The Train Delivery Problem - Vehicle Routing Meets Bin Packing

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

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

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.
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 general 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.

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.

In the train delivery problem we are given a positive integer capacity W and a set S of n items, each with a positive position p_i and a positive integer weight w_i. We seek a partition of S into subsets $\{S_j\}$ (train tours) that minimizes

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

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

* Supported by NSF grant CCF-0728816
** Supported by NSF grant CCF-0635089

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.

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. 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 for 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. 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. 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 into ships that minimizes the overall cost of operation.

Bin-packing is the special case of the train delivery problem where all the p_i's are equal. It is well known [18] that bin-packing is strongly NP-hard and does not admit a polynomial time approximation algorithm 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 $1/\epsilon$ (i.e. 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 general train delivery problem. The problem does not admit an asymptotic approximation scheme in the usual sense as the cost of the solution is determined by the positions p_i. Thus any instance can be scaled so that the optimal solution has arbitrarily large cost without changing the solution itself. 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. *Given an 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]

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 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[2].

Theorem 2. *Given an instance of the train delivery problem such that* $W = poly(n)$*, Algorithm 5 outputs a solution of cost* $(1+O(\epsilon))OPT$ *in time* $W^{e^{O(1/\epsilon)}} + O(n\log(n)) + \log(n)(1/\epsilon)^{O(1/\epsilon)}$.

Related work. Train delivery 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 the general version of the problem. To the best of our knowledge, Uzsoy [32] 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, 13, 12, 31] to name just a few, while application of meta-heuristics to the problem were studied in [29, 25, 23]. The work of Zhang et al. [33] proves approximation ratios for some heuristics, with 7/4 being the best, while showing that others may perform arbitrarily bad. Zhang and Cao [34] also gave an APTAS for the symmetric setting where they assumed that $p_i = w_i$ for all i.

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. Rather than a comprehensive survey, we focus on the previous techniques we extend in this paper. Bin-packing is one of the problems originally shown to be strongly NP hard by Garey and Johnson [18]. Fernandez de la Vega and Lueker [16] obtained the first APTAS. Their algorithm handles big and small demands separately and uses the fact that 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 polynomial in n and exponential in $1/\epsilon$. Subsequently, Karmarkar and Karp [22] gave an asymptotic *fully* polynomial approximation scheme (AFPTAS) using the same framework and *efficiently* solving and rounding the LP relaxation of the problem. Their running time depends polynomially on $1/\epsilon$, rather than exponentially.

[1] An alternative version of our algorithm uses the less severe asymptotic assumption, $\max_i p_i = O(\epsilon)\mathrm{OPT}$, but runs in time $n^{(1/\epsilon)^{O(1/\epsilon)}}$.

[2] Partition: Decide if set of integers S can be partitioned into sets S_1, S_2 s.t the sum of S_1 and S_2 are equal.

Many variants of bin-packing have been considered, (see [10] for a survey). In multi-dimensional bin-packing (See [7, 24, 4, 9] and references therein), the bins are multi-dimensional, but the cost of each bin is still a fixed constant. In variable-size bin-packing (See [17, 30, 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 [15, 27]), 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 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 small items are considered fluid and can be split up arbitrarily among bins [14]. This approach was introduced in the bin-covering problem [11].

The VRP is another widely studied problem. The train delivery problem is the 1-dimensional version of VRP with unsplittable demands [20, 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 total demand and each customer must be served by 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 prior work that specifically considers the 1-dimensional setting. For the metric case, Haimovich, Rinnooy Kan, and Stougie give a constant factor approximation [20]. Bramel et al. study the problem on the Euclidean plane where customer demands are drawn i.i.d from any distribution [6]. Labbé et al. [26] give a 2-approximation for the problem on a tree. Additional heuristics that extend their approach were given in [28, 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 $W = O(\log \log n)$ [19]. Their approximation scheme partitions the customers into disjoint instances (far and $close$) based on the distance from the depot. 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]. They use a shifting technique [3, 21] to partition the instance into disjoint regions, and solve the problem in each region independently.

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

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

items), locations (rather than positions) and demands (rather than weights). We assume the depot is at the origin and that the instance is preprocessed:

Definition 1. *(Preprocessed) An instance is preprocessed if:*
- *No demand is has location $p_i \leq \epsilon \cdot p_{\max}/n$, where $p_{\max} = \max_i p_i$.*
- *All customers are located to the right of the depot.*

If there are demands located closer than $\epsilon \cdot p_{\max}/n$, serve them each with a separate tour. The overall cost for this at most $\epsilon p_{\max} \leq \epsilon \mathrm{OPT}$ as any solution must have cost at least $p_{\max}$. If there are customers on the right 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.

2 Algorithm for Theorem 1

We summarize the main steps of Algorithm 1 and provide details in the following subsections. Its correctness and running time follow from the lemmas below. See the full version of the paper (available on the authors' websites) for all proofs.

Algorithm 1 Asymptotic PTAS for train delivery

Input: A preprocessed input with demands $(p_i, w_i)_{1 \leq i \leq n}$ and train capacity W
Precondition: $\max_i p_i \leq \epsilon^6 \mathrm{OPT}$

1: Round the input using Algorithm 2.
2: **for** $1 \leq i \leq 1/\epsilon$ **do**
3: Let P_i be the i-th partition into regions (as in Definition 3).
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 $\mathrm{Best}(P_i)$ 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. The number of locations is reduced by rounding each location up to the next integer power of $(1+\epsilon)$. We call a demand w_i big if $w_i \geq \epsilon W$ and small otherwise. The rounding technique from classical bin packing is use to reduce the number of distinct big demand sizes at each location.
Partitioning into regions. We partition the demands into disjoint regions based on their distance from the depot (Definition 3) so that each region has only a constant number of locations containing demands. We solve the problem approximately within each region independently. A shifting argument shows that if we do this for a few different partitions, the union of the approximate solutions in at least one of the partitions yields a near optimal solution.
Solving within a region. Within each region, we treat big and small demands differently. We relax the unsplittable constraint for small demands and allow

them to be split up between multiple tours. The relaxed problem is solved by a linear program that considers the big demands individually and the total small demand weight at each location in the region. Since each region contains just a constant number of locations and each location contains a constant number of distinct big demand sizes, the total number of distinct big demands in R is constant. This allows us to solve the relaxed problem in constant time.

We construct a feasible solution from the relaxed solution by adding the small demands greedily, and prove that the solution 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 5). 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 $O(p_{\max})$, where $p_{\max}$ is the location of the furthest demand in the instance.

Our assumption $p_{\max} \leq O(\epsilon^6)\mathrm{OPT}$ ensures that the additive cost incurred by the greedy extension (the p_R terms) is within the desired approximation factor.

2.1 Rounding

Algorithm 2 outputs a rounded instance satisfying Lemma 1. Its proof extends the bin packing rounding analysis of [16].

Algorithm 2 Rounding Instance

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

1: Round each p_i up to the smallest 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 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

Lemma 1. *Given an instance I, 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.*
- *$OPT(I') \leq (1 + O(\epsilon))OPT(I)$.*

2.2 Partitioning into regions

We say that a tour has small expanse if it covers points in a bounded region. Lemma 2 shows that a near optimal solution can be obtained using only tours with this simple structure.

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 2. *There exists a small expanse tour solution of cost* $\leq (1+2\epsilon)OPT$.

We partition instance I' into regions, where each region has large expanse. 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. Thus we can find a near optimal solution by solving each region independently.

Definition 3. *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 $\leq 1/\epsilon$ 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 [21] and an averaging argument to show Lemma 3, which states that at least one partition yields a near optimal solution for I'.

Lemma 3. *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

Definition 4. *(Big demand type) 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) sizes at p. $n(d)$ denotes the 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 5. *(Configuration) A* configuration *f of a tour in R consists of r_f, the furthest location of the tour, and a multiset M_f of big 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 removing the unsplittable constraint on only the small demands. Algorithm 3 rounds the solution of the linear program below to obtain, by Lemma 4, a near optimal solution of the relaxed problem in constant time. The linear program has one variable x_f for each possible tour configuration f. The objective selects 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 the remaining capacities of the tour configurations that extend to the right of p.

$$\begin{aligned} \min \textstyle\sum_{f\in\mathcal{F}} r_f x_f \qquad & s.t. \\ \textstyle\sum_{f\in\mathcal{F}} x_f n_f(d) \geq n(d) \qquad & \text{for all demand types } d \qquad (1) \\ \textstyle\sum_{f:r_f\geq p} c_f x_f \geq \textstyle\sum_{(p_i,w_i)\in S, p_i\geq p} w_i \quad & \text{for all locations } p \qquad (2) \\ x_f \geq 0 & \end{aligned}$$

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 (Definition 4) and $\mathcal{F}$ be the set of tour configurations for region R (Definition 5).
2: Let $x^* = (x^*_f)_{f\in\mathcal{F}}$ be a basic solution of the linear program (Eq. 1-2).
3: Let $\bar{x}_f = \lceil x^*_f \rceil$ for each $f \in \mathcal{F}$.
4: 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 $\mathcal{D}$.

Lemma 4. *Let p_R denote the maximum location in region R, $OPT(R)$ the cost of the optimal solution to the (unrelaxed) problem and T the set of tours output by Algorithm 3. T is a solution to the relaxed problem that covers all demands in R such that the big demands are unsplittable and the small demands are allowed to be split. $Cost(T) \leq OPT(R) + p_R((1/\epsilon)^2 \log(1/\epsilon))(2 + 1/\epsilon^2)$. Algorithm 3 outputs T in time $(1/\epsilon)^{O(1/\epsilon)}$.*

2.4 Extending a relaxed solution of the region

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.

Let $(r_t, c_t)_t$ denote the set of tours output by Algorithm 3 where r_t denotes the maximum location and c_t the remaining capacity of tour t after it has covered

its big demands. Algorithm 4 takes the list of tours $(r_t, c_t)_t$ and extends it to cover the small demands of R in a feasible way (i.e., without splitting any of them). Lemma 5 shows that the greedy extension is a near optimal.

Lemma 5. *The output of Algorithm 4 G has cost$(G) \leq (1 + 2\epsilon)$cost$(T) + 2p_R$.*

3 Algorithm for Theorem 2

We summarize the main steps of Algorithm 5 and provide details in the following subsections. Its correctness and running time follow from the lemmas appearing in the subsections. See the full version of paper for all proofs.

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

Input: train capacity W, demands $(p_i, w_i)_{1 \leq i \leq n}$
Precondition: $W \leq poly(n)$.
1: Partition the instance into *close* and *far* using Algorithm 6
2: Find OPT(far) using Algorithm 7 and 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 location, identify a demand i_c and partition the instance into *close* and far, where far contains the demands with indices less than i_c and *close* the demands with indices greater than i_c.
Optimal solution of far. The partition is such that far contains $O(W)$ total demand. Thus an optimal solution of far uses a constant number of tours (Lemma 9). 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.
Overall solution. The solution is the union of the solutions for *close* and far. We use Algorithm 1 to find a near optimal solution to *close*. 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 holds for *close*, namely that $p_{i_c} = O(\epsilon^6)$OPT, where p_{i_c} denotes the furthest location in *close*.

3.1 Partitioning into *close* and far

Algorithm 6 partitions the instance. Lemma 6 proves that *close* and far can be solved separately at small additional cost. Its proof critically uses the definition of i_c. Lemma 7 also follows by choice of i_c.

Lemma 6. *Algorithm 6 returns two instances* far *and* close *s.t. OPT(close) + OPT(far) $\leq (1 + O(\epsilon))OPT$.*

Lemma 7. *$OPT > 2p_{i_c}/\epsilon^6$.*

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*

3.2 Solving the far instance

The following combinatorial lemma is an extension of Haimovich and Rinnooy Kan's [19] analysis to the case of unsplittable demands. It bounds the total demand in far by $O(W)$ by bounding the number of demands that violate the requirement $p_{i_c} \sum_{j \le i_c} w_j \le \epsilon \sum_{j=1}^{n} w_j p_j$.

Lemma 8. *Let i_c be as in Algorithm 6. Then $\sum_{j \le i_c} w_j = \exp(O(1/\epsilon))W$.*

Lemma 9 implies that OPT(far) uses a constant, c_{far}, number of tours. Then we can solve far using dynamic programming.

Lemma 9. *OPT uses at most $\lceil 2D/W \rceil$ tours, where D the total demand.*

Definition 6. *(Far Configuration) Let c_{far} denote the maximum number of tours OPT(far) may use. A* configuration *for far consists 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 $S^i = \{s_1^i, \ldots s_i^i\}$.*

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

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. 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 $W^{e^{O(1/\epsilon)}}$ time (i.e., polynomial in n assuming $W \le poly(n)$)) using a trivial extension of the dynamic program for the subset sum problem (omitted). Algorithm 7 solves far by iterating all configurations, checking feasibility, and returning the minimum cost feasible configuration.

Lemma 10. *Given an instance of far with demand $We^{O(1/\epsilon)}$ Algorithm 7 finds the optimal solution of far in time $W^{e^{O(1/\epsilon)}}$, which is polynomial in n and W.*

Algorithm 7 Solving the far instance

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

for each configuration f of far as given in definition 6 **do**
  for each tour $j \leq c_{far}$ **do**
    if $\sum_{i \leq c_{far}} s_j^i > W$ **then**
      Mark f infeasible. {capacity of tour is exceeded}
  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 $s_1^i, \ldots, s_i^i$ **then**
      Mark f infeasible. {demands cannot be partitioned}

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

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. M. Azizoglu and S. Webster. Scheduling a batch processing machine with non-identical job sizes. *Intern. J. Prod. Res.*, 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, E. G. Coffman, Jr., 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.
7. A. Caprara. Packing 2-dimensional bins in harmony. In *FOCS*, pages 490–499, Washington, DC, USA, 2002. IEEE.
8. B. Chandran and S. Raghavan. *The vehicle routing problem: latest advances and new challenges*, volume 43, pages 239–261. Springer US, 2008.
9. C. Chekuri and S. Khanna. On multi-dimensional packing problems. In *SODA*, pages 185–194. SIAM, 1999.
10. E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson. *Approximation algorithms for bin packing: a survey*, pages 46–93. PWS Pub. Co., Boston, MA, USA, 1997.
11. J. Csirik, D.S. Johnson, and C. Kenyon. Better approximation algorithms for bin covering. In *SODA*, pages 557–566, PA, USA, 2001. SIAM.
12. L. Dupont and C. Dhaenens-Flipo. Minimizing the makespan on a batch machine with non-identical job sizes: an exact procedure. *Comp. and Op. Res.*, 29(7):807 – 819, 2002.
13. L. Dupont and F. J. Ghazvini. Minimizing makespan on a single batch processing machine with non-identical job sizes. *Eur. J. Auto. Sys.*, 32:431–440, 1998.
14. L. Epstein and A. Levin. An APTAS for generalized cost variable-sized bin packing. *SIAM J. Comp.*, 38:411–428, 2008.
15. L. Epstein and A. Levin. Bin packing with general cost structures. *Mathematical Programming*, pages 1–37, 2010.

16. 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.
17. D K Friesen and M A Langston. Variable sized bin packing. *SIAM J. Comput.*, 15(1):222–230, 1986.
18. M. R. Garey and D. S. Johnson. "strong" NP-completeness results: motivation, examples, and implications. *J. ACM*, 25(3):499–508, 1978.
19. M. Haimovich and A. H. G. Rinnooy Kan. Bounds and heuristics for capacitated routing problems. *Math. of Op. Res.*, 10(4):527–542, Nov., 1985.
20. M. Haimovich, A.H.G. Rinnooy Kan, and L. Stougie. Analysis of heuristics for vehicle routing problems. *Vehicle routing: methods and studies. Manag. Sci. Systems.*, 16:47–61, 1988.
21. 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.
22. N. Karmarkar and R. M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. *FOCS*, pages 312–320, 1982.
23. A. H. Kashan, B. Karimi, and F. Jolai. Effective hybrid genetic algorithm for minimizing makespan on a single-batch-processing machine with non-identical job sizes. *Intern. J. Prod. Res.*, 44:2337–2360, 2006.
24. C. Kenyon and E. Rémila. A near-optimal solution to a two-dimensional cutting stock problem. *Math. Oper. Res.*, 25(4):645–656, 2000.
25. 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. *Int. J. Prod. Econ.*, 98:81–96, 2005.
26. M. Labbé, G. Laporte, and H. Mercure. Capacitated vehicle routing on trees. *Op. Res.*, 39(4):616–622, 1991.
27. C.L. Li and Z.L Chen. Bin-packing problem with concave costs of bin utilization. *Nav. Res. Log.*, 53(4):298–308, 2006.
28. P. Mbaraga, A. Langevin, and G. Laporte. Two exact algorithms for the vehicle routing problem on trees. *Nav. Res. Log.*, 46:75–89, 1999.
29. S. Melouk, P. Damodaran, and P.-Y. Chang. Minimizing makespan for single machine batch processing with non-identical job sizes using simulated annealing. *Inter. J. of Prod. Econ.*, pages 141–147, 2004.
30. F.D. Murgolo. An efficient approximation scheme for variable-sized bin packing. *SIAM J. Comput.*, 16(1):149–161, 1987.
31. N. R. Parsa, B. Karimi, and A. H. Kashan. A branch and price algorithm to minimize makespan on a single batch processing machine with non-identical job sizes. *Com. Op. Res.*, 37(10):1720 – 1730, 2010.
32. R. Uzsoy. Scheduling a single batch processing machine with non-identical job sizes. *Int. J. of Prod. Res.*, 32:1615–1635, July 1994.
33. G. Zhang, X. Cai, C.-Y Lee, and C.K Wong. Minimizing makespan on a single batch processing machine with nonidentical job sizes. *Nav. Res. Log.*, 48:226–240, 2001.
34. Y. Zhang and Z. Cao. An asymptotic PTAS for batch scheduling with nonidentical job sizes to minimize makespan. In *COCOA*, pages 44–51, Berlin, Heidelberg, 2007. Springer-Verlag.