CSIS 0801 Final Year Project Interim Report

Student: Fang, Yimai (ymfang@cs.hku.hk)
Supervisor: Prof. Francis Chin
Topic: Research on online bin packing with vehicle routing
Abstract: A progress report on what has been done in the first stage: discoveries, difficulties and efforts made, with plan about the next stage.

1. Offline algorithms and difficulties

Initially I was concerned with offline two-dimensional bin packing with one-dimensional vehicle routing. The problem is defined by a set of items, each having a height, a width and a destination distance within [0,1]. The destinations are aligned on the same road, along with the depot (starting point) on one side. The goal is to pack items into vehicles of $1 \times 1$ loading space, drive the vehicles to the destinations and minimize the sum of travelled distances (Figure 1).

Considering the absence of the last-in-first-out (LIFO) constraint, the distance a vehicle needs to travel is determined by the farthest item in it: supposing the farthest item has destination distance $d$, the travelled distance of the vehicle is $2d$ (due to round-trip).
Therefore in this one-dimensional (one-directional) vehicle routing, we can ignore the returning trip and only consider how to minimize the one-way distances. If not stated otherwise, “travel distance” in one-directional case shall mean the one-way distance. One more clarification is that one bin corresponds to one vehicle, so the article would use “bin” and “vehicle” interchangeably.

In finding the approximation ratio, it is essential to define a lower bound (OPT) for optimal solution, so that we can say an algorithm (ALG) has approximation ratio no bigger than a number, by comparing ALG to OPT. I define OPT to be the sum of items sizes multiplied with items distances; because each item must at least occupy a loading space of its size and packed in a vehicle traveling to at least its destination. No algorithms can do better than “moving” items to their destinations individually.

A naïve randomized approach was then devised by me, inspired by the randomized algorithms course. For the items, assume that the heights, width and distances are in uniform distribution of $[0,1]$. The algorithm works in the recursive way:

- While there are unpacked items (assuming the number of items is in multiplies of 4), randomly pick 4 unpacked items and try to pack them in a bin (assuming that we know the best way to pack them).
  - If they are successfully packed, the travel distance (as the maximum of 4 random variables) of the vehicle is in Beta distribution with $\alpha=4$, $\beta=1$, whose expected value is 0.8.
  - If unsuccessful, randomly break them into two groups of two items, and try to pack the two groups in two vehicles.
    - If both packing are successful, the travel distance (as the maximum of 2 random variables) of the vehicles are in Beta distribution with $\alpha=2$, $\beta=1$, whose expected value is 2/3.
    - If any packing fails, pack the two items concerned in individual vehicles, whose expected travel distance is 0.5.

Obviously, the average performance of the algorithm depends on the failure probability of packing 4 items in a vehicle and that of packing 2 items in a vehicle. Finding these two probabilities requires geometry insights, and is non-trivial. If, however, we denote them as $p_1$ and $p_2$, we can obtain such a table (Table 1).

<table>
<thead>
<tr>
<th>Bin Type</th>
<th>Success probability</th>
<th>Expected number of such bins</th>
<th>Expected travel distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bins of four</td>
<td>$p_1$</td>
<td>$p_1n/4$</td>
<td>4/5</td>
</tr>
<tr>
<td>Bins of two</td>
<td>$p_2$</td>
<td>$(1-p_1)p_2n/2$</td>
<td>2/3</td>
</tr>
<tr>
<td>Bins of one</td>
<td>1</td>
<td>$(1-p_1)(1-p_2)n$</td>
<td>1/2</td>
</tr>
</tbody>
</table>
Table 1. Probabilistic information on 3 kinds of bins.

Summing up by linearity of expectation, the expected total travel distance is
\[ \frac{1}{5}p_1 + \frac{1}{3}(1-p_1)p_2 + \frac{1}{2}(1-p_1)(1-p_2) \cdot n. \]
Suppose that \( p_1 = 0.6 \) and \( p_2 = 0.9 \) (these are totally arbitrary values for illustration only), then this value would be \( 0.26n \). Since the optimal solution would use a total distance no smaller than \( 1/8 \cdot n \), the average approximation ratio would be no bigger than \( 2.08 \).

The problems with this approach are: (1) the average ratio depends on the distribution of item heights, widths and distances, and the worst case is not discussed; (2) \( p_1 \) and \( p_2 \) are difficult to compute, and the computation process is still based on the distribution of item heights and widths, thus may not be generic.

My second approach was based on “Round and Approx” (R&A), a general method proposed by Bansal, Caprara and Sviridenko. A feasible way of choosing and packing items in a vehicle is called a configuration, associated with a cost (in this problem, the cost is the travel distance determined by the farthest item). The goal is to find a set of configurations that covers all items, with minimal total cost.

Thus the problem is converted to a set cover problem. The main idea is to solve (by duality) the set cover problem first in a linear programming (LP) relaxation to get a fractional optimal solution. Then we apply randomized rounding to obtain part of the final solution. Now that some items are no longer covered due to rounding, an approximation algorithm has to be applied on residual instances to obtain the other part of the solution. Their paper (Bansal, Caprara, & Sviridenko, 2006) focused on enhancing a \( \rho \)-approximation algorithm with LP to achieve \( \ln \rho + 1 \) approximation.

An obvious issue with the approach, which I overlooked, is that the number of configurations is exponential to the number of items. This implies that even if we are given the configurations, simply scanning through them would not be in polynomial time! The paper devised a way (details are tedious and not mentioned in the paper) to check only polynomial number of configurations, but it is based on the assumption that configurations are unweighted. Thus I would need to find my way to reduce the number of configurations. Of course, since items cannot be duplicated in different vehicles, once we select a configuration we can skip all those configurations that contain covered items. This is suitable for dynamically generating configurations, like in a greedy algorithm, rather than providing a complete set of configurations to an LP solver. So if research on this approach should carry on, it would be best to focus on greedy algorithms (e.g. iteratively finding the next most preferred configuration).
2. Problem definition

Following the abovementioned difficulties, the focus of research transferred to the online version of the problem. The key difference, as implied by “online”, is that the items are given one by one, not at once, so that an item must be packed somewhere before the next item comes. The “online” does not mean that the vehicles are departing after a certain time or event – they are supposed to depart together after all items are packed, although this variation would also be of research interest. To put it simple, it is “online bin packing” with “offline routing”.

For simplicity, the routing is still one-directional before Section 5. The article is mainly concerned with two-dimensional packing, but the idea should be applicable to any packing algorithms. As a reminder, the problem is defined by a set of items (given in a sequence), each having a height, a width and a destination distance within (0,1], and we wish to pack them into vehicles of $1 \times 1$ loading spaces, drive the vehicles from one end of the road to the destinations aligned on the same road, and minimize the sum of travelled distances. LIFO constraint is not considered. The items cannot be rotated, except for rotating 90 degrees (exchanging height and width).

There are subtleties in defining “online”: certain information about the items is useful in making packing decisions, and if this is known before the items are packed in sequence, we call it “semi-online”. For example, a courier may know all the orders of the day, and then pack items online (pack as they arrive). Also, sometimes it would be beneficial to re-pack some items, e.g. splitting a bin into two when more items come rather than opening new bins for new items. This is called “online packing with stages” because there is an offline process between each stage or after the final stage, similar to allocating a new space when a dynamic array grows beyond its capacity. The online algorithms discussed below are canonically “online” unless stated otherwise.

3. Heuristics and results

The algorithm is based on the packing-in-strips algorithm of (Coppersmith & Raghavan, 1989), since this online two-dimensional bin packing algorithm is easy to analysis. Later in Section 4 we shall see that for almost any given online bin packing algorithm, we can plug it into the framework and obtain the upper bound of the approximation ratio.

A brief review of the packing-in-strips algorithm is provided here for reference: Items are first rotated so that height is no bigger than width, and then are categorized into three kinds, type-1 if height is more than 1/2, type-2 if height is close to but no more than $(1/2)^k$, and type-3 if height is close to but no more than $(1/3)^k$. $(k>0$ is a natural number.) Vehicles are
also in these three kinds, containing horizontal strips of heights corresponding to the type. Items are first placed to the most suitable used strip, preference followed by the most suitable unused strip, and then generating new strips by splitting bigger strips or the whole unallocated vehicle. The goal of packing in strips is to limit wasted spaces, and achieves approximation of upper bound $3.25 \text{OPT} + 8$. Note that in their paper, OPT is defined more accurately, as the maximum of our OPT and the effect of transporting all type-1 items in individual vehicles. I will use this more accurate definition of OPT in the analysis of Section 4, but for the other parts, my “rough” OPT is used because the statistics are based on it. This discrepancy will be eliminated in the final paper, by re-computing.

Inspired by (Csirik, Frenk, & Labbe, 1993), I tried to modify the algorithm in defining a tie-breaking criterion. For example, when there are multiple most suitable used strips, the original algorithm did not define which strip to choose, while I can require it to take the one with least increment on vehicle travel distance. There are two heuristics: (1) Greedy, or least increment. Since the goal is to minimize total distance, we would not like every vehicle going to the farthest destination, but rather keep the distance as low as we can. So this heuristic picks the vehicle such that by packing the item into it, the least amount of travel distance increment would occur (preferably zero). (2) Concentration. Following the idea that items in the neighborhood should be packed together, this heuristic picks the vehicle whose items have the average distance most close to the item in question. The concentration heuristic seems to be more visually plausible, especially in 2-dimensional routing on a map, because it is similar to what people might do: clustering destinations into delivery regions.

To test my heuristics, I generated 100 test cases, each containing 1000 items whose width, height and distance are in uniform distribution within (0,1]. I implemented the algorithm in C#, with WPF to visualize the packing result. The average performances of the two heuristics are very similar: approximation ratio is 1.9457 on average using the least-increment heuristic, and 1.9435 using the concentration heuristic. And there are similar numbers of test cases where either heuristic outperforms the other. The packing decisions and strips are depicted in (Figure 2) as an illustration; for example, in both results, bin #0 is a type-3 bin, bin #1 is a type-2 bin and bin #2 is a type-1 bin.
While the heuristics provide a guideline on tie-breaking, it is hard to predict the worst case performance (with regards to various distributions of item heights, widths and distances), and it is not strong enough – as is shown below, more rigid rules should be placed on items’ distances in packing decisions.

4. General framework

The simple notion behind the framework is that items of different distances should be tackled as different sub-problems, so that we can easily bound the worst case distance in each sub-problem. Thus the sub-problems are left to any existing bin packing algorithms to solve without caring about distances.

For the one-directional routing, we shall divide the road into infinite number of sections, by the distance from the depot. We set a fineness parameter $0 < \eta < 1$, and define these sections: $(\eta, 1], (\eta^2, \eta], (\eta^3, \eta^2], \ldots$ (Figure 3) Items with distance in different sections are treated as of different sub-problems, thus essentially the vehicles are categorized by the sections. For example, when $\eta = 0.5$, an item of distance 0.6 can only be packed in a vehicle that drives to the first section.

This enforces a stronger bounding so that not all vehicles will need to drive to the farthest customer, and the number of vehicles heading for remote places is limited. I tested the algorithm on the same 100 test cases, and with $\eta = 0.877$ the average approximation ratio is 1.7359, a significant improvement to the two heuristic algorithms. The packing result is
depicted in (Figure 3) as an illustration as well, and the stronger concentration by distance is clearly indicated by the similar colors within each bin. But it is clear that this algorithm potentially wastes more loading space, like in bin #2, thus having vehicles driving to near-by sections at the cost of using more vehicles. On average, 454 vehicles are used for the 1000 items, while for the old algorithms the number was 403 and 404.

The significance of the new algorithm is more than better average performance, but it can be analyzed to establish an upper bound on its approximation ratio in the worst case. Denote the OPT (the abovementioned “more accurate” one, as defined in (Coppersmith & Raghavan, 1989)) of sub-problem $i$ (the $(i+1)$-th section) as $a_i$. Then in the worst case, the items in the first section all have distance slightly larger than $\eta$, while our algorithm treat them as if they all have distance = 1. The same applies to other sections: OPT is bounded by the lower end of the section and ALG is bounded by the higher end. Suppose we plug in an online bin packing algorithm whose $ALG \leq \alpha \cdot OPT + \beta$, where $\alpha$ and $\beta$ are nonnegative constants. Therefore $OPT > \eta a_0 + \eta^2 a_1 + \eta^3 a_2 + \cdots$ and $ALG \leq a a_0 + \beta + \eta(aa_1 + \beta) + \eta^2(aa_2 + \beta) + \cdots < \frac{1}{\eta} \cdot \alpha \cdot OPT + \frac{\beta}{1-\eta}$. Taking the online 2D bin packing algorithm in (Coppersmith & Raghavan, 1989) as an example (i.e. $\alpha = 3.25, \beta = 8$), and letting $\eta = 0.5$, we immediately get a 6.5 $OPT + 16$ bound.

Here it is clear that we should find the best $\eta$ such that $\frac{1}{\eta} \cdot \alpha \cdot OPT + \frac{\beta}{1-\eta}$ is minimized, and the upper bound is thus tightened. This can be achieved by calculus: Minimum is reached when $\left(\frac{1}{\eta} \cdot \alpha \cdot OPT + \frac{\beta}{1-\eta}\right)' = -\eta^2 \alpha OPT + (1 - \eta)^{-2} \beta = 0$. Solving the equality, $(\alpha OPT - \beta) \eta^2 - 2 \alpha OPT \eta + a OPT = 0 \Rightarrow \eta = \frac{2a OPT + \sqrt{4a^2 OPT - 4 \alpha OPT \beta}}{2(\alpha OPT - \beta)}$. Considering that $0 < \eta < 1$, pick $\eta := \frac{\alpha OPT - \sqrt{\alpha OPT^{2} - 4a OPT \beta}}{a OPT - \beta}$. Substituting it back, $\frac{1}{\eta} \cdot \alpha \cdot OPT + \frac{\beta}{1-\eta}$ becomes $\frac{(\alpha OPT - \beta)^2}{\sqrt{\alpha OPT^{2} - 4a OPT \beta}}$ after a tedious simplification. That is to say, if we pick the right $\eta$, the approximation result (total

![Figure 4. Packing results (first 8 vehicles shown) for the first test case.](image)
distance) is at most $ALG \leq \frac{(\alpha OPT - \beta)^2}{\sqrt{\alpha OPT - \beta}}$. Note that when $OPT \to +\infty$, this is $OPT$ and $\eta \to 1$. This can be naturally foreseen, since if there are infinitely many items, the division of road sections should be really dense and every section has sufficient number of items to use up the $\beta$ extra vehicles. This also explains why the algorithm is especially wasteful when dealing with small number of OPT: if there are too many sections, some sections may only contain few items and generating a series of almost empty vehicles.

Moreover, the value of $\eta$ has to be defined before all packing starts, we need to know $OPT$ beforehand (like in a list of orders of items). This is semi-online. For the real online problem, the idea is to use a “golden” constant $\eta$ (large enough) that is good for large OPTs, and this would not hurt the ratio very much even if it is not the best. Small-OPT problems are much more sensitive to this $\eta$, and we can opt for other online methods until the items packed reach the OPT threshold (accumulative knowledge of how much is being packed is needed). This is left for the researches in the next stage.

5. Onto graphs or grids

There are three ways in which the method could be applied to routing on a graph or a two-dimensional grid, and the basic idea shall still be categorizing. These are open for further researches to be carried out in the following month.

(1) This is intended for a graph where destinations (customers) and the depot are vertices
connected by bi-directional edges of certain costs. To simplify, we can first consider that edges are of uniform costs, and the degree of a vertex cannot exceed 3. Then we can traverse the graph from the depot using BFS (or uniform cost search, if the edge costs are not uniform) to obtain a binary tree rooted on the depot that shows the shortest path from the depot to each destination. The optimal distance of each item is bounded by the length of the shortest path. We can then form categories by specifying a range of lengths and/or a number of initial steps that must be shared. For example, Figure 6 illustrates that if paths to two destinations share the first two steps, and are of 1~3 edges, the two are placed inside a category; and in the worst case a vehicle of this category needs to travel on all the 4 edges. This method controls that the worst case travel distance is linear to the best case travel distance, simulating the one-directional solution. However, the number of categories grows too fast in the exponential fashion.

Figure 6. A way of forming categories by path length and initial steps.

(2) This is intended for a 2D grid of customers and the depot is placed at the origin (center). The idea is to divide the map space into concentric squares, and let each “ring” of width \( d \) be divided into 4 regions by the quadrants. Considering only one quadrant. Suppose that the inner most region is Region 1, and the outer ones are Region 2, Region 3... Then the optimal round-trip distance for going to destinations in Region \( k \) would be at least \( 2(k-1)d+2 \). In the worst case the vehicle has to travel in a zigzag way to reach every possible destination in that region (Figure 7), and note that the points on the inner edges belong to Region \( k-1 \), thus round-trip distance is at most \( (2k-1) (d-1)^2 + 2kd \). Comparing the best case and the worst case, a bound may be established.

Figure 7. The worst case travel route inside Region 3 (shaded) of the first quadrant. (d=2)
In (2), the destinations are on an integer coordinates. Here they can be real numbers and the route is also drawn on a continuous plane (displacement vectors not limited to horizontal and vertical ones). The idea is to cluster destinations with a suitable bound on similarity, so that the worst case distance will not differ too much from the best case. Clustering with bounded similarity, however, has been proven NP-hard, and an $O(\log n)$ approximation is feasible where $n$ is the number of destinations (Hasan, Salem, Pupacdi, & Zaki, 2009). Although $O(\log n)$ does not seem to be good enough, but in fact (2) arguably depends on or restricts on the number of destinations as well.

6. References


