Bin-packing using Evolutionary Algorithms

Mayamin Hamid Raha
Department of Computer Science and Engineering
University of Nevada, Reno
Reno, Nevada
mraha@nevada.unr.edu

***Abstract*—We aim to approach a bin packing problem with an evolutionary algorithm. The idea is to use an Elitist Genetic algorithm (GA) for searching through a space of 6 triangles placed one after the other in clockwise direction at the center of a circle having various orientations that maximizes the area coverage inside the circle. The application field of this bin packing approach is related to placement of assets on a ship in such orientations that maximizes the ship's protected region.This work is an attempt to observe whether GA is able to find the optimum solution for a cases like this. We found that GA works really well achieving maximum area coverage of 100 percent inside the circular region.**

***Index Terms*—Elitist Genetic Algorithm, bin-packing, selection, crossover, mutation**

I. Introduction

Large sheets of metal, wood, glass needs to be cut in various industries which gave rise to packing and cutting problems. The solutions to encounter these problems comprises of objectives related to minimizing resource wastage through minimization of the number of cuts and maximization of the area utilization. Packing problems in general are optimization problems that tries to find good arrangement of multiple items in larger containing regions. Cutting and packing problems known as cutting stock problems (CSP) have been existing since 1960 [4], [5].These problems are mostly concerned with packing rectangles of different sizes inside a big rectangle [3], [7]. Polygon packing problems have also gained attention of researches and many works have been done on packing polygons inside rectangles. [9].

Inspired by all these works, we decided to define our problem as a Bin-packing problem and used evolutionary algorithm due to their efficient search methodology. In our work, we are considering the use of genetic algorithm that finds the optimum arrangement of triangles representing the area covered by assets of a ship inside a fixed circular region. Here, the circular region's radius is 256 units from the center of a circle which we are considering the center of ship. We attack this circular region packing by triangle problem with a CHC selection [13] based genetic algorithm to observe which combinations of height and angles of triangles cover most of the area inside a circle where the maximum value range for height and angle between two adjacent sides of triangle is from 1 to 250 units and 128 units respectively.

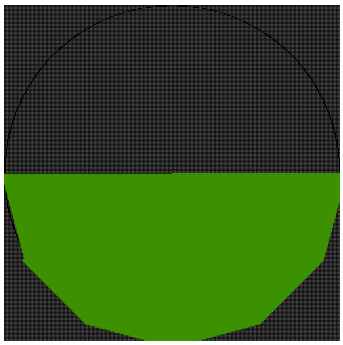


Fig. 1: height1-height6 = [128,128,128,128,128,128] and theta1-theta6 =[30,30,30,30,30,30]

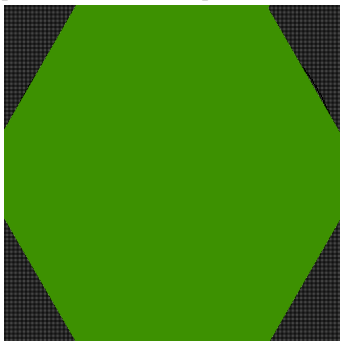


Fig. 2: height1-height6 = [128,128,128,128,128,128] and theta1-theta6 =[60,60,60,60,60,60]

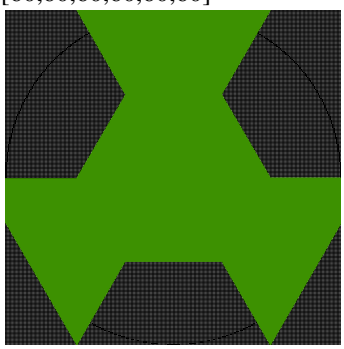


Fig. 3: height1-height6 = [128,64,128,64,128,64] and theta1-theta6 =[60,60,60,60,60,60]

II. Related Work

One of the earliest bin-packing problems that existed is orthogonal bin-packing problem (OPP). In a given finite number of rectangles and a rectangular board, OPP requires a disjunctive placing of rectangles on the board such that the edges are parallel to x and y axis [2].Later on, polygon packing problems evolved and [15] tackled this by placing polygons directly on the board and they used an algorithm that does local optimization by shifting and rotating the polygons ,whereas, [1] approached the problem by placing cluster of 3 polygons on rectangular board. In [9], Jakob et al. used two methods for solving polygon packing problem by applying genetic algorithm directly to polygons and by only applying genetic algorithm [12] to rectangle in which the polygons are embedded followed by use of a deterministic algorithm.

Another branch of cutting stock problems (CSP) is called Open Dimensional Problem (ODP) which consists of determining the best arrangement of a set of rectangular items onto a rectangular strip of material for a fixed height and width where overlapping is not allowed.

Moreover, packing problems can be multi-objective too. In [14],the authors use multi-objective evolutionary algorithms(MOEAs) for optimization of multi-objective formulation of ODP. MOEAs have been used further for 2D ODP [8]. Furthermore recently 3D bin packing problems came into being, where the only difference from Bin-packing is that now the orthogonal packing of given set of rectangular object is inside three dimensional rectangular bins. [11].

Some of the recent advancements in approaching bin-packing problem with genetic algorithms includes [6] where the Score Constrained Packing Problem (SCPP) that involves packing items into a minimal number of bins such that the order and orientation of each items within each bin satisfies a sum constraint has been addressed with an evolutionary algorithm that has option for 3 kinds of recombination. Another recent study on solving irregular bin packing problem was attempted with genetic algorithm by Luo et al. in [10]. This type of problems have a special boundary constraint and the objective is to fir irregular shaped pieces with free rotations to fill up a large irregular stock sheet where goal is to maximize the number of filled pieces.

Motivated by all these approaches we have developed a bin packing approach involving packing of triangles with some special height and angle constraints with in a specific boundary region that comprises of a circular area having a fixed radius.

III. Parameters used

A. Chromosome

- For our genetic algorithm we are using a chromosome of length 90 where there are 6 height values (8 bits each) and 6 corresponding angle values(7 bits each).
- Our chromosome structure is visualized in Figure 4 where h represents height value and a represent corresponding angle values of triangle 1 to 6. The height and angle values are binary comprising of 8 and 7 bits respectively.

Fig. 4: Chromosome structure

- These values were chosen considering the fact that height values can be from 1 to 256 and angle values can be from 1 to 128. In our case all triangles having height near to 256 and angle 60 is considered to be the optimum.

B. Other parameters used

- Through experimentation we found that probability of mutation of 0.05 and probability of cross over of 0.9 works best for our GA.
- We have used CHC based selection with Lamda value of 2.
- We have used 30 different random seeds having values (1-30). We averaged the minimum, maximum and average fitness of our GA in every generation over these 30 random seeds and plotted them in figures that we will see later on in the report.

IV. Fitness calculation

A. Orientation calculation for each triangle

For fitness function calculation, we are considering the calculation of percentage of area covered inside a circle. For this we are using the Pygame library to create a grey color display of size 512 x 512 since 256 x 2 = 512 . After that, we draw a 256 x 256 grid there. Then we draw a circle having radius of 256 from the center of the display. Finally, we start placing triangles from a starting location of positive x axis in clockwise direction. We are considering all the triangles to be isosceles triangle since in case of ships the assets field of view is equally distributed from axis of rotation.The first triangle always starts from positive x-axis and the following triangle's heights have an angle alpha with respect to positive x axis and this angle alpha is not in the chromosome.The angles between 2 similar sides of each isosceles triangles are is given in chromosome in an consecutive order of placement and its name is theta.

Algorithm 1: Calculating alpha angles of triangles (except the first one having theta = alpha and i =0 value)

Result: Alpha angles of each triangle with respect to positive x axis ($Anglevalueindegrees$)
Input: List of angles between two adjacent sides of each isosceles triangles in the order of their placement, $theta\_angles$
foreach $theta\_angles$ *in* $theta\_angles$ *having* $i > 0$ *(where i is the order(index) of triangle in chromosome)* **do**
| alpha(i)= alpha(i-1) + theta(i)
end

B. Drawing the triangles

For drawing the triangles using Pygame, we are required to have location of 2 corner points of each triangle sides. 2 adjacent sides of isosceles triangle has one point in common which is the center of the triangle. We calculated the co-ordinate of end points of each of 3 sides of triangles by solving 3 trigonometric equations for each side.

Algorithm 2: Calculating the co-ordinates of end points of each side of triangles(starting point adjacent sides is the origin)

Result: Co-ordinates $(x, yco - ordinatevalues)$
Input: List of heights and angles between two adjacent sides of each isosceles triangles in the order of their placement, $theta\_angles$

```
a1 = 90
a2 = 0
a3 = 90 + theta(0)
foreach theta_angles in theta_angles (where
 theta(i) > 0 do
    xcos(theta(i)+theta(i-1)) + ysin(theta(i)+theta(i-1))
     = 0
    xcos(theta(i/2)+theta(i-1)) +
     ysin(theta(i/2)+theta(i-1)) = height(i)
    xcos(90+theta(i)+theta(i-1)) +
     ysin((90+theta(i)+theta(i-1)) = 0
    a1 = theta(i) + a1
    a2 = theta(i) + a2
    a3 = theta(i) + a3
end
```

Once the cor-ordinates of each sides end point is found we draw green color filled triangles using pygame's gfxdraw library's drawFilledTrigon() function.

C. Area covered calculation

For area coverage calculation we simply calculate the number of green pixels inside the boundary of the circle and multiply that with pixel size.

$$TotalArea = 3.146 * (radius)^2 \tag{1}$$

We calculate the number of red colored pixels inside the circle and store in a variable named trianglePixelCount. For calculation of area covered inside circle by triangles, we use the following equation:

$$AreaCovered = pixelSize * trianglePixelCount \tag{2}$$

Finally for calculation of percentage of area covered, we use equation 3. We can see how our display looks with various height and theta values in the Figure 1 - 3

$$Fitness = (AreaCovered/TotalArea) * 100 \tag{3}$$

It should be noted that the values that we have used in the code for area calculation is subject to change by a factor of horizontal or vertical grid size since we have done all the calculations considering a 256x256 grid over our display which is of size 512x512. Therefore, our horizontal or vertical grid size in this case is 2.

V. Experiment

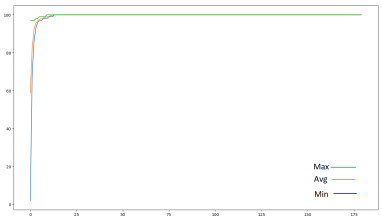
A. On smaller population

We have conducted our experiment in a Dell computer having 64 bit Windows 10 Pro operating system with Intel(R) Core(TM) i9-990KF CPU @ 3.60GHz 3,60 GHz.In this section we observe the performance of GA for various set of parameters :

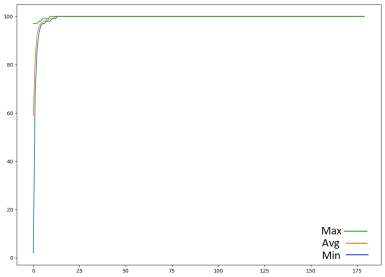
- At first, we begin our experiments with population size of 10 and with 2 different random seeds and run it for 180 generations. We observe in Figure 5 (a) that even with a small population the GA performed really well. These plots were taken by averaging the minimum, average and maximum fitness for each generation over each random seed value. The minimum fitness value starts from a value little more than zero and reaches 100 with in 20 generations, whereas, the average fitness starts from somewhere around 58 and reaches 100 with in 20 generations. The maximum fitness curve starts from 96 percent and reaches 100 within 20 generations.
- Less number of random seeds may contribute to the fact that the GA might be performing very well due to the initial set of chromosomes it started from. For this reason, this time we experimented with 30 different random seeds for population size of 10 and generation size of 180. By seeing the results form Figure5 (b), we see that there is not much difference from Figure5 (a), but we can assure that the GA result is generalized better and there is not scope of GA performing well due to initial starting population.
- We wanted to see if running the GA for more number of generations gives better results. From the results in Figure5 (c) and Figure5 (d) we see that the average and minimum fitness plot is similar .i.e. reaching 100 percent area coverage within 15 epochs approximately where 30 different random seeds have been used. Here, we see a slight improvement in the maximum fitness value in case of experiment with 540 number of generation, compared to the results of GA run for 270 generation. The performance maximum fitness improvement is almost negligible and here also both minimum and average fitness start from near to zero and 58 respectively after which the values keep on increasing from there to 100 within 20 generations.

B. On larger population

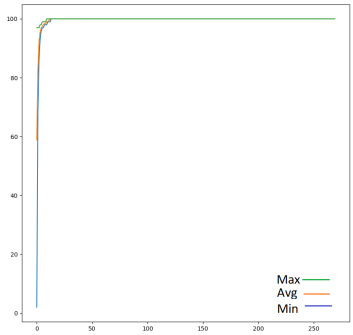
Through experimentation, we found that a larger population size with generation size = 1.5 * population size works well. We have experimented with population sizes of 100, 200 and 300 with generation number 150, 300 and 450 respectively. We explain the results obtained from these experiments in result section.



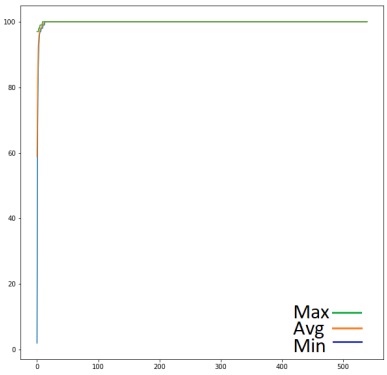
(a) Fitness Vs Generation plot for population size = 10, random seeds = 2 and generation size = 180



(b) Fitness Vs Generation plot for population size = 10, random seeds = 30 and generation size = 180

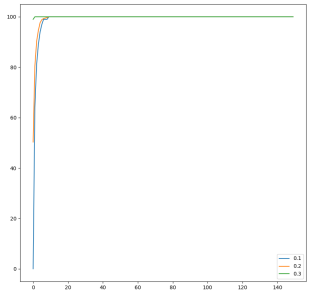


(c) Fitness Vs Generation plot for population size = 10, random seeds = 30 and generation size = 270

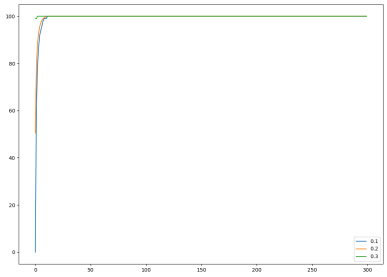


(d) Fitness Vs Generation plot for population size = 10, random seeds = 30 and generation size = 540

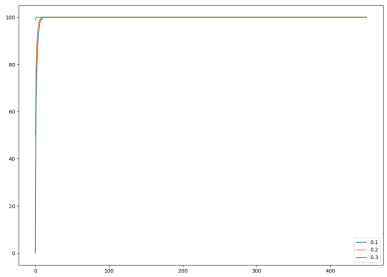
Fig. 5: Experimental results for small population size of 10



(a) Fitness Vs Generation plot for population size = 100, random seeds = 30 and generation size = 150



(b) Fitness Vs Generation plot for population size = 200, random seeds = 30 and generation size = 300



(c) Fitness Vs Generation plot for population size = 300, random seeds = 30 and generation size = 450

Fig. 6: Experimental results for population size 100, 200, and 300 with generation number 150, 300 and 450 respectively

VI. Results

A. Plot description

- We observed a noticeable difference in Fitness Vs Generation plot when we used a larger population size of 100, 200 and 300 and a generation size equal to population size * 1.5 . In Figure 6 (a), we observe that this time average fitness value has started from a lower value than the ones in experiment section and that value is 48, moreover minimum and average fitness plot reaches a

value of 100 with in 10 epochs which is much faster convergence compared to the results of experiments with population size 10.

- In Figure 6 (b), we used a population size of 200 and we ran the GA for 300 generations, we notice that the results are very similar to results in Figure 6 (a). Here, we conclude that increasing size of population beyond 100 did not result in any improvement in terms of fitness.
- Finally, we use a very large population size of 300 and generation number 450 and it is evident from Figure6 (c) that the fitness curves don't get any better than Figure 6 (a) and Figure 6 (b)

B. Discussion

Since we are working on a problem whose optimum is known to us due to its simplicity that is why GA performs very well and converges on an average with in 20 generations. However, here our chromosome size, x = 90 and 2^x is a very large number, Our chromosome can have that many number of combinations which makes it an NP hard problem which is why we considered applying GA on it. Since GA converges so quickly for this problem, we can state that it has a lot of potential for searching through even more complex space and more complex version of this circular bin packing problem.

VII. Conclusion

In conclusion we can say that evolutionary algorithms work very well on NP hard bin-packing problems. Sequential processing was done due to which the experiments with larger populations (100 -300 population size with generation number 150-450) took weeks to train,whereas, all 4 experiments with population size 10 could be completed within 1 day only. In future, we hope to apply parallel processing to make the GA implementation faster. From our observation of all experiments and final results we can say that for this particular problem comparing the time and computational cost trade off, the experiments with larger population set is suitable for problem sets where even a slightest change matters a lot. In our work we used CHC selection and we were able to find optima within 20 generations in an average for 30 different random seeds and various size of population and generation number. Since the application of this packing problem is related to finding height and orientations of assets that maximize the protected region around a ship, it would be interesting to further use GA for placing assets in various locations of ships instead of only placing them in center. In that case, it might be a very effective tool for solving real life problems related to asset placement on ships.

References

[1] Michael Adamowicz and Antonio Albano. Nesting two-dimensional shapes in rectangular modules. *Computer-Aided Design*, 8(1):27–33, 1976.

[2] Brenda S Baker, Edward G Coffman, Jr, and Ronald L Rivest. Orthogonal packings in two dimensions. *SIAM Journal on computing*, 9(4):846–855, 1980.

[3] Dayanne G Coelho, Elizabeth F Wanner, Sergio R Souza, Eduardo G Carrano, and Robin C Purshouse. A multiobjective evolutionary algorithm for the 2d guillotine strip packing problem. In *2012 IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2012.

[4] Paul C Gilmore and Ralph E Gomory. A linear programming approach to the cutting-stock problem. *Operations research*, 9(6):849–859, 1961.

[5] Paul C Gilmore and Ralph E Gomory. A linear programming approach to the cutting stock problem—part ii. *Operations research*, 11(6):863–888, 1963.

[6] Asyl L Hawa, Rhyd Lewis, and Jonathan M Thompson. Exact and approximate methods for the score-constrained packing problem. *European Journal of Operational Research*, 2022.

[7] EBCH Hopper and Brian CH Turton. An empirical investigation of meta-heuristic and heuristic algorithms for a 2d packing problem. *European Journal of Operational Research*, 128(1):34–57, 2001.

[8] Simon Illich, Lyndon While, and Luigi Barone. Multi-objective strip packing using an evolutionary algorithm. In *2007 IEEE Congress on Evolutionary Computation*, pages 4207–4214. IEEE, 2007.

[9] Stefan Jakobs. On genetic algorithms for the packing of polygons. *European journal of operational research*, 88(1):165–181, 1996.

[10] Qiang Luo, Yunqing Rao, and Deng Peng. Ga and gwo algorithm for the special bin packing problem encountered in field of aircraft arrangement. *Applied Soft Computing*, 114:108060, 2022.

[11] Silvano Martello, David Pisinger, and Daniele Vigo. The three-dimensional bin packing problem. *Operations research*, 48(2):256–267, 2000.

[12] Jeffrey R Sampson. Adaptation in natural and artificial systems (john h. holland), 1976.

[13] Anabela Simoes and Ernesto Costa. Chc-based algorithms for the dynamic traveling salesman problem. In *European Conference on the Applications of Evolutionary Computation*, pages 354–363. Springer, 2011.

[14] Xiang Song, CB Chu, YY Nie, and Julia A Bennell. An iterative sequential heuristic procedure to a real-life 1.5-dimensional cutting stock problem. *European Journal of Operational Research*, 175(3):1870–1889, 2006.

[15] Johannes Terno, Rudolf Lindemann, and Guntram Scheithauer. *Zuschnittprobleme und ihre praktische Lösung*. VEB Fachbuchverlag, 1987.