July 2020

Efficient Algorithm for Generating Order Packing Recommendations

Jonathan Yue Shun Poon

Guorui Su

Efficient Algorithm for Generating Order Packing Recommendations

ABSTRACT

Packing an order in the most appropriately sized boxes can improve customer experience, reduce costs, and alleviate stress for the operator that packs the items into the boxes. Products to determine optimal packaging typically consider permutations of packing options and select one that likely minimizes total costs while preventing the likelihood of product damage. However, such products use algorithms where the runtime increases exponentially with the addition of a box or an item in the input, thus making them impractical for orders that require multiple boxes and items. Moreover, such products typically do not take merchant and product variations and specialized needs into consideration. This disclosure describes an efficient general-purpose algorithm to generate packing recommendations. The algorithm produces near-optimal packing recommendations within a reasonable time.

KEYWORDS

- Order packing
- 3D bin packing
- Pack recommendation
- Shopping pack
- Mixed Integer Program (MIP)
- MIP solver
- Integer Linear Programming
- Packing optimization
- Sustainability

BACKGROUND

Retailers, such as online and brick-and-mortar stores, often need to ship products to their customers. When a customer order includes multiple products, the items are typically packed in a single box. Since products come in a variety of sizes and shapes, retailers typically use boxes with a multitude of sizes for packing orders. Packing an order in the most appropriately sized box can improve customer experience, reduce costs, and alleviate stress for the operator that packs the items into the box.

Products are available to suggest optimal packaging material for each customer order. Such products typically select the most optimal packaging by examining permutations of the various potential packing options and determining the one that likely minimizes total costs while preventing the likelihood of product damage during transportation. The total costs include the costs of packaging, dunnage, and shipping. Operators that pack customer orders use the recommended packing materials to pack the items in the customer order. High quality packing recommendations are a core component of the order packing workflow that helps ensure consistent parcels at low cost by reducing wasted packaging materials, improving operator efficiency, and providing greater operational oversight.

However, selection of the most suitable packaging for each order is challenging for a number of reasons such as lack of an efficient optimization algorithm, lack of a scalable solution to obtain data on relevant properties of the items, etc. For instance, current approaches to the computational aspects of 3D bin packing typically ignore practical aspects such as running time of the algorithm, characteristics of the boxes used in packing centers, etc.

Typical optimization algorithms used in packaging recommendation products employ a Mixed Integer Program (MIP) solver that considers box and item dimension. For small orders,

such as those involving only a handful of items packed in a single box, the process can produce high quality packing recommendations. However, the algorithm can require a long running time as the runtime increases exponentially with the addition of a box or an item in the input. As a result, the algorithm is impractical for orders that require multiple boxes with numerous items especially when fulfillment centers are required to pack hundreds of orders per hour.

When the MIP-based algorithm cannot produce a solution quickly, the operation falls back to an algorithm that uses a random approach to provide a recommendation. The random approach employs a stochastic algorithm that tries a large number (potentially, hundreds of thousands) of random packing arrangements and chooses the cheapest among these. While the approach always provides a valid packing recommendation, it is often suboptimal compared to the packing arrangements that operators can employ on their own. Therefore, operators frequently override the recommendations provided by the random approach and lose confidence in the recommendations as a consequence. Moreover, the random approach can also take a significant amount of time to evaluate the random arrangements.

Orders that need to fall back to the random approach for packing recommendations are typically larger in size, thus making up a large portion of the overall packaging and shipping costs. Moreover, even with the random approach, orders with larger items experience high latency, requiring longer to generate a packing recommendation than the duration between finalization of an order and packing of the items. As a result, generation of packing recommendations for orders with larger items may be delayed until after the orders have already been packed, thus making such recommendations moot.

Further, different merchants can vary in their packaging needs. For instance, wholesale merchants that sell larger items require different order packing and shipping strategies compared

to those that sell smaller items that can be shipped in a plastic mailer. Similarly, some types of items have specialized packing needs. For example, to avoid breakage, fragile items may first need to be packed in their own boxes before being placed in a larger box for the whole order. Current products typically do not take such variations and specialized needs into consideration when generating packing recommendations.

DESCRIPTION

This disclosure describes an efficient algorithm to generate packing recommendations for fulfilling orders by packing the ordered products into boxes. The algorithm is a combination of the MIP and naïve approaches that produces near-optimal packing recommendations within a reasonable time. The algorithm involves breaking down the larger problem into smaller parts and employing the MIP technique for the smaller parts. Such an approach can significantly reduce running time to generate packing recommendations.

Inputs to the algorithm include the relevant properties of the available boxes (type, length, width, height, weight, etc.) as well as those of the items in the order (length, width, height, weight, quantity, fragility, liquidity, toxicity, etc.). Based on these inputs, the algorithm generates an optimal packing arrangement for the given order such that the total cost is minimized.

A core assumption of the algorithm is that the shipping cost increases with increase in the number of boxes required for packing the order. Using fewer boxes not only reduces shipping and dunnage costs but also saves labor costs because an operator can pack the order more quickly when fewer boxes are involved. Further, such a heuristic reduces complexity by avoiding the need to keep track of differences between carriers and non-linear pricing structures. The reduced complexity results in faster operation by decreasing the needed calculations.

The algorithm assumes that an MIP algorithm can determine whether a given list of items can fit within a single specified box such that MIP(items, box) returns TRUE or FALSE. Since the operation involves only a single box, it can run quickly since the number of inputs can increase only in a single dimension. The algorithm employs the following steps:

1. Identify a box with the largest volume BMax from the given list of boxes.
2. Select the maximum number of items where the total volume of the items is the percentage to which the volume BMax can be filled, starting with 100%.
3. Verify that the total volume of all selected items can fit in an available box.
    - 3.1. Start with the smallest box that has a volume equal to or greater than the total volume of the selected items via MIP(selected items, smallest box).
    - 3.2. If the items do not fit, continue with the next smallest box via MIP(selected items, box).
    - 3.3. If the selected items do not fit any available box, go back to step 2 to select a smaller set of items by progressively decreasing the fill percentage.
    - 3.4. If the items fit, remove the packed items from the initial list of items, and go back to step 2 to select a new set of items.
4. Repeat until all items in the order are packed.

```
PackItems(items, sortedBoxes, largestBox):
  while (items is not empty):
    for (fillPercentage = 1; fillPercentage > 0; fillPercentage -= 0.05):
      itemsToPack = selectItems(items, fillPercentage)
      boxUsed = packIntoSingleBox(itemsToPack, sortedBoxes)

      if (boxUsed != null)
        items.remove(itemsToPack)
        solution.add(boxUsed, itemsToPack)
  return solution
```

Fig. 1: Example code snippet showing the operation of the packing algorithm

Fig. 1 shows an example code snippet that illustrates the operation of the algorithm described in this disclosure. The **`selectItems`** method is used for optimistic selection of the maximum number of items that can be packed into a single box such that the total volume of the selected items is less than or equal to the largest box volume multiplied by the fill percentage. The parameter **`largestBox`** can be provided to obtain the volume of the largest box. The **`packIntoSingleBox`** method is employed to verify that it is indeed possible to pack the selected items into one of the provided boxes, starting with the smallest box with equal to or greater volume than the total volume of **`ItemsToPack`**. If yes, **`packIntoSingleBox`** returns the box used to pack the item, and the items are removed from the queue. If not, the fill percentage is reduced to select a smaller subset of items, making it more likely that the next call to **`packIntoSinglebox`** will succeed. The process continues until the item queue is empty.

By using total volume as the criterion for rapidly selecting a subset of items, the algorithm can break down large orders into smaller chunks that can be verified quickly by the underlying optimization technique. The runtime of the operation increases linearly with the number of boxes rather than exponentially as would be the case without the initial rapid subsetting. Further, by approximating each chunk using the volume of the selected items and boxes, the algorithm maximizes the likelihood of fitting the selected items within the specified boxes, thus reducing the number of times the MIP approach is invoked in order to yield a solution. Such an operation allows similar handling of order regardless of their size, making it flexible for dealing with larger and more difficult-to-pack orders without modifications.

The algorithm can be expanded with the inclusion of additional features such as handling items that need not be packed in a box, identifying situations where an item is best shipped in a box of its own, etc. Further, the algorithm operation can be repurposed to address the reverse

problem of identifying an optimal set of boxes given a list of orders, thus helping merchants determine the best box sizes for their fulfillment operations. The algorithm can be incorporated into existing cloud-based and/or offline solutions for package recommendations for any merchant. In addition, the functionality of the algorithm can be made accessible via an application programming interface (API), thus permitting its use by third party applications.

Implementation of the algorithm can result in more efficient and effective order packing which reduces package, dunnage, and shipping costs, minimizes damage during shipping, and increases customer satisfaction. Further, reduction in the use and shipping of packing material can contribute to sustainability initiatives. Moreover, the algorithm can help realize savings by increasing fulfillment throughout by reducing packing errors and speeding up the packing pipeline.

CONCLUSION

This disclosure describes an efficient general-purpose algorithm to generate packing recommendations. The algorithm produces near-optimal packing recommendations within a reasonable time. A core assumption is that the shipping cost increases with increase in the number of boxes required for packing the order. Based on this assumption, the algorithm can break down the larger problem into smaller parts and apply MIP to optimize the smaller parts, thus reducing the running time from exponential to linear. Such an operation allows similar handling of order regardless of their size, making it flexible for dealing with larger and more difficult-to-pack orders without modifications. The described algorithm can be incorporated into existing cloud-based and offline solutions for package recommendations for any merchant. Implementation of the algorithm can result in more efficient and effective order packing that

reduces costs, minimizes product damage, increases customer satisfaction, and promotes sustainable operation.