The Train Delivery Problem - Vehicle Routing Meets Bin Packing

Aparna Das*, Claire Mathieu**, and Shay Mozes

Brown University, Providence RI 02918, USA.

Abstract. We consider the *train delivery* problem which is a generalization of the bin packing problem and is equivalent to a one dimensional version of the vehicle routing problem with unsplittable demands.
The train delivery problem is strongly NP-Hard and does not admit an approximation ratio better than 3/2. We design two types of approximation schemes for the problem. We give an *asymptotic* polynomial time approximation scheme, under a notion of asymptotic that makes sense even though scaling can cause the cost of the optimal solution of any instance to be arbitrarily large. Alternatively, we give a polynomial time approximation scheme for the case where W, an input parameter that corresponds to the bin size or the vehicle capacity, is polynomial in the number of items or demands. The proofs combine techniques used in approximating bin-packing problems and vehicle routing problems.

1 Introduction

We introduce the *train delivery* problem, which is a generalization of bin packing. The problem can be equivalently viewed as a one dimensional vehicle routing problem (VRP) with unsplittable demands, or as a specific kind of an offline scheduling problem.

In the VRP setting with unsplittable demands, containers are to be transported from Seattle's harbor to n customers located at positions p_i along a railway that extends into the mainland. The number of containers destined to customer i is w_i, and the maximum number of containers that the freight train can carry is W. All containers destined to a particular customer must be placed on the same train so that they are delivered at the same time. We wish to find a set of train trips to deliver all containers so as to minimize the total length of all trips.

In the bin packing setting, various temperature sensitive products are shipped by sea from southeast Asia to the US. Each product has a weight (in metric tons) and a maximal temperature at which it may be stored (there is no minimum temperature limit). Each ship can carry at most W tons. Since the route is fixed, the cost of operating the ship is determined by the ambient temperature in the cargo area. The lower the temperature, the higher the cost (this can be

* Supported by NSF grant CCF-0728816.
** Supported by NSF grant CCF-0728816.

an arbitrary monotone function). The shipping company is interested in keeping the cost of operations as low as possible while keeping the temperatures low enough so none of the products on board a ship are damaged. The goal is to find a packing of the products in ships so that the overall cost of operating all of the ships is minimal.

Formally, in the train delivery problem we are given a positive integer capacity W and a set S of n items, each with an associated positive position p_i and a positive integer weight w_i. We wish to partition S into subsets $\{S_j\}$ (train tours) so as to minimize

$$\sum_j \max_{i \in S_j} p_i \quad \text{subject to} \quad \forall j \ \sum_{i \in S_j} w_i \leq W.$$

Bin-packing is the special case of the train delivery problem where all the p_i's are equal. It is well known [12] that bin-packing is strongly NP-hard and does not admit a polynomial time approximation algorithm with approximation ratio better than 3/2 unless P=NP, hence those negative results also hold for the train delivery problem. There are, however, algorithms that achieve an approximation factor of $1+\epsilon$ for bin-packing for any $\epsilon > 0$, provided that the cost of an optimal solution is at least $1/\epsilon$ (that is, at least $1/\epsilon$ bins are necessary). Such algorithms are called asymptotic polynomial time approximation schemes (APTAS).

The train delivery problem does not admit an asymptotic approximation scheme in the usual sense. The reason is that the cost of the solution is determined by the positions p_i, so any instance can be scaled so that the cost of an optimal solution is arbitrarily large without changing the solution itself. Therefore, to define a notion of asymptotic approximation scheme for our problem we restrict the ratio of the optimal solution and the maximal position.

Theorem 1. *Given an instance of the train delivery problem such that* $\max_i p_i = O(\epsilon)\,OPT$*, Algorithm 1 outputs a solution of cost* $(1+O(\epsilon))\,OPT$ *in time* $n^{(1/\epsilon)^{O(1/\epsilon)}}$.

In other words, scale the input so that $\max_i p_i = 1$; then we are in the asymptotic regime if the scaled input has optimal cost at least $1/\epsilon$. Given an instance of the train delivery problem we can use any constant factor approximation of the optimal solution to check whether the conditions of Theorem 1 hold. For example we can use the constant factor approximation for the metric vehicle routing problem with unsplittable demands given by [14].

Alternatively, we give a polynomial time approximation scheme (PTAS) for the case where $W = poly(n)$ (or where W is specified in unary). Note that bin-packing is still NP-hard for such instances.

Theorem 2. *Given an instance of the train delivery problem, Algorithm 4 outputs a solution of cost* $(1+O(\epsilon))\,OPT$*. Its running time is* $W^{e^{O(1/\epsilon)}} + n^{(1/\epsilon)^{O(1/\epsilon)}}$ *which is polynomial in* n *and* W.

We note that, unless P=NP, we cannot hope to achieve a PTAS when the conditions of Theorem 2 do not hold. A PTAS that also works when $W > poly(n)$

would give us a polynomial time algorithm, rather than a pseudo polynomial time algorithm, for deciding the NP-hard *partition* problem[1].

Related work. Both bin-packing and vehicle routing are extensively studied in the literature. We do not attempt to provide a comprehensive survey, but focus mostly on the works whose techniques we use in this paper. Bin-packing is one of the problems originally shown to be strongly NP hard by Garey and Johnson [12]. Fernandez de la Vega and Lueker [10] obtained the first APTAS. They handle small and big demands separately. The big demands are rounded so that all possible ways to assign big demands to bins can be enumerated in $O(C_\epsilon)$, where C_ϵ is exponentially large in $1/\epsilon$, but does not depend on n. Subsequently, Karmarkar and Karp [16] gave an asymptotic *fully* polynomial approximation scheme (AFPTAS) using the same framework. Instead of enumeration, they show how to efficiently solve and round an LP relaxation of the problem on just the big demands. Their running time depends polynomially on $1/\epsilon$, rather than exponentially. Many variants of bin-packing have been considered, (see [8] for a survey). In multi-dimensional bin-packing (See [6, 17, 3, 7] and references therein), the constraints on the bins are multi-dimensional, but the cost of each bin is still a fixed constant. In variable-size bin-packing (See [11, 19, 9]) bins of several different sizes are available and the cost of each bin is proportional to its size. In bin-packing with "general cost structure"(See [9, 18]), the cost of a bin is a non-decreasing concave function of the number of elements packed in the bin.

There are AFPTAS for all of these variants and all of those we are aware of handle big and small items separately and use rounding of the big items. They differ substantially in the methods used to approximately solve the problem for the big items and how to combine the small items to obtain a near optimal solution. None of these variants, however, captures the problem we consider.

The VRP is another widely studied problem. The train delivery problem is the 1-dimensional version of the VRP with unequal or unsplittable demands [14, 5, 4] where a set of customers, each with its own demand w_i must be served by vehicles which depart from and return to a single depot. Each vehicle may serve at most W demands and each customer must be served by a single vehicle. The objective is to minimize the total distance travelled by all vehicles[2]. In the 1-dimensional version the depot is located at the origin and the position of customer i is given by p_i.

We are not aware of any prior work that specifically considers the 1-dimensional VRP with unsplittable demands. For the metric case Haimovich, Rinnooy Kan, and Stougie give a constant factor approximation [14]. Bramerl et al. give a probabilistic analysis for the Euclidean plane where customer demands are drawn i.i.d from any distribution [5].

For the splittable case, where customers may be served by multiple vehicles, Haimovich and Rinnooy Kan gave a PTAS for the Euclidean plane when

[1] Partition: Given a set of integers $S = w_1, \ldots, w_n$, decide if S can be partitioned into two sets S_1 and S_2 such that the sum of the numbers in S_1 and S_2 are equal.

[2] The VRP objective is 2 times the objective of the train delivery problem

$W = O(\log \log n)$ [13]. Their approximation scheme partitions the customers into two disjoint instances (far and *close*) based on the distance from the depot and solves each instance independently. The far instance is small enough so that it can be solved exactly by brute force, but sufficiently large, so that the error incurred by solving the instances independently is controlled. The *close* instance is "close" enough to the depot such that for small values of W a constant factor algorithm (that they also present) finds a near optimal solution for *close*. Recently, Adamaszek, Czumaj, and Lingas extended this to $W \leq 2^{\log^{\delta} n}$ (where δ a function of ϵ) [1]. Their algorithm partitions the instance into disjoint regions based on distances from the depot, and solves the problem in each region independently. Their analysis uses a shifting technique [2, 15].

Main techniques. To achieve Theorem 1, we first deal with big demands. We round their positions geometrically, and apply the bin packing rounding scheme from [10] at each position to get a small number of distinct big demands. We then use a scheme similar to [1] to partition the items into disjoint regions and solve the problem for each region independently. A shifting technique as in [1] shows that if we do this for a few different partitions, at least one of them will yield a near-optimal solution for the original instance. In each region of the partition, we exhaustively enumerate all solutions for just the big demands, extend each by greedily adding the small demands, (as is done in bin packing algorithms), and output the lowest cost solution. The crux of our analysis lies in showing that it is possible to construct a near-optimal solution by greedily inserting the small demands to the big demand solutions that we enumerate. See Section 2 for details.

To achieve Theorem 2, we partition the instance into *close* and *far* instances and solve the two resulting instances independently, as was done for the splittable VRP problem in [13]. Our *far* instance is small enough to solve it exactly by dynamic programming, and our *close* instance is solved by Theorem 1. The crux of the analysis is a structural lemma which is an extension of [13] to the unsplittable case. See Section 3 for details.

Preliminaries. For the remainder of the paper we use the language of the vehicle routing problem: We refer to tours (rather than sets), customers (rather than items), locations (rather than positions) and demands (rather than weights). We assume the existence of a depot at the origin.

Without loss of generality we assume that all customers are located to the right of the depot. Otherwise we can solve the right and left sides of the depot separately, as they are analogous to one another, and return the union of the two solutions.

We will use the following lower bound from [14].

Lemma 1. *[14](Rad.) Given an instance I of the train delivery problem, any feasible set of tours that covers all customers must have cost at least*

$$Rad(I) = \frac{2}{W} \sum_{i \in I} p_i \cdot w_i.$$

2 Algorithm for Theorem 1

Our algorithm is summarized in Algorithm 1. We present the high level description first followed by the details of each step.

Algorithm 1 Asymptotic PTAS for train delivery

Input: customers $(p_i, w_i)_{1 \le i \le n}$, train capacity W
Precondition: $\max_i p_i \le \epsilon \mathrm{OPT}$

```
1: Round the input using Algorithm 2.
2: for 1 ≤ i ≤ 1/ε do
3:    Let P_i be the i-th partition into regions (as in Definition 1).
4:    for each non-empty region R of P_i do
5:       for each feasible configuration of R for big demands (as in Definition 3) do
6:          Extend it to a solution for all demands using Algorithm 3.
7:       end for
8:       Let Best(R) be the minimum cost solution after step 7.
9:    end for
10:   Let Best(P_i) = ∪_{R∈P_i} Best(R) be the solution for P_i.
11: end for
```

Output: $\min_i \mathrm{Best}(P_i)$, the minimum cost solution found.

Rounding. We reduce the number of locations by rounding each location up to the next integer power of $(1+\epsilon)$. We call demand w_i big if $w_i \ge \epsilon W$ and small otherwise. We use the classical rounding technique from bin packing algorithms to reduce the number of distinct big demands at each location.

Partitioning into regions. We partition the customers into disjoint regions based on their distance from the depot (Definition 1) so that each region has only a constant number of locations containing customers. We solve the problem approximately within each region independently. Using a shifting argument, we show that if we do this for a few different partitions, the union of the individual approximate solutions in at least one of the partitions yields a near optimal solution for the original instance.

Solving within a region. Within each region, we treat big and small demands differently. Since each region R contains just a constant number of locations and each location contains a constant number of distinct big demands, the total number of distinct big demands in R is constant. This allows exhaustive enumeration of all solutions for the big demands in R. We extend each solution by adding the small demands greedily, and output the solution with the lowest cost. We prove that the solution we output has cost at most $(1+2\epsilon)\mathrm{OPT}(R) + 2p_R$, where $\mathrm{OPT}(R)$ denotes the optimal solution of region R and p_R is the location furthest from the depot in R (Lemma 7).

Our definition of regions ensures that p_R decreases geometrically as the regions get closer to the depot. Thus the sum of p_R over all regions is at most $O(p_{\max})$, where $p_{\max}$ is the location of the furthest customer. Our assumption

$p_{\max} \leq \epsilon \mathrm{OPT}$ ensures that the additive cost incurred by the greedy extension (the p_R terms) is within the desired approximation factor.

We now discuss each of the steps in more detail. Rounding is performed using Algorithm 2. The analysis relies on the following lemma whose proof (in

Algorithm 2 Rounding Instance

Input: train capacity W, customers $(p_i, w_i)_{1 \leq i \leq n}$

1: Round each p_i up to the smallest (possibly negative) integer power of $(1+\epsilon)$.
2: Partition demands $(w_i)_i$ into *big* ($\geq W\epsilon$) and *small* ($< W\epsilon$).
Rounding big demands:
3: **for** each location ℓ with $n_\ell \geq 1/\epsilon^2$ big demands **do**
4: Go through those big demands in decreasing order to partition them into ϵ^{-2} groups such that each group (except possibly one) has cardinality exactly $\lfloor n_\ell \epsilon^2 \rfloor$.
5: **for** each group g **do**
6: Round every demand in g up to the maximum demand in g.
7: **end for**
8: **end for**

Output: rounded instance I' of the train delivery problem

the appendix) is an extension of the bin packing rounding analysis by Fernandez de la Vega and Lueker [10].

Lemma 2. *Given an instance I of the train delivery problem, Algorithm 2 outputs an instance I' such that:*

- *Each p_i has the form $(1+\epsilon)^k$ for some (possibly negative) integer k.*
- *Each location has at most $1/\epsilon^2 + 1$ distinct big demands.*
- *Any feasible solution for I' is also feasible for I.*
- *$OPT(I') \leq (1 + O(\epsilon))OPT(I)$.*

We partition the instance into regions, solve the problem in each region independently and output the union of the solutions. We use a *shifting* technique similar to that of Baker [2] and Hochbaum and Maass [15] to show (Lemma 3) that at least one of the shifted partitions yields a near optimal solution.

Definition 1. *Let I be an instance of the train delivery problem and $p_{\max} = \max_i p_i$. A* block*, defined by an integer i, is the set of customers with locations in $(p_{\max}\epsilon^{i+1}, p_{\max}\epsilon^i]$. A* region *is a group of at most $1/\epsilon$ consecutive blocks.*

For $0 \leq j < 1/\epsilon$, let P_j denote the partition of I into regions, one initial region $(\epsilon^j p_{\max}, p_{\max}]$ and the other regions $(\epsilon^j p_{\max}\epsilon^{(i+1)/\epsilon}, \epsilon^j p_{\max}\epsilon^{i/\epsilon}]$ for $i \geq 0$.

Lemma 3. *Let partitions be as given in Definition 1. There exists a partition P_i such that $\sum_{R \in P_i} OPT(R) \leq (1 + O(\epsilon))OPT$.*

To prove Lemma 3 we first show that a near optimal solution can be obtained using tours which cover points in a bounded region.

Definition 2. *A tour that covers only points between locations p and p', $p \leq p'$, has* expanse p'/p. *A* small *expanse tour has expanse at most* $1/\epsilon$.

Lemma 4. *Let I be an instance of the train delivery problem. There exists a solution using only small expanse tours which costs at most $(1+2\epsilon)OPT(I)$.*

The proof appears in the appendix. Small expanse tours have a simple structure as they cover points in at most two blocks (Recall that the expanse of a block is $1/\epsilon$). Intuitively, only a few tours of the optimal small expanse solution will cover points in more than one region. This allows us to use an averaging argument to prove Lemma 3 (see appendix).

A configuration for a region is a concise description of a solution for the big demands in that region.

Definition 3. *(Configuration) Fix a region R.*

A demand type *is a pair (p, b) where p is the location of a big demand and $b \leq 1/\epsilon^2$ is one of the values of big demands at location p.*

A tour profile *consists of a location r, which is the rightmost location of the tour, and of a multiset of demand types whose demands sum to at most W and whose locations are all at most r. Let c_p be the number of distinct tour profiles.*

A configuration *is a list $(\beta_i)_{i=1}^{c_p}$ s.t. β_i is the number of tours with profile i.*

The profile of a tour roughly describes which points it will cover: If a profile contains a demand type (p, b) in its multiset, then the tour covers an (arbitrary) big demand from location p with demand b.

Definition 4. *A configuration of R is* feasible *if it covers all big demands in R.*

Lemma 5. *The number of configurations of a region R is at most $n^{(1/\epsilon)^{O(1/\epsilon)}}$.*

With each configuration $(\beta_i)_i$, we associate a list T of tours, $(r_t, c_t)_t$ covering big demands, where r_t denotes the maximum location of tour t, and c_t denotes its remaining capacity after it has covered the big demands. Given a feasible configuration, Algorithm 3 takes the associated list of tours $(r_t, c_t)_t$ as input and extends the solution to also cover the small demands of R.

We turn to show that the greedy extension results in a near optimal solution. This is the crux of the whole analysis. The following lemma is a generalization of the Rad bound (Lemma 1) of Haimovich, Rinnooy Kan, and Stougie [14].

Lemma 6. *Let $(p_i, w_i)_i, T$ be the input of Algorithm 3. Let $OPT|T$ be the minimum cost extension of T to a set of tours that covers the small demands (p_i, w_i). For any increasing sequence $(x_s)_s$ such that $x_0 = 0$, we have*

$$Cost(OPT|T) \geq Cost(T) + \frac{2}{W} \sum_{s \geq 1} (x_s - x_{s-1}) \Big(\sum_{i : p_i \geq x_s} w_i - \sum_{t \in T : r_t \geq x_s} c_t \Big).$$

Proof. Let $\rho_s = \sum_{i:p_i \geq x_s} w_i$ denote the total small demand at locations at least x_s. Let $T_s = \{t : r_t \geq x_s\}$ denote the set of input tours that cover some point at location at least x_s. Let γ_s denote the total available capacity of all input tours

Algorithm 3 Greedy Extension

Input: small-demand customers $(p_i, w_i)_i$, list T of tours $(r_t, c_t)_t$ covering big demands s.t tour t has maximum location r_t and remaining capacity c_t.

1: **for** each small demand (p_i, w_i) by order of decreasing p_i **do**
2: **if** there is a tour t with $r_t \geq p_i$ and $c_t \geq w_i$ **then**
3: cover (p_i, w_i) with t and set $c_t := c_t - w_i$
4: **else**
5: add a new tour t with $c_t = W$ and $r_t = p_i$
6: cover (p_i, w_i) with t and set $c_t := c_t - w_i$
7: **end if**
8: **end for**

Output: the resulting tours.

that cover some point at location at least x_s (i.e., $\gamma_s = \sum_{t \in T_s} c_t$). The amount of small demand at locations greater than x_s that cannot be covered by T is $\rho_s - \gamma_s$. Therefore, the number of tours that any extension must add on top of the input tours in order to cover all the demands at locations greater than x_s is at least $\frac{1}{W}(\rho_s - \gamma_s)$. This yields the desired lower bound.

Lemma 7. *Let G be the output of Algorithm 3 on input $(p_i, w_i)_i, T$. Let $OPT|T$ be as in Lemma 6. Then cost$(G) \leq (1 + 2\epsilon)$cost$(OPT|T) + 2p_{\max}$.*

Proof. Consider the new tours added by algorithm 3. Let $(x_s)_{s \geq 1}$ be the set of maximum locations for these tours, sorted in increasing order, and define $x_0 = 0$ for convenience. Let A_s denote the set of additional tours, added by the algorithm, whose maximum point is at least x_s. We have:

$$\text{cost}(G) = \text{cost}(T) + \sum_{s \geq 1} 2(x_s - x_{s-1})|A_s|. \tag{1}$$

Consider x_s. We use the same notations as in the proof of Lemma 6. By the condition in line 1 of the algorithm, since the demands are small and since a new tour is added by the algorithm at location x_s, it must be that every tour $t \in T_s$ has remaining capacity at most ϵW in G. Thus the amount of small demand assigned by G to the tours in T_s is at least $\gamma_s - |T_s|\epsilon W$, and so the amount of small demand assigned by G to new tours is at most $\rho_s - \gamma_s + |T_s|\epsilon W$.

Since G does not open another additional tour until all existing tours (of A_s as well as of T_s) are almost full, we have:

$$|A_s| \leq \frac{\rho_s - \gamma_s + |T_s|\epsilon W}{(1-\epsilon)W} + 1. \tag{2}$$

Substituting (2) into (1) and using Lemma 6 we get that cost(G) is at most:

$$\text{cost}(T) + \frac{1}{1-\epsilon}(\text{Cost}(OPT|T) - \text{Cost}(T)) + \frac{\epsilon}{(1-\epsilon)} \sum_{s \geq 1} 2(x_s - x_{s-1})|T_s| + 2 \max_s x_s.$$

Since $\text{cost}(T) \geq \sum_{s\geq 1} 2(x_s - x_{s-1})|T_s|$ and $\max_s x_s \leq p_{\max}$, we obtain

$$\text{cost}(G) \leq \frac{1}{1-\epsilon}\text{Cost}(OPT|T) + 2p_{\max}.$$

Correctness of Algorithm 1. By Lemma 2 the optimal solution of the rounded instance is a near optimal solution for the original instance. To solve the rounded instance Algorithm 1 tries all $1/\epsilon$ partitions of it into regions. Lemma 3 shows that for at least one of these partitions, P^*, a near optimal solution is obtained by solving in each region independently and combining the solutions. For the rest of the analysis, focus on the execution of Algorithm 1 that uses partition P^*. Let $R_1^*, R_2^*, \ldots, R_r^*$ be the regions of P^*. It remains to show that Algorithm 1 finds a near optimal solution for each region of partition P^*.

Consider a region R_i^* of P^*. Algorithm 1 enumerates all feasible configurations for covering the big demands in R_i^*. In particular, it considers the configuration that agrees with the tours of $\text{OPT}(R_i^*)$ for covering big demands in R_i^*, which we denote by C^*. Algorithm 1 invokes Algorithm 3 on C^* to produce a solution whose cost, by Lemma 7 is at most $(1+2\epsilon)\text{OPT}(R_i^*) + 2p_{R_i^*}$, where $p_{R_i^*}$ is the maximum location in R_i^*.

Applying the above argument to all regions in P^*, and using Lemma 3, Algorithm 1 finds a solution of cost

$$\sum_{i\leq r}(1+2\epsilon)\text{OPT}(R_i^*) + 2p_{R_i^*} = (1+O(\epsilon))\text{OPT} + 2\sum_{i\geq 0} p_{R_i^*}.$$

By definition of regions, $p_{R_i^*}$ is at most $p_{\max}\epsilon^{i/\epsilon}$. Thus

$$\sum_{i\geq 0} p_{R_i^*} \leq \sum_{i\geq 0} 2p_{\max}\epsilon^{i/\epsilon} \leq 4p_{\max} \leq O(\epsilon)\text{OPT},$$

where the last inequality follows by our assumption that $\max_i p_i \leq \epsilon\text{OPT}$.
Running time of Algorithm 1. By inspection, one can see that the bottleneck is the number of configurations, which is $n^{(1/\epsilon)^{O(1/\epsilon)}}$ by Lemma 5.

3 Algorithm for Theorem 2

Algorithm 4 gives the high level description.
Partition into *close* and far instances. We index the customers in decreasing order of their location from the depot and identify a customer i_c. The instance is partitioned into, *close* and far where far contains the customers with indices at most i_c and *close* contains the customers with indices greater than i_c.
Optimal solution of far. The partition is such that far contains $O(W)$ total demand. This implies that an optimal solution of far uses a constant number of tours (Lemma 10). This allows us to enumerate, in polynomial time, all possible such solutions. Using a generalization of the well-known dynamic program for subset sum, we can determine in polynomial time whether a proposed solution is feasible or not, hence an optimal algorithm for far.

Algorithm 4 PTAS for the train delivery problem when $W \leq poly(n)$

Input: train capacity W, customers $(p_i, w_i)_{1 \leq i \leq n}$
Precondition: $W \leq poly(n)$.

1: Partition the instance into *close* and *far* using Algorithm 5
2: Find OPT(far) using Algorithm 6.
3: Find Best(*close*) using Algorithm 1

Output: Best(*close*) $\cup$ OPT(far), as the solution for the whole instance.

Approximate solution of *close*. We use Algorithm 1 to find a near optimal solution to *close*. The cost of the solution is at most $(1+\epsilon)\text{OPT}(close) + 4p_{i_c}$.
Overall solution. The solution is just the union of the solutions for *close* and far. The choice of i_c guarantees the desired approximation factor.
We next discuss each of the steps in greater detail.

Algorithm 5 Partition Close and Far

Input: train capacity W, customers $(p_i, w_i)_{1 \leq i \leq n}$ s.t. $p_1 \geq \cdots \geq p_n$.

1: Let i_c be the smallest index such that
   - $\sum_{j \leq i_c} w_j \geq W/\epsilon$
   - $p_{i_c} \sum_{j \leq i_c} w_j \leq \epsilon \sum_{j=1}^{n} w_j p_j$.

   If no such i_c exist, set $i_c := n$.
2: ***Far:*** Let the far instance consist of the customers indexed by $1, \ldots, i_c$.
3: ***Close:*** Let the *close* consist of the remaining customers $i_c + 1, \ldots n$.

Output: instances far and *close*

Lemma 8. *Given an instance I of the train delivery problem, Algorithm 5 returns two instances* far *and* close *s.t. $OPT(close) + OPT(far) \leq (1 + O(\epsilon))OPT$.*

The proof transforms the optimal solution into separate solutions for *close* and far with small additional cost (see appendix).

To solve the far instance, we first show that the total demand in far is $O(W)$. The following combinatorial lemma is an extension of Haimovich and Rinnooy Kan's [13] analysis to the case with unsplittable demands. It bounds the total demand in the far instance by arguing that there cannot be too many customers that violate the requirement $p_{i_c} \sum_{j \leq i_c} w_j \leq \epsilon \sum_{j=1}^{n} w_j p_j$, see the appendix.

Lemma 9. *Let i_c be as in Algorithm 5. Then $\sum_{j \leq i_c} w_j = \exp(O(1/\epsilon))W$.*

This implies that OPT(far) uses at most a constant c_{far} tours. We then show how to solve such an instance optimally using dynamic programming.

Lemma 10. *Let I be an instance of the problem such that the sum of the demands of all customers in I is D. Then OPT uses at most $\lceil 2D/W \rceil$ tours.*

Definition 5. *(Far Configuration) Let c_{far} denote the maximum number of tours OPT(far) may use. A* configuration *for far consists of:*

- *An ordered list of locations $r_1 \geq r_2 \ldots \geq r_{c_{far}}$, where r_i is the maximum location of the i-th tour.*
- *For every $i \in [1, c_{far}]$, a list S^i of i numbers $S^i = \{s^i_1, \ldots s^i_i\}$.*

The cost *of the configuration is $\sum_{j \leq c_{far}} 2r_j$.*

For $i = 1, 2, \ldots c_{far} - 1$, define an interval $I_i = (r_{i+1}, r_i]$. Let $I_{c_{far}}$ be the interval $[p_{i_c}, r_{c_{far}}]$. The customers in I_i can only be covered by the i tours whose maximum location is at least r_i. The list S^i specifies the total demand from interval I_i that is assigned to each of these tours. Note that S^i does not directly describe how to partition the demands among the i tours. Finding a set of partitions that is consistent with a configuration, or finding out that no such set of partitions exists, is done in $nW^{e^{O(1/\epsilon)}}$ time (i.e., polynomial in n assuming $W \leq poly(n)$)) using a trivial extension of the dynamic program for the subset sum problem (see appendix).

Algorithm 6 solves the far instance by iterating all configurations of OPT(far), checking feasibility, and returning the minimum cost feasible configuration.

Algorithm 6 Solving the far instance

Input: far customers $(p_i, w_i)_i$ with $\sum_i w_i = We^{O(1/\epsilon)}$

for each configuration f of far as given in definition 5 **do**
  for each tour $j \leq c_{far}$ **do**
    if $\sum_{i \leq c_{far}} s^i_j > W$ **then**
      Mark f infeasible. {capacity of tour is exceeded}
    end if
  end for
  for each interval $i \leq c_{far}$, with total demand $dem(I_i)$ **do**
    if Extended DP of subset sum cannot partition $dem(I_i)$ into $s^i_1, \ldots, s^i_i$ **then**
      Mark f infeasible. {demands cannot be partitioned}
    end if
  end for
end for

Output: Solution realized by the minimum cost configuration not marked infeasible.

Lemma 11. *Given a instance of far with demand $We^{O(1/\epsilon)}$ Algorithm 6 finds the optimal solution of far in time $nW^{e^{O(1/\epsilon)}}$, which is polynomial in n and W.*

The choice of p_{i_c} implies the following simple bound on OPT.

Lemma 12. *$OPT > 2p_{i_c}/\epsilon$.*

Proving the Main Theorem 2 is now easy, see appendix for details.

References

1. A. Adamaszek, A. Czumaj, and A. Lingas. Ptas for k-tour cover problem on the plane for moderately large values of k. In *ISAAC*, Berlin, Heidelberg, 2009. Springer-Verlag.
2. B. S. Baker. Approximation algorithms for np-complete problems on planar graphs. *J. ACM*, 41(1):153–180, 1994.
3. N. Bansal, J. R. Correa, C. Kenyon, and M. Sviridenko. Bin packing in multiple dimensions: Inapproximability results and approximation schemes. *Math. Oper. Res.*, 31(1):31–49, 2006.
4. D. J. Bertsimas and D. Simchi-Levi. A New Generation of Vehicle Routing Research: Robust Algorithms, Addressing Uncertainty. *Oper. Res.*, 44(2):286–304, 1996.
5. J. Bramel, Jr. Coffman, Edward G., P. W. Schor, and D. Simchi-Levi. Probabilistic analysis of the capacitated vehicle routing problem with unsplit demands. *Oper. Res.*, 40:1095–1106, November 1992.
6. A. Caprara. Packing 2-dimensional bins in harmony. In *FOCS*, pages 490–499, Washington, DC, USA, 2002. IEEE Computer Society.
7. C. Chekuri and S. Khanna. On multi-dimensional packing problems. In *ACM-SIAM SODA*, pages 185–194, Philadelphia, PA, USA, 1999. Society for Industrial and Applied Mathematics.
8. E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson. Approximation algorithms for bin packing: a survey. pages 46–93, 1997.
9. L. Epstein and A. Levin. An aptas for generalized cost variable-sized bin packing. *SIAM J. Comput.*, 38:411–428, April 2008.
10. W. Fernandez de la Vega and Lueker G. S. Bin packing can be solved within $1 + \epsilon$ in linear time. *Combinatorica*, 1(4):349–355, 1981.
11. D K Friesen and M A Langston. Variable sized bin packing. *SIAM J. Comput.*, 15(1):222–230, 1986.
12. M. R. Garey and D. S. Johnson. " strong " np-completeness results: Motivation, examples, and implications. *J. ACM*, 25(3):499–508, 1978.
13. M. Haimovich and A. H. G. Rinnooy Kan. Bounds and heuristics for capacitated routing problems. *Mathematics of Operations Research*, 10(4):527–542, Nov., 1985.
14. M. Haimovich, A.H.G. Rinnooy Kan, and L. Stougie. Analysis of heuristics for vehicle routing problems. In *Vehicle Routing: Methods and Studies. Management Sci. Systems.*, volume 16, pages 47–61, North Holland, Amsterdam, 1988. Elsevier Science B.V. This is a full inbook entry.
15. D. S. Hochbaum and W. Maass. Approximation schemes for covering and packing problems in image processing and vlsi. *J. ACM*, 32(1):130–136, 1985.
16. N. Karmarkar and R. M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *FOCS*, pages 312–320, Washington, DC, USA, 1982. IEEE Computer Society.
17. C. Kenyon and E. Rémila. A near-optimal solution to a two-dimensional cutting stock problem. *Math. Oper. Res.*, 25(4):645–656, 2000.
18. C.L. Li and Z.L Chen. Bin-packing problem with concave costs of bin utilization. *Naval Research Logistics*, 53(4):298–308, 2006.
19. F.D. Murgolo. An efficient approximation scheme for variable-sized bin packing. *SIAM J. Comput.*, 16(1):149–161, 1987.

A Appendix

A.1 Proof of Lemma 2

Proof. The first three properties are straightforward. We focus on the last property. We first analyze rounding locations . Let I_1 denote the instance obtained from I after rounding just the locations (Line 1). Any length d tour in $\mathrm{OPT}(I)$ can be transformed into a feasible tour for I_1 by extending its length by at most ϵd, so $\mathrm{OPT}(I_1) \leq (1+\epsilon)\mathrm{OPT}(I)$.

Next we analyze rounding demands, by carrying out the de la Vega and Lueker bin packing analysis at each location [10]. Let I" be the instance obtained from I_1 by changing line 6 of the algorithm, rounding demands *down* to the maximum demand of the next group. Clearly, $I" \prec I_1 \prec I'$, and so $\mathrm{OPT}(I') \leq \mathrm{OPT}(I_1) + (\mathrm{OPT}(I') - \mathrm{OPT}(I"))$. However, up to renaming customers, instances I' and I" are almost identical (See Figure A.1). In fact, at each location ℓ, there are at most $\lfloor n_\ell \epsilon^2 \rfloor$ customers of I' that do not correspond to a customer of I", where n_ℓ is the number of big demands at location ℓ. Using a single tour to cover each of those customers yields

$$\mathrm{OPT}(I') \leq \mathrm{OPT}(I") + \sum_\ell 2 n_\ell \epsilon^2 p_\ell \tag{3}$$

But by Lemma 1 and the fact that big demands are at least ϵW, we get

$$\mathrm{OPT}(I_1) \geq \frac{2}{W} \sum_\ell n_\ell \epsilon W p_\ell = \sum_\ell 2 n_\ell \epsilon p_\ell \tag{4}$$

Combining Equations 3 and 4 yields the lemma.

A.2 Proof of Lemma 3

Let S be the minimum cost solution that uses only small expanse tours. Fix a particular partition P_i and let T_i be the set of tours from S that cover points in more than one region in P_i. Since each tour in $t \in T_i$ has small expanse, t covers points in at most two regions of partition P_i. For each $t \in T_i$, make two copies of t, and assign one copy to cover the points in the first active region and the second copy to cover the points in the second active region of t. After the modifications all tours cover points in only one region. We obtain:

$$\sum_{R \in P_i} \mathrm{OPT}(R) \leq S + \sum_{t \in T_i} \mathrm{length}(t).$$

Summing over all partitions, we obtain:

$$\sum_{0 \leq i < 1/\epsilon} \sum_{R \in P_i} \mathrm{OPT}(R) \leq \sum_{0 \leq i < 1/\epsilon} \left(S + \sum_{t \in T_i} \mathrm{length}(t) \right) \tag{5}$$

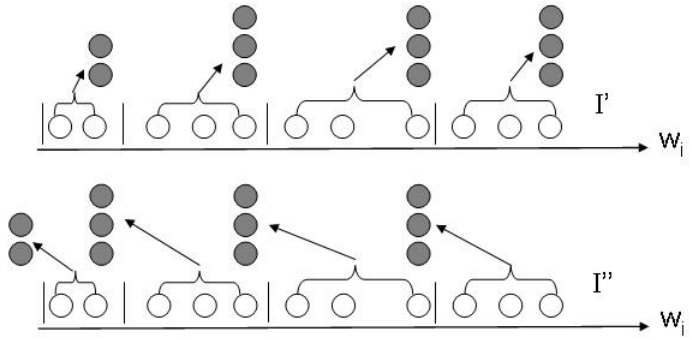

Fig. 1. Rounding the big demands at location p in the proof of Lemma 2. The big demands in instance I (white circles) are sorted and partitioned into groups. The big demands in I' (top gray circles) are obtained by rounding demands up to the maximum in each group. The big demands in I'' (bottom gray circles) are obtained by rounding down each demand to the maximum of the next group.

Note that for $i \neq j$, T_i and T_j are disjoint; a tour $t \in T_i$ spans across two consecutive regions in P_i and thus two consecutive blocks. These consecutive blocks are in the same region in partition P_j, thus $t \notin T_j$. This implies that the right hand side of Equation 5 is at most $(1/\epsilon + 1)S$. Thus we have that,

$$\sum_{0 \leq i < 1/\epsilon} \sum_{R \in P_i} \mathrm{OPT}(R) \leq (1/\epsilon + 1)S$$

As the sum on the left hand side has $1/\epsilon$ terms, there must exist a term i for which $\sum_{R \in P_i} \mathrm{OPT}(R) \leq (1+\epsilon)S$. The proof follows as $S \leq (1+2\epsilon)\mathrm{OPT}$ by Lemma 4.

A.3 Proof of Lemma 4

Start from the optimal solution for I and consider any tour t. Let p_t be the maximum location of the customers covered by t. For every $i \geq 0$, define a new tour t_i that covers the customers covered by t in the interval $(\epsilon^{i+1}p_t, \epsilon^i p_t]$. Replace t by the collection of tours $(t_i)_{i \geq 0}$ (See Figure A.3).

Together, the tours t_i cover exactly the customers initially covered by t, so the new solution is still feasible. By construction, the tours t_i all have small expanse, so the new solution uses only small expanse tours. We have: $\mathrm{OPT}(I) = \sum_t 2p_t$, and the cost of the new solution is at most

$$\sum_t 2(1 + \epsilon + \epsilon^2 + \ldots)p_t < \mathrm{OPT}(I)/(1-\epsilon) \leq (1+2\epsilon)\mathrm{OPT}(I).$$

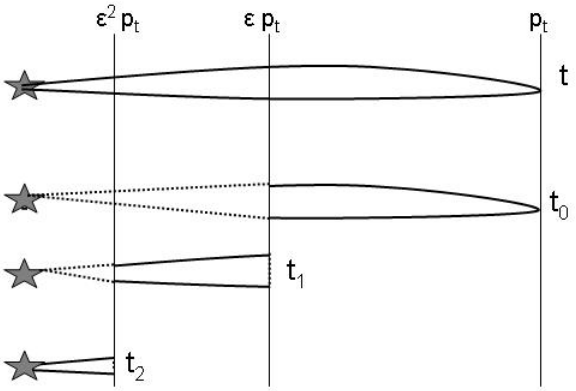

Fig. 2. Lemma 4. The depot is the star. Tour t of length p_t is replaced by t_0, t_1, t_2, by adding the dashed segments from the depot. No points are covered by the dashed segments so that t_i only covers points in $(p_t\epsilon^{i+1}, p_t\epsilon^i]$.

A.4 Proof of Lemma 5

Proof. R spans $(\epsilon^{1/\epsilon}p, p]$ for some p. By Lemma 2 it follows that there are at most $c_{loc} = (1/\epsilon)^2 \log(1/\epsilon)$ locations in R. Moreover each location has only $c_{dem} = 1/\epsilon^2$ distinct values of big demands. Thus there are at most $c_{type} = c_{loc} \cdot c_{dem} = (1/\epsilon)^4 \log(1/\epsilon)$ demand types.

Since big demands have value at least $W\epsilon$, at most $1/\epsilon$ big demands can be taken to make a sum that is $\leq W$. Thus the number of tour profiles is at most $c_p = c_{loc} \cdot \sum_{j \leq 1/\epsilon} (c_{type})^j = (1/\epsilon)^{O(1/\epsilon)}$. There are at most n tours covering big demands. Thus the number of different configurations is n^{c_p}.

A.5 Proof of Lemma 8

Proof. We show how to modify the tours of OPT so that each tour only covers points in the far instance or only covers points in the *close* instance. Let T be the set of tours of OPT which cover points in both instances. For each $t \in T$ cut t at location p_{i_c} to get three pieces: the first piece goes from the depot to location p_{i_c} and covers only points in *close*, the second piece and covers only customers in far and the third piece goes from p_{i_c} back to the depot covering only points in *close*. Concatenate the first and third pieces together at p_{i_c} to get a tour that covers only points in the *close* instance. Let T_2 be the set of second pieces of each tour in T. While there exists at least two pieces in T_2 each covering at most $W/2$ demand, concatenate the pieces together at p_{ci_c} into a new piece covering at most W demand. After all concatenations are done, all but at most one piece in T_2 covers at least $W/2$ demand. Add a single round trip connection from p_{i_c} to depot to each piece in T_2 to get tours covering only points in the far instance.

The total cost to modify T into tours covering only points in far or *close* is the cost of T plus the cost of the additional round trips required to patch up

the pieces in T_2 into tours. Let $\mathrm{dem}(i)$ denote the total demand of customers with indices $\leq i$, i.e $\mathrm{dem}(i) = \sum_{j \leq i} w_j$. Since all but one concatenated piece in T_2 covers at least $W/2$ demand, the number of round trips required is at most $\mathrm{dem}(i_c)/(W/2) + 1 < 3\mathrm{dem}(i_c)/W$. Thus the total cost of additional round trips is at most $2(3\mathrm{dem}(i_c)/W)p_{i_c}$ which implies that

$$\mathrm{OPT}(close) + \mathrm{OPT}(far) \leq \mathrm{OPT} + 6\frac{\mathrm{dem}(i_c)}{W}p_{i_c} \tag{6}$$

By choice of i_c, $\mathrm{dem}(i_c) \leq \epsilon \sum_{j=1}^{n} w_j p_j$. Using Lemma 1 we obtain

$$6\mathrm{dem}(i_c)/W \cdot p_{i_c} \leq 6\epsilon \sum_j w_j p_j / W \leq 3\epsilon \mathrm{OPT},$$

as desired.

A.6 Proof of Lemma 9

Proof. Let i_0 be minimum such that $\sum_{i=1}^{i_0} w_i \geq W/\epsilon$. By definition of i_c, for every $i \in [i_0, i_c)$ we have $(w_1 + \cdots + w_i)p_i > \epsilon \sum_j w_j p_j$. Equivalently:

$$\forall i \in [i_0, i_c), \quad \frac{1}{\epsilon}\frac{w_i p_i}{\sum_j w_j p_j} > \frac{w_i}{w_1 + \cdots + w_i}.$$

Summing over $i \in [i_0, i_c)$ implies

$$\frac{1}{\epsilon} > \sum_{i \in [i_0, i_c)} \frac{w_i}{w_1 + \cdots + w_i}.$$

Go through the sequence $(w_i)_{i_0 \leq i < i_c}$ in order of increasing i, to greedily partition the w_i's into groups $g_1, g_2, \ldots$ such that for every group g, we have $W/\epsilon \leq \sum_{i \in g} w_i < W(1/\epsilon + 1)$. Letting $W_0 = w_1 + \cdots + w_{i_0}$ and, for group g_ℓ, $W_\ell = \sum_{i \in g_\ell} w_i$, we can rewrite the right hand side as

$$\sum_{\ell \geq 1} \sum_{i \in g_\ell} \frac{w_i}{W_0 + \cdots + W_{\ell-1} + \sum_{i' \in g_\ell, i' \leq i} w_{i'}} \geq \sum_{\ell \geq 1} \frac{W_\ell}{W_0 + \cdots + W_\ell}.$$

Since all W_g's are equal to within a $(1+\epsilon)$ factor, this is at least

$$\frac{1}{1+\epsilon} \sum_{\ell \geq 1} \frac{1}{\ell + 1} \geq \frac{1}{2} \log(\#(\text{groups})).$$

Thus the number of groups is at most $\exp(2/\epsilon)$. Since each group has total demand at most $W(1/\epsilon + 1)$, the total demand is $\exp(O(1/\epsilon))W$, as desired.

A.7 A Generalization of the Dynamic Program for Subset-Sum

Let $\{w_1, \ldots w_m\}$ be the demands in interval I_i. The dynamic program populates a table Q. Table element $Q[j, s_1, \ldots s_i]$ specifies whether the demands $w_1 \ldots w_j$ can be partitioned into i sets whose sums are $s_1 \ldots s_i$. The table is populated using the following recurrence: $Q[j, s_1, \ldots s_i]$ is true if any of the following are true: $Q[j-1, s_1 - w_j, s_2, \ldots s_i]$, $Q[j-1, s_1, s_2 - w_j, s_3, \ldots s_i]$, $Q[j-1, s_1, s_2, s_3 - w_j \ldots s_i]$, $\ldots, Q[j-1, s_1, s_2, \ldots, s_{i-1}, s_i - w_j]$. For the base case $Q[1, s_1, \ldots s_i]$ is true if $w_1 = s_j$ for some $j \leq i$ and all the other $s_k = 0$ for all $k \neq j$. Otherwise $Q[1, s_1, \ldots s_i]$ is false. We are interested in the entry $Q[m, s_1, \ldots s_i]$ which specifies whether the partition S^i can be realized or not.

A.8 Proof of Lemma 10

Proof. We can assume that all but at most one tour covers at least $W/2$ demand. Otherwise if there are two tours, each covering at most $< W/2$ demand, they can be merged together at the depot.

A.9 Proof of Lemma 11

Proof. Correctness: Algorithm 6 iterates through all possible configurations and checks the feasibility of each configuration. Fix a configuration. Line 3 verifies that the load of no tour is greater than W. Line 8 verifies that the demand in I_i can be partitioned into $s_1^i, \ldots s_i^i$.
Running Time: We analyze the number of configurations possible for far. As the total demand in far is $We^{O(1/\epsilon)}$, by Lemma 10, $c_{far} = e^{O(1/\epsilon)}$. The number L of locations with customers in the far instance is $We^{O(1/\epsilon)}$. Thus there are $L^{c_{far}} = W^{e^{O(1/\epsilon)}}$ possible right most locations for the c_{far} tours.

For each of the c_{far} intervals, there is a list of at most c_{far} numbers where each number is at most W. Thus there are $W^{c_{far}^2} = W^{e^{O(1/\epsilon)}}$ possible lists $\{s_1^i, \ldots s_i^i\}_i$. Therefore, the total number of configurations $W^{e^{O(1/\epsilon)}}$, which is a polynomial in n when $W \leq poly(n)$.

Next we analyze the time required to verify the feasibility of a configuration. Line 3 takes polynomial time as they involve summing a constant number of values. The extended version of the subset sum DP requires $O(c_{far} \cdot n \cdot W^{c_{far}})$ time since the table Q has size $n \cdot s_1 \cdot s_2 \cdot \ldots \cdot s_i \leq n \cdot W^{c_{far}}$, and each entry can be computed in constant time by looking up at most $c_{far} + 1$ entries. Thus Line 8 takes time $n \cdot W^{e^{O(1/\epsilon)}}$. Therefore, the total running time of Algorithm 6 is $n \cdot W^{e^{O(1/\epsilon)}}$.

A.10 Proof of Lemma 12

Proof. By definition of i_c, $\sum_{j \leq i_c} w_j \geq W/\epsilon$. Using Lemma 1 we have

$$\text{OPT} > 2 \sum_{j \leq i_c} \frac{w_j p_j}{W} \geq 2 \sum_{j \leq i_c} \frac{w_j p_{i_c}}{W} \geq 2 \frac{W}{\epsilon} \frac{p_{i_c}}{W}.$$

A.11 Proof of Theorem 2

Correctness of Algorithm 4. By Lemma 8 OPT(far) plus OPT(*close*) is a near optimal solution of the original instance. It remains to show that Algorithm 4 computes near optimal solutions for both far and *close*. Lemma 11 proves that Algorithm 6 computes the optimal solution of the far instance.

Now we focus on cost of solution for *close*. Using the notation from the proof of Theorem 1, let P^* be the partition of the *near* for which Lemma 3 holds and let $R_1^*, R_2^*, \ldots, R_r^*$ be the regions of P^*. It remains to show that Algorithm 1 finds a near optimal solution for each region of partition P^*. Applying the same argument as in the proof of Theorem 1 we can show that Algorithm 1 finds a solution of cost

$$\sum_{i \leq r} (1+2\epsilon)\mathrm{OPT}(R_i^*) + 2p_{R_i^*} = (1+O(\epsilon))\mathrm{OPT} + 2\sum_{i \geq 0} p_{R_i^*}.$$

The farthest customer in *close* is at location $\leq p_{i_c}$. Thus by definition of regions the right most location of a region R_i^* is $p_{R_i^*} = p_{i_c}\epsilon^{i/\epsilon}$. Thus we have

$$\sum_{i \geq 0} p_{R_i^*} \leq \sum_{i \geq 0} 2p_{i_c}\epsilon^{i/\epsilon} \leq 4p_{i_c} \leq \epsilon\mathrm{OPT},$$

where the last inequality follows by Lemma 12, $p_{i_c} \leq \epsilon\mathrm{OPT}$.

Thus the cost of the solution output by Algorithm 4 is at most

$$\mathrm{OPT}(far) + (1+O(\epsilon))\mathrm{OPT}(close) + \epsilon\mathrm{OPT},$$

which is $(1+O(\epsilon))\mathrm{OPT}$ by Lemma 8.

The running time. By Lemma 11 far can be computed by Algorithm 6 in time $nW^{e^{O(1/\epsilon)}}$. By theorem 1 Algorithm finds a solution for the *close* instance in time $n^{(1/\epsilon)^{O(1/\epsilon)}}$. Thus the running time of Algorithm 4 is $nW^{e^{O(1/\epsilon)}} + n^{(1/\epsilon)^{O(1/\epsilon)}}$.