutting stock problem, knapsack problem, strip packing problem, scheduling problem, air cargo loading problem, pallet loading problem, etc. An extensive literature survey has been done and discussed below based on 1D bin packing approaches, 2D bin packing approaches and 3D bin packing approaches.
Three dimensional bin packing problem considers all three dimensions namely length, breadth and height of bin for packing into a container. Falkenauer and Delchambre (1992) used GA for finding the optimal cost solution for packing and line balancing. In bin packing, a set of bins to be packed into a set of containers such that the number of containers utilized have been minimum and inturn the cost. In line balancing problem, processing the given job has to be done in minimum number of workstations by satisfying the constraints namely any workstation can do any operation
Alphabetical genetic encoding scheme were used. Guntram (1992) proposed an approximation algorithm for solving container loading problem using layer-by-layer approach. Chen et al (1995) considered the problem of loading containers with cartons of non-uniform size, change in carton orientation and overlapping constraint. Due to strong NP-hardness of the problem, integer programming model can only solve instances where a few cartons to be packed optimally. George (1996) investigated the case of multiple-containers loading for pipe packing. They followed three strategies for packing. The first strategy was sequential packing of pipes into a container, the second was preallocation approach, in which rules were formulated for packing pipes and the third was simultaneous packing of pipes into multiple-containers. Thomas et al (1996) applied fuzzy logic and genetic concept for dynamic dual-bin packing problem to maximize the number of bins packed of varying weights. As the problem was dynamic, each bin was examined only once
Hopper and Turton (1997) reviewed the application of GA to the 2D and 3D bin packing problem. They dealt with bin packing problem classifications, genetic operators and other strategies. They concluded that the GA can be used for a variety of optimization problems and proved that the complexity of the problem increases with increase in geometrical parameters and constraints. Lai and Xue (1998) developed an algorithm to pack several customer orders into multi-containers. Dimension of the cargo bins were limited, with the objective to pack more number of bins into a container. Bins were packed along the lengthwise from left - bottom most corner of the container and without overlapping. Chien and Wu (1999) developed a heuristic module which divides the empty container space into columns and layers for determining the 29 container loading pattern. This was achieved by ranking the bins based on volume and type. Different loading procedures were followed to fill the remaining empty space and thereby the container volume was utilized maximally

III. ALGORITHM

Online Algorithms - In these items arrive one at a time (in unknown order), each must be put in a bin, before considering the next item. the ONLINE algorithm doesn't know what's coming down the pipe, or even how long the pipe is.

Next Fit - When processing the next item, see if it fits in the same bin as the last item. Start a new bin only if it does not. (Incredibly simple to implement) If M is the number of bins in the optimal solution, then Next Fit never uses more than 2M bins.
Example. Given the set of items S = {4, 8, 5, 1, 7, 6, 1, 4, 2, 2} and bins of size 10, pack the items into as few bins as possible using Next Fit.

The results of using the Next fit algorithm are shown in the figure below.

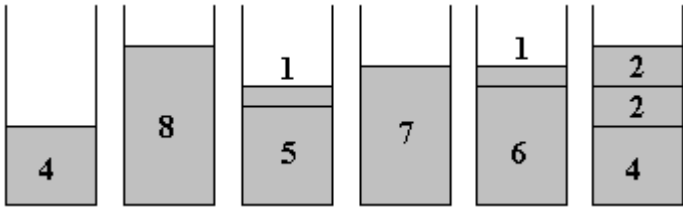


Figure 1 - Packing under Next Fit

A total of six bins are required to pack the items under Next Fit (compared to a best packing that requires 4 bins). This algorithm is clearly wasteful of bin space, since the bins we close may not be very full (such as the first bin in the example). However, it does not require memory of any bin except the current one. This might be important if memory costs are a consideration, or if the staging area is limited
If memory or staging is not an issue, we should be able to improve the performance of the algorithm by considering previous bins that might not be full. In effect, we expect that if we pack individual bins better, the overall solution will be better. Formalizing this idea leads to the First Fit algorithm

First Fit - Next Fit can be easily improved: rather than checking just the last bin, we check all previous bins to see if the next item will fit. Start a new bin, only when it does not.(First Fit easy to implement in O(N^2) time) First Fit never uses more than 2M bins, if M is the optimal.

Example. Given the set of items *S* = {4, 8, 5, 7, 6, 1, 4, 2, 2, 1} and bins of size 10, pack the items into as few bins as possible using First Fit.

The solution is shown here.

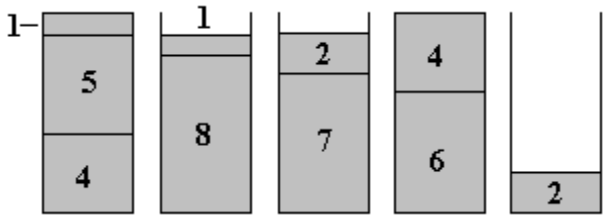


Figure 2 - Packing under First Fit

This algorithm requires memory of the available space in previous bins, and that these bins be available for further packing. Of course, there are variations to this idea such as limiting the memory or staging to *M* bins or looking ahead *d* items in the steam.

We might try to extend the idea in the First Fit algorithm by placing items in the best possible bin, that is, in the bin that will be filled the most by placing the item in it. In many cases we would fill the bin exactly. The general idea is to obtain the best global packing by getting the best local packing of individual bins.

Best Fit - This strategy places the next item in the tightest spot. That is, put it in the bin so that smallest empty space is left.(Also easy to implement in O(N log N) time). Best Fit never uses more than 1.7M bins if M is optimal.
Example. Given the set of items *S* = {4, 8, 5, 1, 7, 6, 1, 4, 2, 2} and bins of size 10, pack the items into as few bins as possible using Best Fit.

This packing is shown in Figure 3.

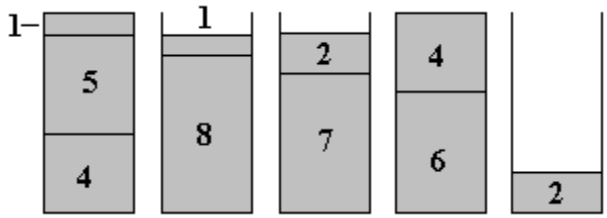


Figure 3 - Packing under Best Fit

Offline Algorithms – In these all items given upfront, we can view the entire sequence upfront.

First Fit Decreasing - Here the items are ordered into their non increasing order, and then in this order the next item is always packed into the first bin where it fits (easy to implement in O(N log N) time) if M is the optimal number of bins, then FFD never uses more than 11M/9 + 4 bins.
Example. Given the set of items *S* = {4, 8, 5, 1, 7, 6, 1, 4, 2, 2} and bins of size 10, pack the items into as few bins as possible using the BP-FFD .

The items in sorted order are *S* = {8, 7, 6, 5, 4, 4, 2, 2, 1, 1}. Applying the First Fit algorithms gives the result below:

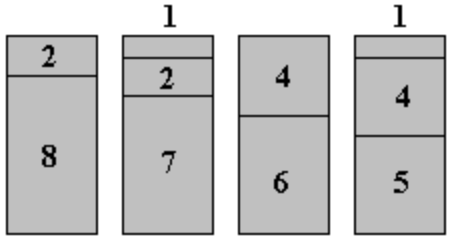


Figure 4 - Packing under First Fit Decreasing

The BP-FFD algorithm gives the same results. In both instances the packings improved when sorting was allowed.

Genetic Algorithm -

In a genetic algorithm, a population of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem is evolved toward better solutions. Each candidate solution has a set of properties (its chromosomes or genotype) which can be mutated and altered; traditionally, solutions are

represented in binary as strings of 0s and 1s, but other encodings are also possible.[2]

The evolution usually starts from a population of randomly generated individuals, and is an iterative process, with the population in each iteration called a *generation*. In each generation, the fitness of every individual in the population is evaluated; the fitness is usually the value of the objective function in the optimization problem being solved. The more fit individuals are stochastically selected from the current population, and each individual's genome is modified (recombined and possibly randomly mutated) to form a new generation. The new generation of candidate solutions is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.

A typical genetic algorithm requires:

1. a genetic representation of the solution domain,
2. a fitness function to evaluate the solution domain.

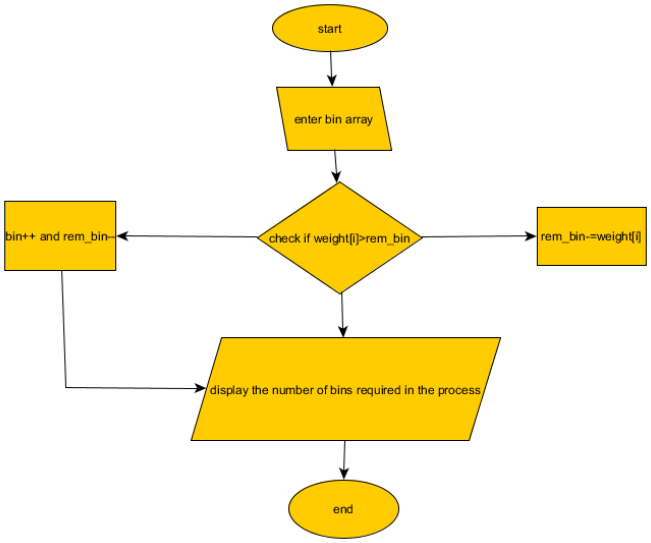
A standard representation of each candidate solution is as an array of bits.[2] Arrays of other types and structures can be used in essentially the same way. The main property that makes these genetic representations convenient is that their parts are easily aligned due to their fixed size, which facilitates simple crossover operations. Variable length representations may also be used, but crossover implementation is more complex in this case. Tree-like representations are explored in genetic programming and graph-form representations are explored in evolutionary programming; a mix of both linear chromosomes and trees is explored in gene expression programming.

Once the genetic representation and the fitness function are defined, a GA proceeds to initialize a population of solutions and then to improve it through repetitive application of the mutation, crossover, inversion and selection operators.

IV. Pseudocode and Flow Diagrams

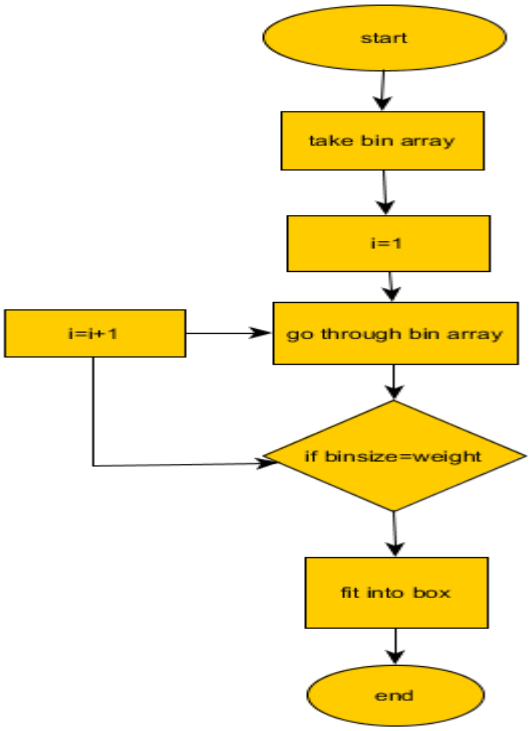
1)Next Fit-

```
For all objects i= 1 to n
   If object i fits in current bin
      Put object i in current bin.
   Else
      Create new bin, assign it the current bin, pack object i
   End if
End for
```



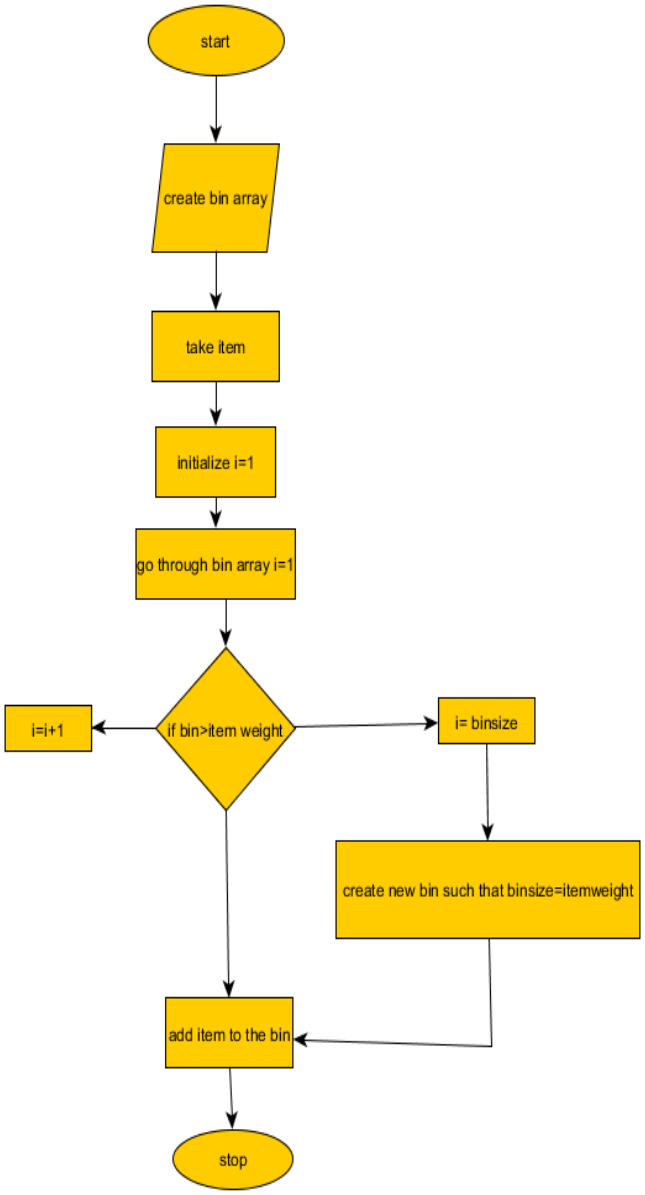
2)First Fit-

```
For all objects i=1 to n
   For all bins j=1 to n
      If Object i fits in bin j
         Put object i in bin j.
         Break the loop and put the next object.
      end if
   end for
   If Object i did not fit in any available bin
      Create new bin and put object i
   end if
end for
```



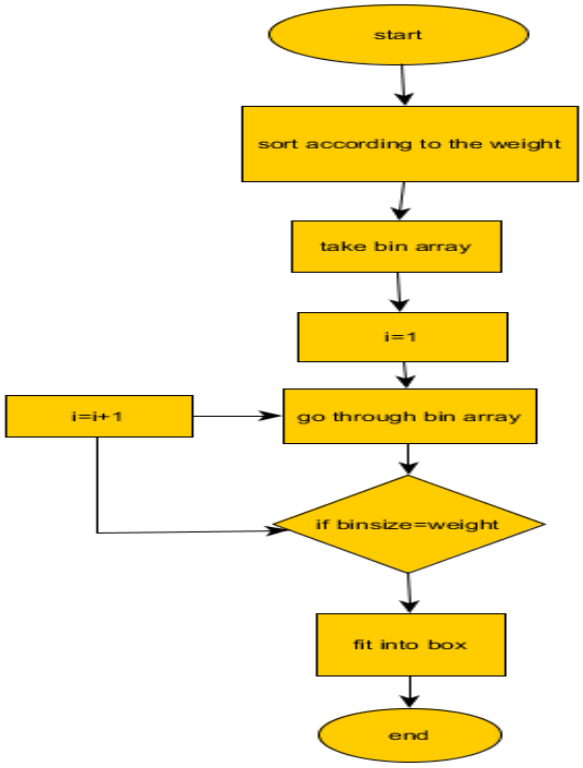
3)Best Fit-

For all objects i=1 to n
  For all bins j = 1 to n
    If object I fits in j
      Calculate the remaining capacity of bin
    End if
  End for
  Pack object i in bin j, where j is the bin with min remaining capacity after adding the object
  If no such bin exists then open a new bin and add object
End for



4)First Fit Decreasing-

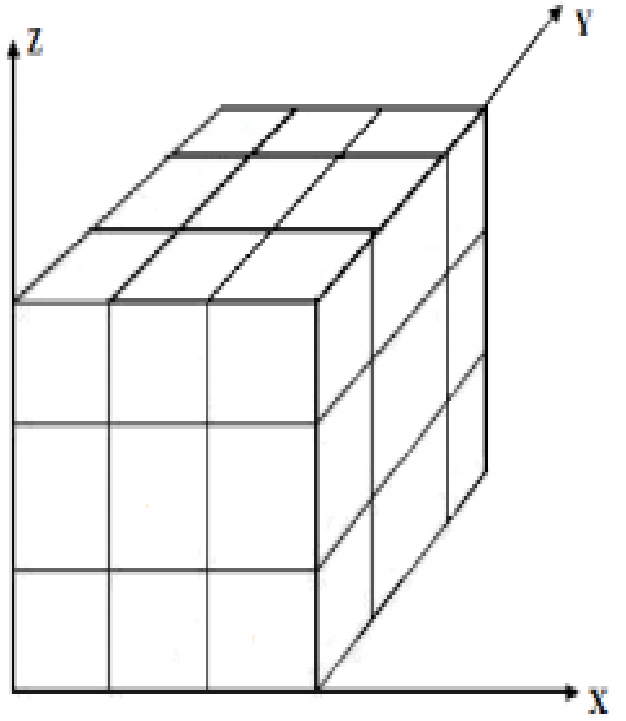
For all objects i=1 to n
  Sort objects in decreasing order
  For all bins j=1 to n
    If Object i fits in bin j
      Put object i in bin j.
      Break the loop and put the next object.
    end if
  end for
  If Object i did not fit in any available bin
    Create new bin and put object i
  end if
end for



V. Problem Formulation

In this section, we detail our evolutionary approach by presenting the adopted mathematical formulation and the evolutionary algorithm based on the following assumptions.

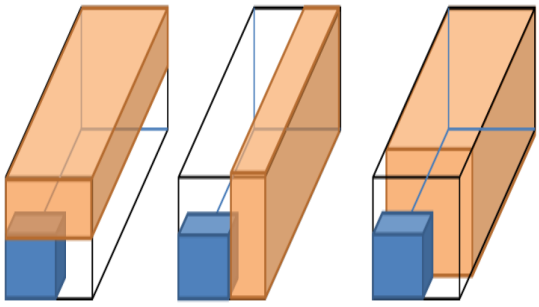
Assumptions In our work we suppose that:
The containers are identical (weight,shape, type) and each is waiting to be delivered to its destination.
Initially containers are stored at the platform edge or at the vessel.
A container can be unloaded if all the floor which is above is unloaded
The containers are loaded from floor to ceiling
We are given a set of cuboids container localised into a three dimensions cartesian system



Input parameters
Let's consider the following variables:
i: Container index,
n1: Maximum containers number on the axis X
n2: Maximum containers number on the axis Y
n3: Maximum containers number on the axis Z
Nc floor: Maximum containers number per floor, Nc floor= n1*n2
Nfloor: Total number of floors
Nc floor (j) : the containers number in the floor j
Ncmax: Maximum containers number, with N' = n1.n2.n3
Nc: the containers number

VI. Proposed Methodology

- Outline of the algorithm
- Initialization
  - [1] Assume we need to pack m boxes into N containers.
  - [2] We start with initializing a population of n chromosomes - suitable solutions for the problem. Each chromosome consists of 2 parts: box packing sequence (BPS) and container loading sequence (CLS). Box packing sequence is permutations of {1,...,m} that defines an order the boxes are packed and container loading sequence is permutation of {1,...,N} containers that defines an order in which the containers are loaded.
  - [3] Four special chromosomes are created in a population by sorting BPS in descending order according to boxes volume, length, width and height. The rest of chromosomes are generated randomly.
- Packing
  - [4] In next step each chromosome is being mapped into a packing solution.
  - [5] A concept of empty maximal spaces (EMS) is used to represent free space in containers. Essentially, EMS is a list of largest empty cubic space available for packing not contained in any other EMS. Empty maximal spaces are represented with their maximum and minimum coordinates. EMS is displayed in picture below.



  - [6]
  - [7]
  - [8] We prefer EMS 1 to EMS 2 if minimum coordinates of 1 are smaller than minimum coordinates of EMS 2.
  - [9] Once we have chosen EMS, we have to choose box orientation. When a box has several possible placements in one EMS, than the one with smallest margin is selected.
- Evaluation
  - [10] After a packing procedure for current population was done, we evaluate a fitness of each chromosome using measure of free space in the containers.
- Selection

[11] A new population is created by performing **Selection** and **Crossover** on existing population.
[12] E chromosomes with the best fit within the population are selected directly to next population, so the best solution found can survive to end of run. And the rest chromosomes are selected for tournament.
[13] To perform tournament we randomly select two chromosomes and compare them. Winner goes to mating pool as parent in hope that the better parents will produce better offsprings.

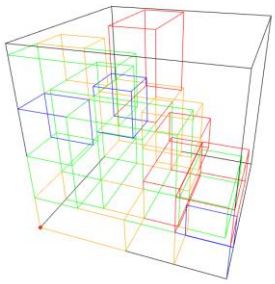
- Crossover and mutation
  [14] When promising parents are selected it's time for crossover.
  [15] Crossover creates a new offsprings using parents chromosomes. Crossover is made in hope that new chromosomes will have good parts of old chromosomes and maybe the new chromosomes will be better. However, need to leave some part of population to survive to next generation.
  [16] Each pair of chromosomes in mating pool goes to the next population with some probability (it's a parameter) and otherwise two their offsprings are generated with crossover.
  [17] We randomly select two cutting points i and j. First offspring gets genes i+1:j of first parent and the rest missing genes it gets by sweeping second parent circularly starting from j+1 and checking wether it has appeared in offspring. The second offspring is created by changing roles of first and second parent.
  [18] After a crossover is done, we perform mutation on new offsprings with some probability (it's a parameter). It's done to prevent falling all solutions in the population into a local optimum. Mutation changes randomly the new offspring. For each gene sequence we randomly select 2 genes and switch them.
- End
  [19] After some number of iterations through **Packing** -> **Evaluation** -> **Selection** -> **Crossover** return the best solution in last population.
- Parameters

1. **Population size** defines number of chromosomes in population.
2. **Elitism size** number of the best chromosomes (in context of fitness) to be copied to new population.
3. **Crossover probability** says how often will be crossover performed. If there is no crossover, offspring is exact copy of parents. If there is a crossover, offspring is made from parts of parents' chromosome.
4. **Mutation probability** says how often will be parts of chromosome mutated.

VII. Result and Discussion

```
library(BoxPacking)

# create containers
containers <- list()
n_containers <- 4

for (i in 1:n_containers) {
  containers <- c(containers,
                  Container(length = 2, height = 2, width = 2)
                  )
}


# create boxes
boxes <- list()
n_boxes <- 20

for (i in 1:n_boxes) {
  length <- sample(c(0.4, 0.5, 1), 1)
  height <- sample(c(0.4, 0.5, 1), 1)
  width <- sample(c(0.4, 0.5, 1), 1)

  boxes <- c(boxes,
             Box(length = length, height = height, width = width)
             )
}

# Box Packing
solution <-
  PerformBoxPacking(containers = containers,
                    boxes = boxes,
                    n_iter = 4,
                    population_size = 20,
                    elitism_size = 5,
                    crossover_prob = 0.5,
                    mutation_prob = 0.5,
                    verbose = TRUE,
                    plotSolution = TRUE
                    )
```

Results of the following implementation



VIII. Conclusion and Future work

In this work, we have presented an evolutionary approach to solve the problem of containers organization at the port Problem. Our objective is to respect customers' delivery deadlines and to reduce the number of container handles. We proposed a brief literature review on the bin packing problem and some of its variants, especially the container stowage planning problem. Then, we described the mathematical formulation of the problem. After that, we presented our optimization approach which is an evolutionary algorithm based on genetic operators. We also detailed the use genetic algorithm for solutions improving. The experimental results were later presented by showing the influence of the number of containers in a chromosome and the influence of the number of chromosomes per population on the convergence and the simulation time.

This implementation though lacks the dimensionality to work in any given shape which is very important from industry point of view. Hence being able to implement our work in different shapes is the way forward to make our algorithm industry viable.

IX. References

https://www.geeksforgeeks.org/bin-packing-problem-minimize-number-of-used-bins/

https://pdfs.semanticscholar.org/5bef/ca07fc703127cc765576bda1edcfb7cef803.pdf

https://delta1epsilon.github.io/2016/3D-bin-packing-problem-in-R/

https://github.com/delta1epsilon/BoxPacking

http://thejjjunk.ucoz.com/papers/AGeneticAlgorithmForTheBinPackingProblem.pdf

https://www.codeproject.com/Articles/633133/ga-bin-packing