The Train Delivery Problem - Vehicle Routing Meets Bin Packing

Aparna Das*† Claire Mathieu*† Shay Mozes*

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 problem is also equivalent to the problem of minimizing the makespan on a single batch machine with non-identical job sizes in the scheduling literature.

The train delivery problem is strongly NP-Hard and does not admit an approximation ratio better than 3/2. We design the first 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 consider 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 the scheduling problem of minimizing the makespan on a single batch machine with non-identical job sizes.

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

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

We describe some applications of the different formulations of the train delivery problem. In the scheduling setting, integrated circuits are tested by subjecting them to thermal stress for an extended period of time (burn-in). Each circuit has a prespecified minimum burn-in time (p_i) and a number of boards needed (w_i). Since circuits may stay in the oven for a period longer than their burn-in time, it is possible to place different products as a batch in the oven simultaneously as long as the capacity of the oven (the number of boards in the oven) is not exceeded. The processing time of each batch is the longest minimum exposure time among all the products in the batch. Once processing is begun on a batch, no product can be removed from the oven until the processing of the batch is complete. We wish to find a partition of the circuits into batches so that the total processing time of all batches (the makespan) is minimized.

*Brown University, Providence RI 02918, USA. {aparna,claire,shay}@cs.brown.edu

†Supported by NSF grant CCF-0728816.

In the VRP setting with unsplittable demands, containers are to be transported from a 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 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.

Bin-packing is the special case of the train delivery problem where all the p_i's are equal. It is well known [17] 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).

Our Results. We give the first approximation schemes for the train delivery problem. The 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. In other words, scale the input so that $\max_i p_i = 1$; then we are in the asymptotic regime if the cost of the scaled input is $\Omega(1/\epsilon^6)$.

Theorem 1.1. *For any instance of the train delivery problem such that* $\max_i p_i = O(\epsilon^6)OPT$*, Algorithm 1 outputs a solution of cost* $(1+O(\epsilon))OPT$ *in time* $O(n\log(n)) + \log(n)(1/\epsilon)^{O(1/\epsilon)}$.[1]

Given an instance of the train delivery problem we can use any constant factor approximation to check whether the conditions of Theorem 1.1 hold. For example, we can use the constant factor approximation for the metric VRP with unsplittable demands given by [19].

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. We emphasize that the PTAS applies even when the conditions of Theorem 1.1 do not hold.

Theorem 1.2. *Given an instance of the train delivery problem such that* $W = poly(n)$*, Algorithm 5 outputs a solution of cost* $(1+O(\epsilon))OPT$*. Its running time is* $W^{e^{O(1/\epsilon)}} + O(n\log(n)) + \log(n)(1/\epsilon)^{O(1/\epsilon)}$.

We note that, unless P=NP, we cannot hope to achieve a PTAS when the conditions of Theorem 1.2 do not hold. A PTAS that also works when $W > poly(n)$ would give us a

[1] It is possible to trade off running time with the severity of the asymptotic assumption. An alternative version of our algorithm uses a less severe asymptotic assumption, $\max_i p_i = O(\epsilon)\mathrm{OPT}$, but runs in time $n^{(1/\epsilon)^{O(1/\epsilon)}}$.

polynomial time algorithm, rather than a pseudo polynomial time algorithm, for deciding the NP-hard *partition* problem[2].

Both of our results can be extended to the variant of the one dimensional VRP with unsplittable demands and multiple depots. We omit the details.

Related work. The train delivery problem in its formulation as the problem of minimizing the makespan on a single batch machine with non-identical job sizes has been extensively studied in the past two decades, and the present paper gives the first asymptotic PTAS for this problem. To the best of our knowledge, Uzsoy [31] was the first to consider the problem. He proved that it is NP-Hard and presented a few heuristics that were evaluated empirically. Many others have considered the problem since. Various heuristics are given in [2, 12, 13, 30] to name just a few, while application of meta-heuristics to the problem were studied in [28, 24, 22]. The work of Zhang et al. [32] stands out in its theoretical treatment of the worst case behaviour of several heuristics. They prove constant approximation ratios for some heuristics, with 7/4 being the best, while showing that others may perform arbitrarily bad.

The techniques we use in this paper draw on those used in the literature for the bin-packing problem and the vehicle routing problem. Both problems 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 [17]. Fernandez de la Vega and Lueker [15] obtained the first APTAS. Their algorithm handles big and small demands separately and uses the fact the small demands can be ignored initially and added greedily to any near optimal solution of just the big demands. The big demands are rounded and a linear program is used to find a near optimal solution for them in time $O(C_\epsilon)$, where C_ϵ is exponentially large in $1/\epsilon$, but does not depend on n. Subsequently, Karmarkar and Karp [21] gave an asymptotic *fully* polynomial approximation scheme (AFPTAS) using the same framework and showing how to *efficiently* solve and round the 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 [10] for a survey). In multi-dimensional bin-packing (See [7, 23, 4, 9] 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 [16, 29, 14]) 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 [14, 26]), 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. None of these variants, however, captures the problem we consider. In some of these variants, in contrast to the classical bin-packing problem, the small demands can make an important contribution to the cost of the solution and must be handled more carefully. The results differ substantially on how much consideration is given to small items while computing a solution of the big items. In the case of bin-packing with "general cost structure", the authors of [14] consider small items to be fluid so that they can be split up arbitrarily among bins. (A similar approach was also used in the bin-covering problem of [11]).

The VRP is another widely studied problem. The train delivery problem is the 1-dimensional version of the VRP with unequal or unsplittable demands [19, 6, 5] 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

[2]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.

a single vehicle. The objective is to minimize the total distance travelled by all vehicles[3]. 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 [19]. Bramel et al. give a probabilistic analysis for the Euclidean plane where customer demands are drawn i.i.d from any distribution [6]. Labbé et al. [25] give a 2-approximation for the problem on a tree. Additional heuristics that extend their approach were given in [27, 8].

For the splittable case, where customers may be served by multiple vehicles, Haimovich and Rinnooy Kan gave a PTAS for the Euclidean plane when $W = O(\log \log n)$ [18]. 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 [3, 20].

Main techniques. To achieve Theorem 1.1, we round the positions of all demands geometrically and then apply the bin packing rounding scheme from [15] to the big demands 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 yields a near-optimal solution for the original instance. In each region of the partition, we solve a relaxed problem where we pretend that the small demands are fluid and can be split arbitrarily among different tours, as is done in [14, 11]. We find a near optimal solution for the relaxed problem and extend it greedily into a feasible solution for unsplittable small demands, as was done in bin packing algorithms. 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 relaxed solution we compute. See Section 2 for details.

To achieve Theorem 1.2, 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 [18]. Our far instance is small enough to solve it exactly by dynamic programming, and our $close$ instance is solved by Theorem 1.1. The crux of the analysis is a structural lemma which is an extension of [18] 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 our input is preprocessed as defined below:

Definition 1.3. *(Preprocessed) An instance is preprocessed if:*
- *No demand is located closer than $\epsilon \cdot p_{\max}/n$ from the depot, where $p_{\max} = \max_i p_i$.*
- *All customers are located to the right of the depot.*

If there are any demands located closer than $\epsilon \cdot p_{\max}/n$ from the depot, we can serve them each with a separate tour. This will have overall cost at most $2\epsilon p_{\max}$, which is at most ϵOPT given that any solution must have cost at least $2p_{\max}$. Finally if there are customers on the right

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

and left of the depot, we can solve each side separately, as they are analogous to one another, and return the union of the two solutions. We will use the following lower bound from [19].

Lemma 1.4. *[19](Rad Lower Bound.) For any instance I of the train delivery problem, OPT(I) has cost at least Rad(I) $= \frac{2}{W}\sum_{i\in I} p_i \cdot w_i$.*

2 Algorithm for Theorem 1.1

The algorithm is given in Algorithm 1. We present high level description first and details in subsections.

Algorithm 1 Asymptotic PTAS for train delivery

Input: A preprocessed input with demands $(p_i, w_i)_{1\le i\le n}$ and train capacity W
Precondition: $\max_i p_i \le \epsilon\mathrm{OPT}$

1: Round the input using Algorithm 2.
2: **for** $1 \le i \le 1/\epsilon$ **do**
3: Let P_i be the i-th partition into regions (as in Definition 2.4).
4: **for** each non-empty region R of P_i **do**
5: Get a *relaxed* solution covering all demands in R treating small demands as fluid using Algorithm 3.
6: Extend the *relaxed* solution to cover small demands feasibly using Algorithm 4.
7: Let Best(P_i) be the union of the solutions found for each region $R \in P_i$.

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 demands into disjoint regions based on their distance from the depot (Definition 2.4) so that each region has only a constant number of locations containing demands. 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. We relax the unsplittable constraint for small demands and think of them as fluid, allowing each small demand to be split up between multiple tours. Using a linear program we solve the relaxed problem considering all the big demands and the total volume of small demand fluid at each location in the region.

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 us to obtain a solution for the relaxed problem in constant time.

We construct a feasible solution from the relaxed solution by adding the small demands greedily. We prove that the solution we output has cost at most $(1+2\epsilon)\mathrm{OPT}(R)+O((1/\epsilon)^5)p_R$, where $\mathrm{OPT}(R)$ denotes the optimal solution of region R and p_R is the location furthest from the depot in R (Lemma 2.10).

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 demand in the instance. Our assumption $p_{\max} \le O(\epsilon^6)\mathrm{OPT}$ ensures

that the additive cost incurred by the greedy extension (the p_R terms) is within the desired approximation factor.

2.1 Rounding

Rounding is performed using Algorithm 2.

Algorithm 2 Rounding Instance

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

1: Round each p_i up to the smallest integer power of $(1+\epsilon)$.
2: Partition demands $(w_i)_i$ into *big* ($\ge W\epsilon$) and *small* ($< W\epsilon$).
Rounding big demands:
3: **for** each location ℓ s.t. n_ℓ, the number of big demands at ℓ, is at least $1/\epsilon^2$ **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.

Output: rounded instance I' of the train delivery problem

The analysis relies on the following lemma whose proof (in the appendix) is an extension of the bin packing rounding analysis by Fernandez de la Vega and Lueker [15].

Lemma 2.1. *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 (non-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.*
- *$OPT(I') \le (1 + O(\epsilon))\, OPT(I)$.*

2.2 Partitioning into regions

We first define a simple structure for a tour, namely we say a tour has small expanse if it covers points in a bounded region. We show that a near optimal solution can be obtained using only tours with this simple structure.

Definition 2.2. *A tour that covers only points between locations p and p', $p \le p'$, has* expanse p'/p. *A* small *expanse tour has expanse at most $1/\epsilon$.*

Lemma 2.3. *Let I' be an instance of the train delivery problem. There exists a solution using only small expanse tours which costs at most $(1+2\epsilon)\, OPT(I)$.*

By Lemma 2.3 (proved in the appendix), we can concentrate on finding the optimal small expanse solution.

We partition instance I' into regions, where each region has large expanse. We find an optimal solution for each region independently and output the union of these solutions. Intuitively, as the expanse of the region is large only a few tours of the optimal small expanse solution will cover points in more than one region.

Definition 2.4. *Let I' be a rounded instance of the train delivery problem and $p_{\max} = \max_i p_i$. A* block*, defined by an integer i, is the set of demands 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 $i \geq 0$.

Note that there are $1/\epsilon$ possible ways to partition I' into regions. We use a *shifting* technique similar to that of Baker [3] and Hochbaum and Maass [20] and an averaging argument to show that at least one partition yields a near optimal solution for I'. See appendix.

Lemma 2.5. *There exists a partition P_j s.t. $\sum_{R \in P_j} OPT(R) \leq (1 + O(\epsilon))OPT$.*

2.3 Solving the relaxed problem in a region

Within a region, the big demands and tours of the region can be described concisely.

Definition 2.6. *(Big demand type) Fix a region R of the rounded instance I'. A big demand* type *is a pair (p, b) where p is the location of a big demand and b is one of the at most $1/\epsilon^2$ big demand (rounded) values at location p. Let $n(d)$ denote the total number of demands of type d in region R.*

The configuration of a tour roughly describes which points it will cover: for each occurrence of demand type (p, b) in its multiset the tour covers one of the demands from location p with value b.

Definition 2.7. *(Configuration) Fix a region R of rounded instance I'. A* configuration *f of a tour in R consists of a location r_f, which is the furthest location of the tour, and a multiset M_f of demand types, each with location at most r_f, whose values sum up to at most W.*

Let c_f denote the remaining capacity of a tour with configuration f (i.e., $c_f = W - \sum_{(p,b) \in M_f} b$). For any big demand type d, let $n_f(d)$ denote the multiplicity of d in M_f. Let S be the set of small demands in a region R. We relax our problem by pretending that the small demands can be split up among multiple tours. Then we use the following linear program to solve the relaxed problem. The linear program has one variable x_f for each possible tour configuration f. The goal of the linear program is to select a minimum cost multiset of tour configurations such that two constraints are satisfied: Constraint 1 ensures that all big demand types are covered by the selected tour configurations and constraint 2 ensures that for each location p, the small demands further right than p can be covered with by the remaining capacities of the tour configurations that extend to the right of p.

$$\min \sum_{f \in \mathcal{F}} r_f x_f$$

$$s.t \sum_{f \in \mathcal{F}} x_f n_f(d) \geq n(d) \qquad \text{for all demand types } d \qquad (1)$$

$$\sum_{f : r_f \geq p} c_f x_f \geq \sum_{(p_i, w_i) \in S, p_i \geq p} w_i \qquad \text{for all locations } p \qquad (2)$$

$$x_f \geq 0$$

Lemma 2.8. *Algorithm 3 outputs a solution in time $(1/\epsilon)^{O(1/\epsilon)}$.*

Algorithm 3 Solve Relaxed Region

Input: R a region of I', S the set of small demands in R.

1: Let $\mathcal{D}$ be the set of big demand types for region R (Definition 2.6).
2: Let $\mathcal{F}$ be the set of tour configurations for region R (Definition 2.7).
3: Let $x^* = (x^*_f)_{f\in\mathcal{F}}$ denote a basic optimal solution of the linear program of Equations (1-2).
4: Let $\bar{x}_f = \lceil x^*_f \rceil$ for each $f \in \mathcal{F}$.
5: Cover the big demand types in $\mathcal{D}$ with tours specified by the $(\bar{x}_f)_{f\in\mathcal{F}}$. (Some tours may only be assigned a partial list of the big demands listed in its configuration.)

Output: The resulting set of tours covering $\mathcal{D}$.

Algorithm 3 outputs a solution in constant time (Lemma 2.8, see appendix for proof). It rounds the solution of the above linear program to obtain a solution to the relaxed problem. The solution returned by Algorithm 3 covers all big demands types in region R and is a near optimal solution for the relaxed problem in region R. The quality of the solution is guaranteed by the following lemma whose proof is in the appendix.

Lemma 2.9. *Let p_R denote the maximum location in region R. Let T be the set of tours output by Algorithm 3. T covers all big demands in region R and T covers all small demands if each small demand is allowed to be covered by multiple tours. Let $OPT(R)$ be the cost of the optimal solution to the (unrelaxed) problem in region R. We have that,*

$$Cost(T) \leq OPT(R) + p_R((1/\epsilon)^2 \log(1/\epsilon))(2 + 1/\epsilon^2)$$

2.4 Extending a relaxed solution of a region

Let $(r_t, c_t)_t$ denote the set of tours output by Algorithm 3 where with r_t denoting the maximum location and c_t denoting the remaining capacity of tour t after it has covered the big demands. Algorithm 4 takes the list of tours $(r_t, c_t)_t$ as input and extends the solution to cover the small demands of R in a feasible way (i.e., without splitting any of them).

Algorithm 4 Greedy Extension

Input: small demands $(p_i, w_i)_i$ and a list T of tours $(r_t, c_t)_t$. r_t is the furthest location of tour t and c_t is its remaining capacity.

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 $c_t := c_t - w_i$

Output: the resulting tours.

Lemma 2.10 shows that the greedy extension is a near optimal solution. See appendix.

Lemma 2.10. *Let G be the output of Algorithm 4 on input $(p_i, w_i)_i = T$. Then $cost(G) \leq (1 + 2\epsilon)cost(T) + 2p_R$.*

2.5 Proof of Main Theorem 1.1

Correctness of Algorithm 1. By Lemma 2.1 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 2.5 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 invokes Algorithm 3 to produce a set of tours T that cover all big demands in region R_i^* and all the small demands if each small demand is treated as fluid (i.e is allowed to be split up among multiple tours).

Given T, Algorithm 4 produces a solution that covers small demands feasibly (without splitting) and by Lemma 2.10 the cost of the resulting solution for R_i^* is at most $(1+2\epsilon)\text{cost}(T)+2p_{R_i^*}$, where $p_{R_i^*}$ is the maximum location in R_i^*. By Lemma 2.9, the cost of region R_i^* is at most $(1+2\epsilon)\text{OPT}(R_i^*)+\ell(\epsilon)p_{R_i^*}$, where $\ell(\epsilon) = (1+2\epsilon)(1/\epsilon)^2\log(1/\epsilon)(2+1/\epsilon^2)+2 = O((1/\epsilon)^5)$.

Applying the above argument to all regions in P^*, and using Lemma 2.5, Algorithm 1 outputs a solution of cost

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

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

$$\ell(\epsilon)\sum_{i\ge 0} p_{R_i^*} \le \ell(\epsilon)\sum_{i\ge 0} p_{\max}\epsilon^{i/\epsilon} \le 2\ell(\epsilon)p_{\max} \le O(\epsilon)\text{OPT},$$

where the last inequality follows by our assumption that $\max_i p_i \le O(\epsilon^6)\text{OPT}$.

3 Algorithm for Theorem 1.2

The algorithm is given in Algorithm 5. We present high level description first and details in subsections.

Algorithm 5 PTAS for the train delivery problem when $W \le poly(n)$

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

1: Partition the instance into *close* and *far* using Algorithm 6
2: Find OPT(far) using Algorithm 7.
3: Find Best(*close*) using Algorithm 1

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

Partition into *close* and far instances. We index the demands in decreasing order of their location from the depot and identify a demand i_c. The instance is partitioned into, *close* and far where far contains the demands with indices at most i_c and *close* contains the demands 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 3.3). 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.

Approximate solution of *close*. We use Algorithm 1 to find a near optimal solution to *close*. The cost of the solution is at most $(1+\epsilon)\text{OPT}(close) + O((1/\epsilon)^5)p_{i_c}$.

Overall solution. The solution is the union of the solutions for *close* and far. It is crucial that, on the one hand, far does not contain too much demand, so it can be solved efficiently. On the other hand, far contains enough demand so that the condition of Theorem 1.1 holds for *close*, namely that $p_{i_c} = O(\epsilon^6)\mathrm{OPT}$, where p_{i_c} denotes the furthest location in *close*.

3.1 Partitioning into *close* and far

Algorithm 6 Partition Close and Far

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

1: Let i_c be the smallest index such that
- $\sum_{j \le i_c} w_j \ge W/\epsilon^6$
- $p_{i_c} \sum_{j \le i_c} w_j \le \epsilon \sum_{j=1}^{n} w_j p_j$.

If no such i_c exist, set $i_c := n$.

2: ***Far:*** Let the far instance consist of the demands indexed by $1, \ldots, i_c$.

3: ***Close:*** Let the *close* consist of the remaining demands $i_c + 1, \ldots n$.

Output: instances far and *close*

By Lemma 3.1 the optimal can be transformed into separate solutions for *close* and far at small additional cost (see appendix).

Lemma 3.1. *Given an instance I of the train delivery problem, Algorithm 6 returns two instances* far *and* close *s.t. $OPT(close) + OPT(far) \le (1 + O(\epsilon))OPT$.*

3.2 Solving the far instance

To solve the far instance, we show that the total demand in far is $O(W)$. The following combinatorial lemma is an extension of Haimovich and Rinnooy Kan's [18] analysis to the case of unsplittable demands. It bounds the total demand in the far instance by bounding the number of demands that violate the requirement $p_{i_c} \sum_{j \le i_c} w_j \le \epsilon \sum_{j=1}^{n} w_j p_j$, see the appendix.

Lemma 3.2. *Let i_c be as in Algorithm 6. Then $\sum_{j \le i_c} w_j = \exp(O(1/\epsilon))W$.*

This implies that $\mathrm{OPT}(far)$ uses a constant, c_{far}, number of tours. We show how to solve such an instance optimally using dynamic programming.

Lemma 3.3. *Let I be an instance of the problem such that the sum of all the demands in I is D. Then OPT uses at most $\lceil 2D/W \rceil$ tours.*

Definition 3.4. *(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 $r_1 \ge r_2 \ldots \ge r_{c_{far}}$ s.t. r_i is the maximum location of tour i.*
- *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 $\sum_{j \le c_{far}} 2r_j$.*

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 demands 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 $W \le poly(n)$)) using a trivial extension of the dynamic program for the subset sum problem (see appendix).

Algorithm 7 solves the far instance by iterating all configurations of $\mathrm{OPT}(far)$, checking feasibility, and returning the minimum cost feasible configuration.

Algorithm 7 Solving the far instance

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

1: **for** each configuration f of far as given in definition 3.4 **do**
2: **for** each tour $j \leq c_{far}$ **do**
3: **if** $\sum_{i \leq c_{far}} s_j^i > W$ **then**
4: Mark f infeasible. {capacity of tour is exceeded}
5: **for** each interval $i \leq c_{far}$, with total demand $dem(I_i)$ **do**
6: **if** Extended DP of subset sum cannot partition $dem(I_i)$ into $s_1^i, \ldots, s_i^i$ **then**
7: Mark f infeasible. {demands cannot be partitioned}

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

Lemma 3.5. *Given an instance of far with demand $We^{O(1/\epsilon)}$ Algorithm 7 finds the optimal solution of far in time $nW^{e^{O(1/\epsilon)}}$, which is polynomial in n and W.*

3.3 Proof of Main Theorem 1.2

Theorem 1.2 is proved using Lemma 3.6, which follows by the choice of p_{i_c}. See appendix.

Lemma 3.6. $OPT > 2p_{i_c}/\epsilon^6$.

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] Meral Azizoglu and Scott Webster. Scheduling a batch processing machine with non-identical job sizes. *International Journal of Production Research*, 38:2173–2184, June 2000.

[3] B. S. Baker. Approximation algorithms for np-complete problems on planar graphs. *J. ACM*, 41(1):153–180, 1994.

[4] 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.

[5] 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.

[6] J. Bramel, Jr. Coffman, Edward G., P. W. Shor, and D. Simchi-Levi. Probabilistic analysis of the capacitated vehicle routing problem with unsplit demands. *Oper. Res.*, 40:1095–1106, November 1992.

[7] A. Caprara. Packing 2-dimensional bins in harmony. In *FOCS*, pages 490–499, Washington, DC, USA, 2002. IEEE Computer Society.

[8] Bala Chandran and S. Raghavan. *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces Series*, chapter Modeling and Solving the Capacitated Vehicle Routing Problem on Trees, pages 239–261. Springer US, 2008.

[9] 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.

[10] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson. *Approximation algorithms for bin packing: a survey*, pages 46–93. PWS Publishing Co., Boston, MA, USA, 1997.

[11] Janos Csirik, David S. Johnson, and Claire Kenyon. Better approximation algorithms for bin covering. In *SODA '01: Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 557–566, Philadelphia, PA, USA, 2001. Society for Industrial and Applied Mathematics.

[12] L. Dupont and F. Jolai Ghazvini. Minimizing makespan on a single batch processing machine with non-identical job sizes. *European Journal of Automation Systems*, 32:431–440, 1998.

[13] Lionel Dupont and Clarisse Dhaenens-Flipo. Minimizing the makespan on a batch machine with non-identical job sizes: an exact procedure. *Computers and Operations Research*, 29(7):807 – 819, 2002.

[14] L. Epstein and A. Levin. An aptas for generalized cost variable-sized bin packing. *SIAM J. Comput.*, 38:411–428, April 2008.

[15] 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.

[16] D K Friesen and M A Langston. Variable sized bin packing. *SIAM J. Comput.*, 15(1):222–230, 1986.

[17] M. R. Garey and D. S. Johnson. " strong " np-completeness results: Motivation, examples, and implications. *J. ACM*, 25(3):499–508, 1978.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] A. Husseinzadeh Kashan, B. Karimi, and F. Jolai. Effective hybrid genetic algorithm for minimizing makespan on a single-batch-processing machine with non-identical job sizes. *International Journal of Production Research*, 44:2337–2360, 2006.

[23] C. Kenyon and E. Rémila. A near-optimal solution to a two-dimensional cutting stock problem. *Math. Oper. Res.*, 25(4):645–656, 2000.

[24] S.-G. Koh, P.-H. Koo, D.-C. Kim, and W.-S. Hur. Scheduling a single batch processing machine with arbitrary job sizes and incompatible job families. *International Journal of Production Economics*, 98:81–96, 2005.

[25] M. Labbé, G. Laporte, and H. Mercure. Capacitated vehicle routing on trees. *Operations Research*, 39(4):616–622, 1991.

[26] C.L. Li and Z.L Chen. Bin-packing problem with concave costs of bin utilization. *Naval Research Logistics*, 53(4):298–308, 2006.

[27] P. Mbaraga, A. Langevin, and G. Laporte. Two exact algorithms for the vehicle routing problem on trees. *Naval Research Logistics*, 46:75–89, 1999.

[28] S. Melouk, P. Damodaran, and P.-Y. Chang. Minimizing makespan for single machine batch processing with non-identical job sizes using simulated annealing. *International Journal of Production Economics*, pages 141–147, 2004.

[29] F.D. Murgolo. An efficient approximation scheme for variable-sized bin packing. *SIAM J. Comput.*, 16(1):149–161, 1987.

[30] N. Rafiee Parsa, B. Karimi, and A. Husseinzadeh Kashan. A branch and price algorithm to minimize makespan on a single batch processing machine with non-identical job sizes. *Computers and Operations Research*, 37(10):1720 – 1730, 2010.

[31] R. Uzsoy. Scheduling a single batch processing machine with non-identical job sizes. *International Journal of Production Research*, 32:1615–1635, July 1994.

[32] G. Zhang, X. Cai, C.-Y Lee, and C.K Wong. Minimizing makespan on a single batch processing machine with nonidentical job sizes. *Naval Research Logistics*, 48, 2001.

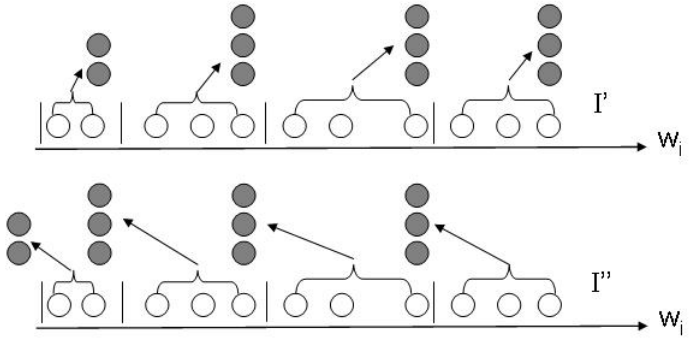

Figure 1: Rounding the big demands at location p in the proof of Lemma 2.1. The big demands in instance I (white circles) are sorted and partitioned into groups. The big demands in I' (top gray circles) are obtained by rounding demands up to the maximum in each group. The big demands in I'' (bottom gray circles) are obtained by rounding down each demand to the maximum of the next group.

A Appendix

A.1 Proof of Lemma 2.1

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

$$\mathrm{OPT}(I_1) \leq (1+\epsilon)\mathrm{OPT}(I). \tag{3}$$

Next we analyze rounding demands, by carrying out the de la Vega and Lueker bin packing analysis at each location [15]. 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, $\mathrm{OPT}(I") \leq \mathrm{OPT}(I_1)$, and so

$$\mathrm{OPT}(I') \leq \mathrm{OPT}(I_1) + (\mathrm{OPT}(I') - \mathrm{OPT}(I")). \tag{4}$$

Note that up to renaming demands, instances I' and I" are almost identical (See Figure A.1). In fact, at each location ℓ, there are at most $\lfloor n_\ell \epsilon^2 \rfloor$ demands of I' that do not correspond to a demand of I", where n_ℓ is the number of big demands at location ℓ. Using a single tour to cover each of those demands yields

$$\mathrm{OPT}(I') \leq \mathrm{OPT}(I") + \sum_\ell 2 n_\ell \epsilon^2 p_\ell \tag{5}$$

Also, by Lemma 1.4 and the fact that big demands are at least ϵW, we get

$$\mathrm{OPT}(I_1) \geq \frac{2}{W} \sum_\ell n_\ell \epsilon W p_\ell = \sum_\ell 2 n_\ell \epsilon p_\ell \tag{6}$$

so $\mathrm{OPT}(I') - \mathrm{OPT}(I'') \leq \epsilon \mathrm{OPT}(I_1)$. Substituting that into 4 and using 3 shows the lemma. □

A.2 Proof of Lemma 2.3

Start from the optimal solution for I and consider any tour t. Let p_t be the maximum location of the demands covered by t. For every $i \geq 0$, define a new tour t_i that covers the demands 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 2).

Together, the tours t_i cover exactly the demands 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: $\mathrm{OPT}(I) = \sum_t 2 p_t$, and the cost of the new solution is at most

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

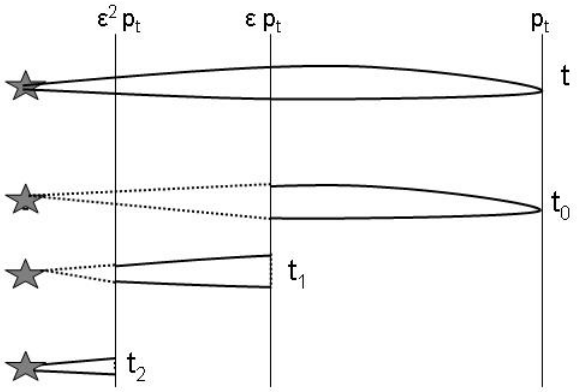

Figure 2: Lemma 2.3. The depot is the star. Tour t of length p_t is replaced by t_0, t_1, t_2, by adding the dashed segments from the depot. No points are covered by the dashed segments so that t_i only covers points in $(p_t\epsilon^{i+1}, p_t\epsilon^i]$.

A.3 Proof of Lemma 2.5

Let S be the minimum cost solution that uses only small expanse tours. Fix a particular partition P_j and let T_j be the set of tours from S that cover points in more than one region in P_j. Since each tour in $t \in T_j$ has small expanse, t covers points in at most two regions of partition P_j. For each $t \in T_j$, 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:

$$\sum_{R \in P_j} \text{OPT}(R) \leq S + \sum_{t \in T_j} \text{length}(t).$$

Summing over all partitions, we obtain:

$$\sum_{0 \leq j < 1/\epsilon} \sum_{R \in P_j} \text{OPT}(R) \leq \sum_{0 \leq j < 1/\epsilon} \left(S + \sum_{t \in T_j} \text{length}(t) \right) \tag{7}$$

Note that for $j \neq i$, T_i and T_j are disjoint; a tour $t \in T_j$ spans across two consecutive regions in P_j and thus two consecutive blocks. These consecutive blocks are in the same region in partition P_i, thus $t \notin T_i$. This implies that the right hand side of Equation 7 is at most $(1/\epsilon + 1)S$. Thus we have that,

$$\sum_{0 \leq j < 1/\epsilon} \sum_{R \in P_j} \text{OPT}(R) \leq (1/\epsilon + 1)S$$

As the sum on the left hand side has $1/\epsilon$ terms, there must exist a term i for which $\sum_{R \in P_j} \text{OPT}(R) \leq (1+\epsilon)S$. The proof follows as $S \leq (1+2\epsilon)\text{OPT}$ by Lemma 2.3.

A.4 Proof of Lemma 2.8

Proof. The bottleneck to the running time of Algorithm 3 is the time required to solve the linear program. This can be done in polynomial time in the number of variables and constraints. The linear program has one variable for each tour configuration and a constraint for each demand type and each location in region R. Below we show that there are $(1/\epsilon)^{O(1/\epsilon)}$ tour configurations, $(1/\epsilon)^2 \log(1/\epsilon)(1 + 1/\epsilon^2)$ demand types and locations $(1/\epsilon)^2 \log(1/\epsilon)$. This implies that the running time of Algorithm 3 is $poly((1/\epsilon)^{O(1/\epsilon)}) = (1/\epsilon)^{O(1/\epsilon)}$.

R spans $(\epsilon^{1/\epsilon}p, p]$ for some p. By rounding locations as in Lemma 2.1 it follows that there are at most $c_{loc} = (1/\epsilon)^2 \log(1/\epsilon)$ locations in R.[4] Moreover each location has only $1/\epsilon^2 + 1$ distinct values of big demands. Thus the number of demand types is at most $c_{type} = c_{loc} \cdot (1+1/\epsilon^2) = (1/\epsilon)^2 \log(1/\epsilon)(1+1/\epsilon^2)$.

[4]The number of locations can be less than c_{loc} since we don't need to count locations with no demands.

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 configurations is at most $c_{loc} \cdot \sum_{j \leq 1/\epsilon} (c_{type})^j = (1/\epsilon)^{O(1/\epsilon)}$. □

A.5 Proof of Lemma 2.9

Proof. Note that $(\bar{x}_f)_{f \in \mathcal{F}}$ satisfies all constraints of the linear program (Equations 1-2). Thus the tours associated with $(\bar{x}_f)_{f \in \mathcal{F}}$ cover all big demand types (constraint 1) and all fluid small demands by (constrain 2) in R.

The proof of Lemma 2.8 shows that the linear program has $c = c_{loc} + c_{type} = ((1/\epsilon)^2 \log(1/\epsilon))(2 + 1/\epsilon^2)$ constraints other than the non-negativity constraints. Thus a basic optimal solution x^* has at most c fractional coordinates. Thus $\sum_{f \in \mathcal{F}} r_f \bar{x}_f \leq (\sum_{f \in \mathcal{F}} r_f x^*_f) + c \cdot p_R \leq \mathrm{OPT}(R) + c \cdot p_R$. The last inequality uses the fact that the optimal solution to the relaxed problem is at most $\mathrm{OPT}(R)$. □

A.6 Proof of Lemma 2.10

Proof. Consider the new tours added by algorithm 4. 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:

$$\mathrm{cost}(G) = \mathrm{cost}(T) + \sum_{s \geq 1} 2(x_s - x_{s-1})|A_s|. \tag{8}$$

Let $\sigma_s = \sum_{i : p_i \geq x_s} w_i$ denote the total small demand at locations at least x_s. Let $T_s = \{t : r_t \geq x_s\}$ denote the set of tours of T that go beyond location x_s i.e., $r_t \geq x_s$. Let α_s denote the total available capacity of all tours of T_s (i.e., $\alpha_s = \sum_{t \in T_s} c_t$). 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 $\alpha_s - |T_s|\epsilon W$, and so the amount of small demand assigned by G to new tours is at most $\sigma_s - \alpha_s + |T_s|\epsilon W$. Since σ_s is the total small demand located to the right of location s and α_s is the amount of remaining capacity on the tours T_s, constraint (2) of the linear program implies that $\sigma_s - \alpha_s \leq 0$ for all s. Thus the total amount of demand assigned by G to new tours is at most $|T_s|\epsilon W$.

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:

$$|A_s| \leq \frac{|T_s|\epsilon W}{(1-\epsilon)W} + 1 = \frac{\epsilon |T_s|}{(1-\epsilon)} + 1. \tag{9}$$

Substituting (9) into (8) we get that $\mathrm{cost}(G)$ is at most:

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

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

$$\mathrm{cost}(G) \leq \frac{1}{1-\epsilon} \mathrm{cost}(T) + 2p_R.$$

For $\epsilon < 1/2$ we get, $\mathrm{cost}(G) \leq (1 + 2\epsilon)\mathrm{cost}(T) + 2p_R$. □

A.7 Running time of Algorithm 1

Rounding locations and the big demands at each location, and partitioning the instance into regions can all be done in time $O(n \log(n))$. Each partition P_i consists of $O(\epsilon \log(n)/\log(\epsilon))$ regions. In each region we solve the relaxed problem and compute the greedy extension. By Lemma 2.8 Algorithm 3 solves the relaxed problem in time $(1/\epsilon)^{O(1/\epsilon)}$ and by inspection one can see that the greedy extension can be computed in at most $O(n)$. As Algorithm 1 computes a solution for each of the $1/\epsilon$ possible partitions, the final run time of Algorithm 1 is $O(n \log(n)) + \log(n)(1/\epsilon)^{O(1/\epsilon)}$.

A.8 Proof of Lemma 3.1

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 goes further than p_{i_c} and covers demands only 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_{i_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 with indices $\le i$, i.e $\mathrm{dem}(i) = \sum_{j \le i} w_j$. 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

$$\mathrm{OPT}(close) + \mathrm{OPT}(far) \le \mathrm{OPT} + 6\frac{\mathrm{dem}(i_c)}{W}p_{i_c} \tag{10}$$

By choice of i_c, $\mathrm{dem}(i_c)p_{i_c} \le \epsilon \sum_{j=1}^{n} w_j p_j$. Using Lemma 1.4 we obtain

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

as desired. □

A.9 Proof of Lemma 3.2

Proof. Let i_0 be minimum such that $\sum_{i=1}^{i_0} w_i \ge W/\epsilon^6$. 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 $(w_i)_{i_0 \le i < i_c}$ 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 except, perhaps, the last one we have $W/\epsilon \le \sum_{i \in g} w_i < W(1/\epsilon + 1)$. Letting $W_0 = w_1 + \cdots + w_{i_0 - 1}$ and, for group g_ℓ, $W_\ell = \sum_{i \in g_\ell} w_i$, we can rewrite the right hand side as

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

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

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

Thus the number of groups is at most $\exp(2/\epsilon)$ (plus possibly one more to account for the last group). Since each group has total demand at most $W(1/\epsilon + 1)$, the total demand is $\exp(O(1/\epsilon))W$, as desired. □

A.10 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.11 Proof of Lemma 3.3

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.12 Proof of Lemma 3.5

Proof. Correctness: Algorithm 7 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 6 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 3.3, $c_{far} = e^{O(1/\epsilon)}$. The number L of locations with demands 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 6 takes time $n \cdot W^{e^{O(1/\epsilon)}}$. Therefore, the total running time of Algorithm 7 is $n \cdot W^{e^{O(1/\epsilon)}}$. □

A.13 Proof of Lemma 3.6

Proof. By definition of i_c, $\sum_{j \leq i_c} w_j \geq W/\epsilon^6$. Using Lemma 1.4 we have

$$\text{OPT} \geq 2 \sum_{j \leq i_c} \frac{w_j p_j}{W} \geq 2 \sum_{j \leq i_c} \frac{w_j p_{i_c}}{W} \geq 2 \cdot \frac{W}{\epsilon^6} \frac{p_{i_c}}{W}.$$

□

A.14 Proof of Theorem 1.2

Correctness of Algorithm 5. By Lemma 3.1 OPT(far) plus OPT($close$) is a near optimal solution of the original instance. It remains to show that Algorithm 5 computes near optimal solutions for both far and $close$. Lemma 3.5 proves that Algorithm 7 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 1.1, let P^* be the partition of the $near$ for which Lemma 2.5 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 1.1 we can show that Algorithm 1 finds a solution of cost

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

where $\ell(\epsilon) = O(1/\epsilon^5)$ as defined in the proof of Theorem 1.1.

The farthest demand 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

$$\sum_{i\geq 0} p_{R_i^*} \leq \sum_{i\geq 0} p_{i_c}\epsilon^{i/\epsilon} \leq 2p_{i_c} \leq \epsilon^6 \mathrm{OPT},$$

where the last inequality follows by Lemma 3.6, $p_{i_c} \leq \epsilon^6 \mathrm{OPT}$.

Thus the cost of the solution output by Algorithm 5 is at most

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

which is $(1 + O(\epsilon))\mathrm{OPT}$ by Lemma 3.1.

The running time. By Lemma 3.5 far can be computed by Algorithm 7 in time $nW^{e^{O(1/\epsilon)}}$. By Theorem 1.1 Algorithm 1 finds a solution for the *close* instance in time $O(n \log n/\epsilon)+O(\log n\cdot(1/\epsilon)^{O(1/\epsilon)})$. Thus the running time of Algorithm 5 is $nW^{e^{O(1/\epsilon)}} + O(n \log n/\epsilon) + O(\log n \cdot (1/\epsilon)^{O(1/\epsilon)})$.