Efficient Cargo Allocation: An Algorithmic Approach to Dynamic Bin Packing

Videep Ekbote

Abstract

In logistics and supply chain management, efficiently packing diverse cargo into containers is a critical problem. This project addresses dynamic bin packing using various algorithmic approaches, specifically designed for managing bins with fixed capacities and objects of different sizes and colors. Each cargo requires special handling based on its characteristics, and the system assigns appropriate bins using optimized strategies. This paper explores two main algorithms — Largest Fit and Compact Fit — and demonstrates their applications in efficient cargo allocation. This project uses the most efficient way of solving the bin packing problem in logarithmic time complexity, using AVL trees.

Introduction

The dynamic bin packing problem, which involves assigning objects of varying sizes to bins with limited capacities, is central to numerous logistical challenges. This project focuses on solving such problems using a set of tailored algorithms that optimize the allocation of objects based on both size and specific handling requirements.

The system developed models a real-world cargo management system, where each bin and object is assigned a unique identifier (ID), and objects are assigned to bins based on their color, representing special handling instructions.

Problem Definition

The cargo management system must:

- Efficiently pack objects of varying sizes and colors into bins with limited capacities.
- Assign objects based on specific algorithms for different object types (colors).
- Ensure that each object is placed into a bin with sufficient remaining capacity, based on the selected algorithm.

Each bin has a specific capacity, and objects must be packed using one of the following algorithms:

1. **Largest Fit Algorithm**: Objects are placed in the bin with the largest remaining capacity.
2. **Compact Fit Algorithm**: Objects are placed in the bin with the smallest remaining capacity that can still accommodate the object.

Methodology

To tackle this dynamic bin packing problem, we modeled the system using Python and implemented two primary algorithms, each catering to different cargo types:

- **Largest Fit Algorithm**: Optimized for placing large objects, this algorithm selects the bin with the greatest available space, ensuring that large cargo items are packed efficiently.
- **Compact Fit Algorithm**: Designed for compact packing, this algorithm assigns objects to the smallest bin that can still hold the item, reducing wasted space.

The system models bins and objects as classes, ensuring efficient packing through methods that dynamically adjust bin capacities and object assignments.

Object and Bin Management

The cargo system assigns objects to bins based on the object color:

- **Blue Cargo**: Objects are packed using the Compact Fit algorithm, prioritizing the bin with the smallest remaining capacity and the lowest ID.
- **Yellow Cargo**: Similar to Blue Cargo, but bins with the greatest ID are selected.
- **Red Cargo**: These objects use the Largest Fit algorithm, selecting the bin with the most remaining capacity and the smallest ID.
- **Green Cargo**: Red Cargo with the highest bin ID is selected.

We used a total for $4 + m$ AVL trees along, among which two contained nested AVL trees. Using AVL trees ensured that we could obtain a logarithmic time complexity.

The first AVL tree contains the bin capacities and bin id's and the ordering is non decreasing in the bin capacities and increasing in bin id's, while the second AVL tree contains the bin capacities in non decreasing order and bin id's in decreasing order.

Another two AVL trees were used to store the bin id's sorted in increasing order and store the object id's sorted in increasing order. Each bin was an instance of the Class Bin, and has an attribute of another AVL tree which stored objects in increasing order of capacities.

Computational Complexity

We have used a total of $4 + m$ AVL trees. The first three trees store a total of $n{+}m$ objects each and the fourth tree stores a total of m objects. So, the Space Complexity is *O(n+m)*

For adding and deleting an object from a bit, we first traverse the object tree to find the corresponding bin it belongs to, this takes up *O(logm)* time. Then we search for the appropriate bin in the bin tree, delete the bin, change its capacity and insert it back into both the bin trees. This takes up *O(logn)* time. So, overall time complexity is *O(logn + logm).*

Results

After implementing and testing the algorithms, we observed the following key results:

- The **Largest Fit Algorithm** successfully packed large objects into fewer bins, reducing the overall number of bins required for heavy-duty shipments.
- The **Compact Fit Algorithm** was particularly effective for standard cargo, ensuring minimal wasted space.
- The system dynamically adjusted to various object sizes and bin capacities, maintaining efficient object placement even under constraints.

Conclusion

This project presents an effective solution to the dynamic bin packing problem through the implementation of the Largest Fit and Compact Fit algorithms. By testing these algorithms in a simulated cargo management system, we demonstrated their effectiveness in improving packing efficiency.