International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 09 Issue: 07 | July 2022 www.irjet.net p-ISSN: 2395-0072

BIN PACKING ALGORITHM IN GOLANG

Pramod Patil G S[1], Mr. Ravishankar Holla[1]*

[1,2]*Department of ECE, R V College of Engineering, Bangalore*

Abstract - The 21st century is an era of digital revolution. There have been advancements in the technology over the two decades in various domains like the Information Technology (IT) sector, Education, Finances, Commerce and so on. There is a shift from mechanical and electronic analogue technology to digital technology. Commerce sector is now widely popular as E-Commerce. In the first step, thorough study Golang, Microservices and docker will done to understand the concept of Design process. After the study the necessary software tools, programming languages and libraries studied along with construction methodology, algorithm will be designed. Designed Algorithm implemented in the Golang, and this algorithm is converted into Gorilla/ Mux API. API converted into independent microservices and the microservices converted into docker image. Docker image is adopted to testing software by QA team and check for test cases and then this is adopted to Seller Portal.

In This, System achieves following inferences, Successful testing with Single SKU, Successful testing with Multiple SKU, passed all the Unit test cases in Unit testing, In SonarQube testing 12.6 percentage of duplication over 2.4k lines found, 5. 2 duplicate blocks found, 6. 10 code smells found, 7. 0 bugs, vulnerabilities and security hotspots found.

Key Words: Digital revolution, E-commerce, Bin packing

1. Introduction

E commerce packaging often refers to packaging that is used to ship the products directly to our customers. With the e-commerce industry growing due to covid-19, shipping boxes are now becoming an important part of the unboxing experience for most consumers. The main thing to consider when choosing e-commerce packaging is how well it will protect the product. The organization's seller portal displays packaging, labeling, and shipping information. So, our goal is to automate packaging selections in this portal against orders based on volumetric weights of products in order to reduce shipping costs. [1] High-quality packaging is viewed as an additional investment by the majority of ecommerce businesses. What they don't realize is that proper packaging reduces the company's costs significantly. Good packaging protects the product during shipping, reducing the likelihood of customers returning it.[4]

1.1 Gota: Dataframes and Series

The process of analysing, modifying, and processing raw data and datasets to gain insights from them is known as data analysis. Python and R are the most often used data analysis languages. However, Go is becoming increasingly popular for this reason.[3] Gota is a Go programming language dataframe and data wrangling package. Gota is similar to the Python Pandas library and is designed to interact with Gonum, a scientific computing package in Golang, akin to Pandas and Numpy. The Gota module simplifies data wrangling (transformation and manipulation) operations in Go.[5] It supports Go's built-in data types as well as a variety of file formats such as JSON, CSV, and HTML.

1.2 Filter Values in Gota

We utilize to filter values. Apply a filter to the dataframe object. This accepts dataframe.F, to which we provide a struct literal. The struct literal accepts a column name Colname, a comparator Comparator, and a value Comparando, which is the value for which the dataframe should be filtered.

- series.Eq → Equal to.
- series.Neq → Not Equal to.
- series.Greater → Greater than .
- series.GreaterEq → Greater than or Equal to.
- series.Less → Less than.
- series.LessEq → Less than or Equal to.
- series.In (http://series.In) → Is contained In.

The Select method assists us in selecting columns from a dataframe. [df.Select] (http://df.Select) accepts a slice of two integers that represent the number of columns that can be selected. We can choose rows by utilizing the dataframe object's Subset function. dataFrame. Subset accepts a slice of two integers representing the number of rows that can be chosen. Gota has a variety of dataframe manipulation features.[4]

Display packed function displays packed items in the bin.

We will analyze technique to characterizing possible packing and building optimal solutions in this research. The challenge of providing a relevant issue formulation adds to the difficulty of finding optimal solutions for the Three-Dimensional Bin Packing Problem. In order to formulate the problem, we will assume that each item I in the finite set S has three dimensions wi, hi, and di. Each identical bin b has W, H, and D dimensions. The items and bins are rectangular boxes with three dimensions that match to the values for width, height, and depth.[7]

The pieces are permitted to rotate orthogonally to distinguish the answer from earlier efforts. To rotate an item, simply swap its width (wi), height (hi), and depth (di) values around in an ordered fashion. Each item-box has six rectangular facets, but only three are different because "opposite" facets are identical.[3]

Each of the three facets can be rotated orthogonally to create a new box form. As a result, each item can have up to six alternative rotation configurations.

For single-dimensional problems, linear programming algorithms have been used. Evolutionary algorithms have been utilized in some ways instead of heuristic algorithms [1,3]. Evolutionary algorithms could have been a good strategy for discovering a solution (in this case, the solution space of the Bin packing issue) due to their ability to search enormous spaces, but evolutionary methods have a few shortcomings: There is little continuity between solution and problem, which means that if you modify the problem parameters little, the solution changes significantly.[6] Heuristics are more detailed, and some provide worst-case results.

2. Mathematical formulation for bin packing

Bin packing being an NP-Hard issue implies that an exhaustive search for the optimal solution is computationally intractable in general, and that there is thus no known real computationally feasible optimal solution approach for the problem. As a result, different methods of obtaining a solution must be discovered. Heuristic solution methods are the most prevalent. Items are packed one at a time, with no backtracking (once packed, an item is not repacked). Formal reasoning derived from one of the following packing algorithms can be used to select an item to be packed.[8]

2.1 First Fit

Unassigned item is placed in the first bin that has enough space. If no such bin exists, allocate the item to a new bin.

2.2 First Fit Decreasing.

Almost identical to First Fit, except that the items are sorted in decreasing order first before being packed.

2.3 Last Fit

Packs unassigned item into last available bin. Searching is identical to First Fit, however the bins are ordered in reverse. If no such bin exists, move the item to a new bin.[7]

2.4 Best Fit

The Best Fit algorithm places an item in the bin that has the most space among those that fit the item. To be more specific. Items are packed one at a time in the order specified. Determine set B of containers into which the item fits before determining the bin. If B is empty, create a new bin and place the item in it. Otherwise, place the item in the B bin with the least available capacity.[7]

3. Bin Pack Problem Solution Specifications

To conduct the core of the bin packing, the developed system employs a heuristic technique.

The use of these heuristic approximate algorithms in the system to solve the bin packing problem:

i. guarantees a solution,

ii. obtains a solution in a reasonable time (i.e. solution is computationally feasible to obtain),

iii. allows for general data input, and

iv. provides continuity between the solution and the problem.

It goes without saying that failing to satisfy I and (ii) would be unsatisfactory. If the system does not meet (iii), it loses its generality and flexibility. Condition (iii) is especially important since bins (or containers) must be packed with things (or cargo) of varying sizes.

Failure to fulfill (iv) would also make it impossible for users to retrieve data and test alternative solutions during the problem-solving process. As a result, failing to satisfy (iv) indicates that the system does not readily support this functionality.[6]

The system employed two basic heuristic bin packing algorithms: the First Fit Decreasing and the Best Fit. They were chosen above other heuristic algorithms because they run faster and produce solutions that are considerably closer to ideal than most other heuristic algorithms.

3.1 Specifications of the software

We have three packing directions: width direction, height direction, and depth way. As previously stated, each Item has six different rotation kinds. Consider the following item: Rotating about the x, y, and/or z axes yields the six rotation types. The bins are packed one at a time, and the algorithms pack the object using a sequence of pivot points.

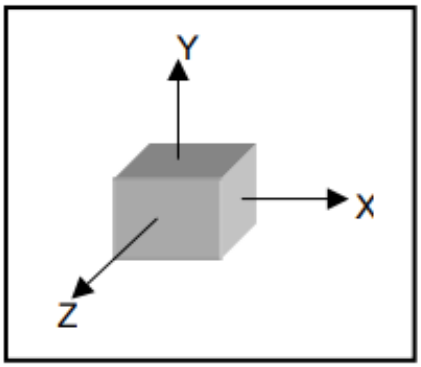

Fig 1. 3D coordinates

As a result, it looks that Best Fit is more likely to yield a solution that is closer to the ideal solution than First Fit Decreasing, but because these values on the upper bound for the number of bins required are quite close, it is worthwhile to use both algorithms.[6]

Table 1: Box rotation options

Rotation Type	First axis to rotate about	Second axis to rotate about
0	-	-
1	Z	-
2	Y	-
3	X	Y
4	X	-
5	X	Z

3.2 Methodology

In the first step, thorough study Golang, Microservices and docker will done to understand the concept of Design process. After the study the necessary software tools, programming languages and libraries studied along with construction methodology, algorithm will be designed. Designed Algorithm implemented in the Golang, and this algorithm is converted into Gorilla/ Mux API. API converted into independent microservices and the microservices converted into docker image. Docker image is adopted to testing software by QA team and check for test cases and then this is adopted to Seller portal.

We use Bin packing algorithm which means The bin packing issue is an optimization problem in which things of varying sizes must be packed into a finite number of bins or containers, each with a defined capacity, in such a way that the number of bins needed is minimized.

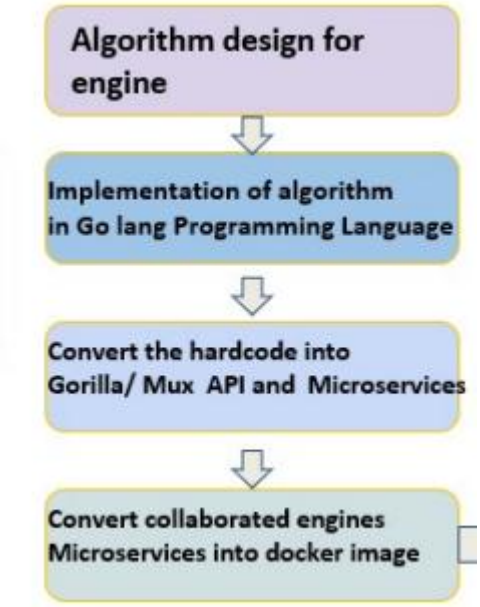

Fig 2. Methodology

Next step….,

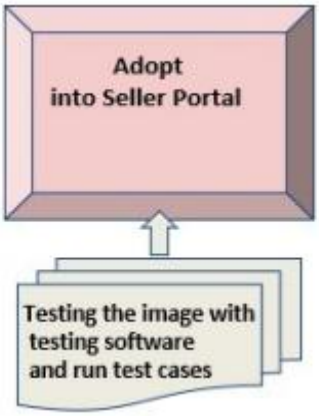

Fig 3. Testing

The problem has numerous applications, including container filling, truck loading with weight capacity limits, media file backups, and technology mapping in FPGA semiconductor chip design.

3.3 Best Fit Algorithm for bin packing

Choose a packing direction. Each bin has three packing directions: a width (or x) direction, a height (or y)

direction, and a depth (or z) direction. Pack each container one at a time.[7]

First, we select a pivot point. The pivot is a (x, y, z) coordinate that represents a place in a specific 3D bin where an attempt will be made to pack an item. The pivot will be located at the item's back bottom left corner. If the object cannot be packed at the pivot point, it is rotated until it can or until all six potential rotation types have been tried. If the item still cannot be packed at the pivot point after rotating it, we move on to packing another item and add the unpacked item to a list of objects that will be packed when the other items are packed.[8]

3.4 First Fit Decreasing Algorithm for bin packing

To pack an item, one must first choose a packing direction. The bin's longest side corresponds to the packing strategy Then, rotate each item so that The longest side of this item is the packing side. [8] If we are packing by width, for example, we want the The item's breadth is determined by its longest side, so for instance, if the packing direction is breadth and the If the item's current height is more than its width, then turn the item.

If, following the rotation(s), the item cannot fit into the bin (i.e. one or more of the items' dimensions exceed the bin's corresponding dimension), we rotate the item till the second longest side of this item corresponds to the packing direction. If the item still does not fit into the bin after the rotation(s), we rotate it till the third longest side of the item corresponds to the packing orientation. Then, depending on the packing direction, sort the objects in decreasing order of width, height, or depth.[8]

The Bin Packing Problem and the Cutting Stock Problem are two NP-hard combinatorial optimization problems that are connected. Exact solution methods can only be employed for extremely tiny cases, hence heuristic methods must be used for real-world situations. Researchers have recently begun to apply evolutionary techniques to these challenges, such as Genetic Algorithms and Evolutionary Programming. We employed an ant colony optimization (ACO) strategy to tackle both the Bin Packing and Cutting Stock Problems in this paper. [7] We describe both a pure ACO technique and an ACO approach enhanced with a basic but highly successful local search algorithm.

It is also demonstrated that the hybrid ACO strategy requires different parameter values than the pure ACO approach and provides more robust performance across diverse issues with a single set of parameter values. When the local search algorithm is conducted with random restarts, it performs noticeably lower than when paired with ACO.

4 Results & Discussions

The net/http package in Go provides a wide range of HTTP protocol functionality. Complex request routing, such as segmenting a request url into separate parameters, is something it struggles with. Fortunately, there is a highly popular package for this, which is well known in the Go community for its high code quality. The gorilla/mux package is used in this example to define routes with named arguments, GET/POST handlers, and domain limitations. gorilla/mux is a package that modifies the default HTTP router in Go. It includes a slew of features designed to boost productivity when developing web apps. It also conforms to Go's default request handler signature function (w http.ResponseWriter, r *http.Request), allowing the package to be mixed and matched with other HTTP libraries such as middleware or existing apps.

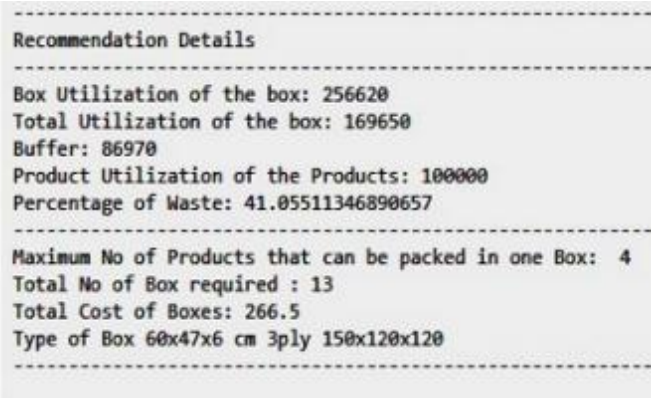

Fig 4. Result of Single SKU

4.1 Unit Testing

Unit testing is a software testing technique used in computer programming to check the suitability of individual pieces of source code, sets of one or more computer programme modules, and the associated control data, usage processes, and operating procedures.[5]

Each unit or individual component of the software application is tested as part of the unit testing process. It represents the initial stage of functional testing. The purpose of unit testing is to confirm the functionality of individual unit components. A unit is a single testable component that may be tested as part of the application software development process. Unit testing is used to ensure that isolated code is correct. An specific application function or piece of code is referred to as a unit component. Unit testing is typically conducted using the white box testing methodology by developers.[5]

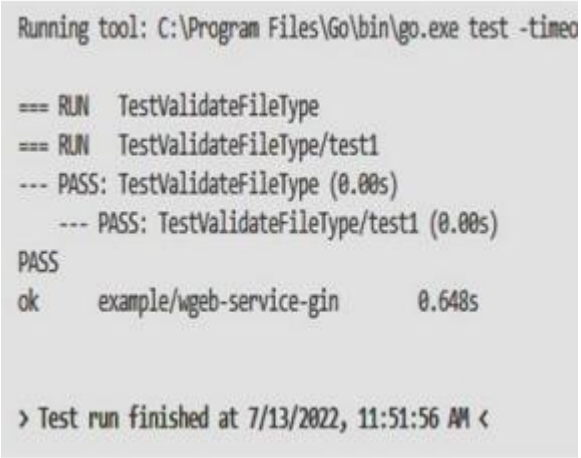

Fig 5. Results of Unit Testing

5. CONCLUSIONS

Bin packing is a tremendously intriguing mathematical model issue, yet research on it is surprisingly new.

In this work, we looked at how to implement the optimization of packing 3-D boxes into a finite number of bins and shown that the program will find a solution in a fair amount of time. Most alternative implementations suffer from the major disadvantage of failing to converge to a solution and hence running "indefinitely." Our meticulous design also includes the visualisation of the solution, which means that the exact placement and orientation of an item in a bin is known.[9]

The GGA is a genetic algorithm that has been substantially modified to fit the structure of grouping issues. These are the problems in which the goal is to discover a good partition of a set or to group the members of the set together. The bin packing problem (BPP) is a well-known NP-hard grouping problem in which items of varying sizes must be put into fixed-capacity bins. On the other hand, based on their dominance criterion, Martello and Toth's reduction method is one of the best OR strategies for BPP optimization to date.[8]

The Bin Packing Problem (BPP) is defined as follows ([Garey and Johnson, 79]): given a finite set O of numbers (item sizes) and two constants C (the bin's capacity) and N (the number of bins), is it possible to 'pack'all the items into N bins, i.e. does there exist a partition of O into N or less subsets, such that the sum of elements in any This NP-complete choice problem easily leads to the associated NP-hard 2 optimization problem, which is the focus of this paper: what is the best packing, i.e. what is the smallest number of subsets in the previously specified partition? Because BPP is NP-hard, there is no known optimal algorithm that runs in polynomial time.[9]

[Garey and Johnson, 79], on the other hand, cite simple heuristics that can be proven to be no worse (but also no better) than a relatively tiny multiplication factor above the ideal number of bins. The concept is simple: starting with one empty bin, take the goods one by one, searching the bins so far used for a space large enough to contain them.

REFERENCES

1. Frederick Ducatelle, John Levine. Ant Colony Optimisation for Bin Packing and Cutting Stock Problems, Proceedings of the UK Workshop on Computational Intelligence, 2001, Edinburgh.

2. Emanuel Falkenauer, A hybrid grouping genetic algorithm for bin packing, Journal of Heuristics, 1996

3. Fekete, S. P., J. Schepers, A new exact algorithm for general orthogonal 3d-dimensional knapsack problems, Lecture Notes in Computer Science, 1997

4. Leon Kos, Joze Duhovnik, Rod Cutting Optimization with Store Utilization, International design conference – DESIGN, Dubrovnik, 2000.

5. Silvano Martello, David Pisinger and Daniel Vigo, The three dimensional bin packing problem, Institute for Operations Research and the Management Sciences (INFORMS), Linthicum, Maryland, USA, 2000

6. Peter Ross, Sonia Schulenburg, Hyper-heuristics: learning to combine simple heuristics in bin-packing problem, Discrete Applied mathematics ACM, 2000.

7. Erick Dube, Leon R. Kanavathy, OPTIMIZING THREE-DIMENSIONAL BIN PACKING THROUGH SIMULATION,

8. B. Sampaio A R Rubin J, “Improving microservice-based applications with runtime placement adaptation,” J Internet Serv Appl, vol. 10, no. 3, 2019, issn: 1999-5403. doi: 10.3391/fi1002456.

9. C. G. Rosa N Cavalcanti D, “Adaptive middleware in go - a software architecturebased approach,” Appl. Sci. 2022, vol. 10, no. 2, 2018, issn: 1999-5903. doi: 10.3390/fi10020014.

10. A. Kokkinaki, R. Dekker, M. de Koster, C. Pappis, and W. Verbeke, “E-business models for reverse logistics: Contributions and challenges,” in Proceedings.