



# Approximation Algorithms for Replenishment Problems with Fixed Turnover Times

Thomas Bosman<sup>1</sup>(✉) , Martijn van Ee<sup>6</sup>, Yang Jiao<sup>2</sup>,  
Alberto Marchetti-Spaccamela<sup>3,5</sup>, R. Ravi<sup>2</sup> , and Leen Stougie<sup>1,4,5</sup>

<sup>1</sup> Vrije Universiteit, Amsterdam, The Netherlands  
{thomas.bosman,l.stougie}@vu.nl

<sup>2</sup> Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA, USA  
{yangjiao,ravi}@andrew.cmu.edu

<sup>3</sup> Sapienza University of Rome, Rome, Italy  
alberto@dis.uniroma1.it

<sup>4</sup> Centrum voor Wiskunde en Informatica (CWI),  
Amsterdam, The Netherlands  
stougie@cwi.nl

<sup>5</sup> Erable, Inria, Paris, France

<sup>6</sup> Netherlands Defence Academy, Den Helder, The Netherlands  
m.v.ee.01@mindef.nl

**Abstract.** We introduce and study a class of optimization problems we coin replenishment problems with fixed turnover times: a very natural model that has received little attention in the literature. Nodes with capacity for storing a certain commodity are located at various places; at each node the commodity depletes within a certain time, the turnover time, which is constant but can vary between locations. Nodes should never run empty, and to prevent this we may schedule nodes for replenishment every day. The natural feature that makes this problem interesting is that we may schedule a replenishment (well) before a node becomes empty, but then the next replenishment will be due earlier also. This added workload needs to be balanced against the cost of routing vehicles to do the replenishments. In this paper, we focus on the aspect of minimizing routing costs. However, the framework of recurring tasks, in which the next job of a task must be done within a fixed amount of time after the previous one is much more general and gives an adequate model for many practical situations.

Note that our problem has an infinite time horizon. However, it can be fully characterized by a compact input, containing only the location of each store and a turnover time. This makes determining its computational complexity highly challenging and indeed it remains essentially unresolved. We study the problem for two objectives: MIN-AVG minimizes the average tour length and MIN-MAX minimizes the maximum tour length over all days. For MIN-MAX we derive a logarithmic factor approximation for the problem on general metrics and a 6-approximation for the problem on trees, for which we have a proof of NP-hardness. For MIN-AVG we present a logarithmic approximation on general metrics, 2-approximation for trees, and a pseudopolynomial time algorithm for the line. Many intriguing problems remain open.

# 1 Introduction

Imagine the following particular inventory-routing problem. A set of automatic vendor machines are spread over a country or a city. They have a certain turnover time: the number of days in which a full machine will be sold out. Replenishment is done by vehicles. Let us assume that turnover times are machine dependent but not time dependent, and that it is highly undesirable to have an empty machine. However, the holding costs of the machine are negligible, so that we will always fill the machine to capacity. There is nothing against replenishing a machine before it has become empty, but then the next replenishment will due earlier as well. That is, the deadline of the next replenishment is always within the turnover time after the last replenishment. Equivalently, in any consecutive number of days equal to the turnover time, at least one replenishment has to take place. Replenishing a machine earlier to combine it with the replenishment of another machine that is due earlier may lead to cost savings. The feature that makes this problem so special w.r.t. existing literature, is that it can be compactly modeled by only specifying for every machine its location and the turnover time. The feature is very natural but has hardly been studied in the existing literature. There are intriguing basic open complexity questions, and some highly non-trivial results.

The motivation for studying this problem comes *linea recta* from a business project for the replenishment of ATMs in the Netherlands, in which some of the co-authors are involved. The replenishment of the ATMs of all the large banks in the Netherlands has been outsourced to a single company: Geld Service Nederland. Of course the real-life ATM replenishment problem is not as stylized as described above; the turnover time is not strictly the same over time but subject to variability, there are restrictions on the routes for the vehicles, etc. But the feature that is least understood in the ATM-problem is exactly the problem of how to deal with the trade-off between replenishing an ATM earlier than its due date leading to a higher frequency of replenishments and the savings on vehicle routing costs.

Formally, an instance of the problem that we study in this paper, which we baptize the REPLENISHMENT PROBLEM WITH FIXED TURNOVER TIMES (RFTT), consists of a pair  $(G, \tau)$ , where  $G = (V \cup \{s\}, E, c)$  is a weighted graph with a designated depot vertex  $s$  and weights on the edges  $c : E \rightarrow \mathbb{R}_+$ , and turnover times  $\tau \in \mathbb{N}^{|V|}$ , indicating that  $v_j \in V$  should be visited at least once in every interval of  $\tau_j$  days.

A solution consists, for each day  $k$ , of a tour  $T_k$  in  $G$  starting in and returning to the depot  $s$  and visiting a subset of the vertices  $J_k \subseteq V$ . It is feasible if  $v_j \in \bigcup_{k=t+1}^{t+\tau_j} J_k, \forall t$  and  $\forall v_j \in V$ . We will focus on solutions that repeat themselves after a finite amount of time, that is, in which  $(T_k, \dots, T_{k+\ell}) = (T_{k+\ell+1}, \dots, T_{k+2\ell})$  for some  $\ell$ , and all  $k$ . Since all turnover times are finite, this is no real restriction.

We consider two versions of RFTT. In the first version, called MIN-AVG, the goal is to find a feasible solution that minimizes the average tour length. In the MIN-MAX problem, we want to find a feasible solution that minimizes the maximum tour length over all days.

We emphasize that the particular feature of this model, that jobs or visits to clients recur and need to be done within each job-specific consecutive time interval occurs naturally in many problem settings. It allows any job of a recurring task to be done before its deadline, but then the next job of the task comes earlier and hence its deadline. This is a feature that, despite its natural applicability, has hardly been studied in the literature from a theoretical point of view.

*Related work.* As mentioned before, our problem can be seen as a special case of the INVENTORY ROUTING PROBLEM (IRP) [8]. Here, clients (vertices) have their own storage with a certain capacity and for each day a demand is specified. The clients pay holding cost over their inventory. However, omitting inventory cost, we can interpret our problem as such an inventory routing problem in which the demand at any given location is the same every day, leading to a very small input description of our problem consisting only of a location and a turnover time (storage capacity divided by daily demand), which makes it incomparable to the inventory routing problem from a complexity point of view. Indeed it is unclear if the decision version of our problem is in NP or in co-NP.

Another closely related problem is the PERIODIC LATENCY PROBLEM [9], which features the recurring visits requirement of RFTT. We are given recurrence length  $q_i$  for each client  $i$  and travel distances between clients. Client  $i$  is considered *served* if it is visited every  $q_i$  time units. The server does not return to the depot at the end of each time unit (e.g. day), but keeps moving continuously between clients at uniform speed. Another difference between PERIODIC LATENCY PROBLEM and RFTT is the objective function. Coene et al. [9] study two versions of the problem: one that maximizes the number of served clients by one server, and one that minimizes the number of servers needed to serve all clients. They resolve the complexity of these problems on lines, circles, stars, trees, and general metrics.

A problem that does share the compact input size and is in fact a very special case of our problem is known under the guise of PINWHEEL SCHEDULING. It has been introduced to model the scheduling of a ground station to receive information from a set of satellites without data loss. In terms of our problem no more than one vertex can be replenished per day and all distances to the depot are the same; the interesting question here is if there exists a feasible schedule for replenishing the vertices. Formally, a set of jobs  $\{1, \dots, n\}$  with periods  $p_1, \dots, p_n$  is given, and the question is whether there exists a schedule  $\sigma : \mathbb{N} \rightarrow \{1, \dots, n\}$  such that  $j \in \bigcup_{k=t+1}^{t+p_j} \sigma_k, \forall t \geq 0$  and  $\forall j$ .

PINWHEEL SCHEDULING was introduced by Holte et al. [17], who showed that it is contained in PSPACE. The problem is in NP if the schedule  $\sigma$  is restricted to one in which for each job the time between two consecutive executions remains constant throughout the schedule. In particular this holds for instances with density  $\rho = \sum_j 1/p_j = 1$  [17]. They also observed that the problem is easily solvable when  $\rho \leq 1$  and the periods are harmonic, i.e.  $p_i$  is a divisor of  $p_j$  or vice versa for all  $i$  and  $j$ . As a corollary, every instance with  $\rho \leq \frac{1}{2}$  is feasible.

Chan and Chin [7] improved the latter result by giving an algorithm that produces a feasible schedule for PINWHEEL SCHEDULING whenever  $\rho \leq \frac{2}{3}$ . In [6], they improved this factor to  $\frac{7}{10}$ . Later, Fishburn and Lagarias [14] showed that every instance with  $\rho \leq \frac{3}{4}$  has a feasible schedule. All these papers work towards the conjecture that there is a feasible schedule if  $\rho \leq \frac{5}{6}$ . That this bound is tight can be seen by the instance with  $p_1 = 2$ ,  $p_2 = 3$  and  $p_3 = M$ , with  $M$  large. This instance cannot be scheduled, but has a density of  $\frac{5}{6} + \frac{1}{M}$ .

The complexity of PINWHEEL SCHEDULING has been open since it was introduced. It was only recently shown by Jacobs and Longo [18] that there is no pseudopolynomial time algorithm solving the problem unless SAT has an exact algorithm running in expected time  $n^{O(\log n \log \log n)}$ , implying for example that the randomized exponential time hypothesis fails to hold [5, 10]. Since the latter is unlikely, one could conclude that PINWHEEL SCHEDULING is not solvable in pseudopolynomial time. It remains open whether the problem is PSPACE-complete.

Similar to PINWHEEL SCHEDULING, the  $k$ -SERVER PERIODIC MAINTENANCE PROBLEM [2, 11, 19] has  $n$  jobs, each with a specified periodicity and a processing time. Each server may serve at most one job per time unit. However, job  $i$  is required to be served exactly every  $m_i$  days apart rather than within every  $m_i$  days. The case  $k = 1, c_j = 1$  for all  $j$  is analogous to PINWHEEL SCHEDULING, except for the exact periodicity constraint. For any  $k \geq 1$ , Mok et al. [19] have shown it is NP-complete in the strong sense. For the special case when  $m_i$  are multiples of each other or when there are at most 2 different periodicities, they have shown it is in P.

Other related problems with a compact input representation include real-time scheduling of sporadic tasks [1, 3], where we are given a set of recurrent tasks. On a single machine, EDF (Earliest Deadline First) is optimal. However, we remark that the complexity of deciding whether a given set of tasks is feasible has been open for a long time and only recently proved showing that it is coNP-hard to decide whether a task system is feasible on a single processor even if the utilization is bounded [12].

Another related problem is the BAMBOO GARDEN TRIMMING PROBLEM introduced by Gasieniec et al. [16]. There are  $n$  bamboos, each having a given growth rate, which may be viewed as inducing a periodicity. On each day, a robot may trim at most one bamboo back to height 0. The goal is to minimize the maximum height of the bamboos. Gasieniec et al. provide a 4-approximation for the general case and a 2-approximation for balanced growth rates.

*This paper.* We investigate the computational complexity of both the MIN-MAX and the MIN-AVG version of RFTT. Mostly we will relate their complexity to the complexity of PINWHEEL SCHEDULING. Some interesting inapproximability results follow from this relation. After that, we will start with some special cases. In Sect. 3, we give our most remarkable result, a constant factor approximation for MIN-MAX on a tree, next to a less remarkable constant approximation for the MIN-AVG version on the tree. In the same section, we show for MIN-AVG that the problem can be solved to optimality in pseudopolynomial time on

line metrics. Finally, in Sect. 4, we present logarithmic factor approximations for both problem versions on general metrics.

## 2 Complexity

In this section, we investigate the computational complexity for both objectives. Since our problem requires finding a shortest tour visiting some subset of vertices for every day, it is at least as hard as the TRAVELING SALESMAN PROBLEM (TSP). However it is also interesting to note that the problems are at least as hard as PINWHEEL SCHEDULING as well. For the MIN-MAX objective there is a direct reduction showing that a factor 2 approximation is at least as hard as PINWHEEL SCHEDULING: construct an unweighted star with the depot at the center and each leaf corresponding to a job in the pinwheel instance. This instance has value 2 only if there exists a pinwheel schedule and at least 4 otherwise.

For the MIN-AVG RFTT the reduction is a bit more involved, and given in the appendix of the full version of this paper [4].

**Theorem 1.** *On series-parallel graphs, MIN-AVG RFTT is at least as hard as PINWHEEL SCHEDULING.*

We note that this hardness result is incomparable to the TSP reduction. Pinwheel is neither known to be NP-hard nor in NP, although it is conjectured to be PSPACE-complete.

Lastly, as Theorem 2 shows, the MIN-MAX RFTT remains hard even on star graphs (where TSP is trivial). A reduction can be found in the appendix of the full paper [4].

**Theorem 2.** *MIN-MAX RFTT is NP-hard on star graphs.*

## 3 Approximation on Trees

In this section we give a 2-approximation for MIN-AVG and a 6-approximation for MIN-MAX on trees.

We start out with a simplifying result, which will also be of use in the next sections.

**Lemma 1.** *Given an instance  $(G, \tau)$  of RFTT, let  $\tau'$  be found by rounding every turnover time in  $\tau$  down to a power of 2. Then  $OPT(G, \tau') \leq 2OPT(G, \tau)$  for both MIN-AVG and MIN-MAX objectives.*

*Proof.* Let  $(G, \bar{\tau})$  denote the instance found from  $(G, \tau)$  by rounding every turnover time up to a power of 2. Since any schedule remains feasible if we round up the turnover times, we have that  $OPT(G, \bar{\tau}) \leq OPT(G, \tau) \leq OPT(G, \tau')$ .

Suppose we have an optimal solution for  $(G, \bar{\tau})$  in which  $\bar{T}_k$  is scheduled on day  $k$ . We can construct a feasible schedule for  $(G, \tau')$  by scheduling the concatenation of  $\bar{T}_{2k-1}$  and  $\bar{T}_{2k}$  on day  $k$ . The maximum tour length in this schedule is at most twice that of the optimal solution for  $(G, \bar{\tau})$  and every tour from the original schedule is visited exactly twice in the new schedule, so this yields a factor 2 increase in both the MIN-MAX and the MIN-AVG objective.

In the remainder we assume w.l.o.g. that  $G$  is rooted at  $s$  and that turnover times are increasing on any path from the depot to a leaf node in  $G$ . Furthermore, for an edge  $e$  in  $E$  we define  $D(e)$  to be the set of vertices that are descendants of  $e$ . We also need the following definition.

**Definition 1 (tt-weight of an edge).** For any edge in  $G$  we define:

$$q(e) = \min_{j \in D(e)} \tau_j.$$

We call this quantity the *tt-weight (turnover time-weight)* of  $e$ .

This definition allows us to express the lowerbound in Lemma 2.

**Lemma 2 (tt-weighted tree).** For an instance  $(G, \tau)$  of the RFTT on trees it holds that the average tour length is at least:

$$L(G, \tau) := 2 \sum_{e \in E} \frac{c(e)}{q(e)}.$$

*Proof.* This follows immediately from the fact that  $\frac{2}{q(e)}$  lower bounds the number of times edge  $e$  must be traversed on average in any feasible solution.

Since the maximum tour length is at least the average tour length, Lemma 2 also provides a lower bound for the MIN-MAX objective.

An approximation for MIN-AVG RFTT is thus found by rounding all turnover times to powers of 2 and then visit each client  $j$  on every day that is a multiple of  $\tau_j$ . Since in that case the lower bound of Lemma 2 is exactly attained on the rounded instance, Lemma 1 implies the following theorem.

**Theorem 3.** There is a 2-approximation for MIN-AVG RFTT on trees.

### 3.1 MIN-MAX

We will now show that we can achieve a 6-approximation for MIN-MAX RFTT on trees by providing a 3-approximation algorithm if all turnover times are powers of 2 and then applying Lemma 1.

The main idea is to take a TSP-tour and recursively split it to obtain a schedule for the clients with increasing turnover times. During the splitting process, we assign each client  $j$  on that tour to a congruence class  $\bar{a}_{\tau_j} = \{k \in \mathbb{N} | k \equiv a \pmod{\tau_j}\}$  for some  $a \leq \tau_j$ , to indicate we want to visit  $j$  on each day in  $\bar{a}_{\tau_j}$ . Similarly, we distribute all edges  $e$  to a congruence class  $\bar{a}_{q(e)}$ . We do this ensuring that on any given day, we can create a tour through all clients associated with that day, using the edges associated with that day plus a small set of extra edges.

Let us define some further notation. For a given congruence class  $\bar{a}_m \subseteq \mathbb{N}$ , we denote  $g(\bar{a}_m) \subseteq V$  the set of vertices and  $f(\bar{a}_m) \subseteq E$  the set of edges assigned to that class. Note that  $\bar{a}_m$  and  $(\bar{a} + m)_m$  define the same congruence class, so

$f(\bar{a}_m) = f(\overline{(a+m)}_m)$ . Then, for any  $k \in \mathbb{N}$  we have that  $J_k$ , the set of clients we need to visit on day  $k$ , is

$$J_k = \bigcup_{m \in \mathbb{N}, a \leq m | k \in \bar{a}_m} g(\bar{a}_m).$$

---

**Algorithm 1.** Algorithm for recursively constructing  $f(\cdot)$  and  $g(\cdot)$

---

**function** RECURSE TREESCHEDULE( $\mathbf{d}, a, m$ )

**Require:**  $\mathbf{d}$ , a connected sequence of edges in  $G$ , powers of 2 turnover times  $\tau$ ;  $a, m$ , integers

**if**  $\mathbf{d} \neq \emptyset$  **then**

$f(\bar{a}_m) = \{e \in \mathbf{d} \mid q(e) = m\}$

$g(\bar{a}_m) = \{j \in V(\mathbf{d}) \mid \tau_j = m\}$

$k = \max_{k'} \text{ s.t. } \sum_{i \in [k'-1] | q(d_i) > m} \frac{c(d_i)}{q(d_i)} \leq \frac{1}{2} \sum_{i \in [n] | q(d_i) > m} \frac{c(d_i)}{q(d_i)}$

$\mathbf{d}^1 = (d_1, \dots, d_{k-1})$

$\mathbf{d}^2 = (d_{k+1}, \dots, d_n)$

RECURSE TREESCHEDULE( $\mathbf{d}^1, a, 2m$ ), RECURSE TREESCHEDULE( $\mathbf{d}^2, a + m, 2m$ )

**end if**

**end function**

---

The assignment of vertices and edges to classes is guided by the recursive splitting of a TSP-tour in  $G$ . The full procedure for constructing  $f(\cdot)$  and  $g(\cdot)$  is shown in Algorithm 1. The algorithm is initially called with  $\mathbf{d}$ , a TSP-tour visiting all vertices in  $G$ , and  $a = m = 1$  and will determine the set of vertices to be visited on every day (i.e., those congruent to  $\bar{a}_1$ ). Then the first (second) recursive call determines the sets of vertices with turnover time 2 that will be visited on odd (even) days. Analogously, RECURSE TREESCHEDULE( $\mathbf{d}^1, a, m$ ) will return the set of vertices with turnover time  $m$  to be visited on days in the congruence class  $\bar{a}_m$  and the two recursive calls will return the set of vertices with turnover time  $2m$  that are visited on days  $a, a + 2m, a + 4m, \dots$  and  $a + m, a + 3m, a + 5m, \dots$ , respectively.

In the remainder we assume that any call to  $f(\cdot)$  and  $g(\cdot)$  returns the empty set for any argument that is not explicitly handled in Algorithm 1. Note that we use the notation  $V(A)$  to denote the vertices incident to edges in  $A \subseteq E$ .

**Lemma 3.** *After Algorithm 1 terminates, each vertex  $j$  appears in some set  $g(\bar{a}_{\tau_j})$  for some  $a$ .*

*Proof.* Note that  $\mathbf{d}^1 \cap \mathbf{d}^2 = \emptyset$  and that  $|\mathbf{d}^1 \cup \mathbf{d}^2| = |\mathbf{d}| - 1$ ; since  $\mathbf{d}$  is a connected set of edges then in each call to RECURSE TREESCHEDULE,  $V(\mathbf{d}^1) \cup V(\mathbf{d}^2) = V(\mathbf{d})$ . Therefore no vertex is skipped in the construction of  $g(\cdot)$ .

In order to find a tour on day  $k$  through the vertices in  $J_k$  we use edges in  $\bigcup_{h=1,2,\dots,m} f(\bar{a}_h)$ ; as we already observed this set of edges does not necessarily connect vertices in  $g(\bar{a}_m)$  to the depot. The next lemma shows that a

tree that connects all vertices in  $g(\bar{a}_m)$  to the root can be found by considering  $\bigcup_{h=1,2,\dots,m} f(\bar{a}_h)$  and adding a shortest path from some vertex in  $g(\bar{a}_m)$  to the depot.

**Lemma 4.** *Let  $a, m$  be such that  $f(\bar{a}_m)$  is nonempty. Let  $P$  be the set of edges on the shortest path connecting some arbitrary edge in  $f(\bar{a}_m)$  to the root of  $G$ . Then the following edge set forms a connected component:*

$$T(\bar{a}_m) := P \cup \left( \bigcup_{h=1,2,\dots,m} f(\bar{a}_h) \right).$$

Moreover,  $T(\bar{a}_m)$  spans  $\bigcup_{h=1,2,\dots,m/2,m} g(\bar{a}_h)$ .

*Proof.* To prove our first claim, we first show that for  $k \leq m$ ,  $f(\bar{a}_k)$  either induces at most one connected component, or each component it induces is incident to a component induced by  $\bigcup_{h=1,2,\dots,k/2} f(\bar{a}_h)$ . Then, we will show that if  $f(\bar{a}_k)$  induces at most one connected component, it is incident to  $P \cup \left( \bigcup_{h=1,2,\dots,k/2} f(\bar{a}_h) \right)$ .

Suppose  $f(\bar{a}_k)$  does not induce at most one component. Note that  $f(\bar{a}_k)$  is the subset of edges in some connected edge sequence  $\mathbf{d}$  through  $G$  that have tt-weight  $k$ . But by the way tt-weight is defined and the fact that  $G$  is a tree, a simple path connecting disjoint edges with tt-weight  $k$ , can only consist of edges with tt-weight at most  $k$ . So every two components in  $f(\bar{a}_k)$  are connected through a path of edges with tt-weight of at most  $k$ . Moreover since the sequence  $\mathbf{d}$  used to construct  $f(\bar{a}_k)$  is a subset of the sequence used to construct  $f(\bar{a}_{k/2})$ , by induction these connecting paths must be contained in  $\bigcup_{h=1,2,\dots,k/2} f(\bar{a}_h)$ , as required.

Next we show that for any  $k \leq m$  such that  $f(\bar{a}_k) \neq \emptyset$ , if  $f(\bar{a}_k)$  is not incident to  $P$  then it is incident to  $\bigcup_{h=1,\dots,k/2} f(\bar{a}_h)$ .

Let  $\mathbf{d}$  be the sequence that was used to construct  $f(\bar{a}_k)$ . Since  $\mathbf{d}$  contains all edges in  $f(\bar{a}_m)$  and  $P$  contains at least one such edge, there exists a minimal path  $Q$  that contains some edge  $e$  in  $f(\bar{a}_k)$  such that  $Q$  is connected to  $P$ . Moreover since  $Q$  is minimal and  $P$  contains the root,  $e$  must be the edge furthest away from the root on  $Q$ . This implies that all edges on  $Q$  have tt-weight  $k$  or less. Now suppose that  $Q$  contains edges with tt-weight strictly less than  $k$ . Then those edges are necessarily in  $\bigcup_{h=1,\dots,k/2} f(\bar{a}_h)$  and therefore  $f(\bar{a}_k)$  is incident to that set. If not then  $Q$  is strictly contained in  $f(\bar{a}_k)$  and therefore  $f(\bar{a}_k)$  is connected to  $P$ .

The first claim of our lemma now follows by induction.  $P \cup f(\bar{a}_1)$  is clearly connected. If  $P \cup \bigcup_{h=1,2,\dots,k/2} f(\bar{a}_h)$  is connected, we get that  $f(\bar{a}_k)$  is either empty or is connected to  $P$  or to  $\bigcup_{h=1,2,\dots,k/2} f(\bar{a}_h)$ , and the result follows.

To prove our second claim, suppose that for some  $k$  and  $j \in g(\bar{a}_k)$  it holds that no edge incident to  $j$ , is in  $\bigcup_{h=1,2,\dots,k} f(\bar{a}_h)$ . We will show that  $j$  appears on  $P$ , from which our claim immediately follows.

Let  $\mathbf{d}$  be the sequence used to construct  $g(\bar{a}_k)$ . The edge  $e$  incident to  $j$  that is closest to the root, satisfies  $q(e) \leq k$ . So, it cannot be in  $\mathbf{d}$  otherwise it would

be contained in  $\bigcup_{h=1,2,\dots,k} f(\bar{a}_h)$ . But this implies that  $e$  cuts off every edge in  $\mathbf{d}$  from the root, and therefore  $e$  appears on  $P$ , as claimed, concluding the proof.

The next lemma allows us to bound the cost of edges included in  $f(\bar{a}_h)$ .

**Lemma 5.** *During each (recursive) call to RECURSETREESCHEDULE, it holds that*

$$\sum_{e \in \mathbf{d} \mid q(e) \geq m} m \frac{c(e)}{q(e)} + \sum_{h=1}^{m/2} \sum_{e \in f(\bar{a}_h) \mid q(e)=h} c(e) \leq L(G, \tau).$$

*Proof.* The proof is by induction on  $m$ . Since we initially call the algorithm with  $\mathbf{d}$  a TSP-tour in  $G$ , which visits each edge twice, it clearly holds for  $m = 1$ .

Now for  $m > 1$ , suppose it holds for all smaller  $m$ . Without loss of generality, suppose we have a call to the function with input  $\mathbf{d}^1, a, m$ , such that  $\mathbf{d}, a, m/2$  are the input parameters for its parent in the call stack.

$$\begin{aligned} & \sum_{e \in \mathbf{d}^1 \mid q(e) \geq m} m \frac{c(e)}{q(e)} + \sum_{h=1,2,\dots,\frac{m}{2}} \sum_{e \in f(\bar{a}_h) \mid q(e)=h} c(e) \\ = & \sum_{e \in \mathbf{d}^1 \mid q(e) \geq m} m \frac{c(e)}{q(e)} + \sum_{e \in f(\bar{a}_{m/2}) \mid q(e)=m/2} \frac{m}{2} \frac{c(e)}{q(e)} + \sum_{h=1,2,\dots,\frac{m}{4}} \sum_{e \in f(\bar{a}_h) \mid q(e)=h} c(e) \\ \leq & \sum_{e \in \mathbf{d}^1 \mid q(e) \geq m} \frac{m}{2} \frac{c(e)}{q(e)} + \sum_{e \in f(\bar{a}_{m/2}) \mid q(e)=m/2} \frac{m}{2} \frac{c(e)}{q(e)} + \sum_{h=1,2,\dots,\frac{m}{4}} \sum_{e \in f(\bar{a}_h) \mid q(e)=h} c(e) \\ \leq & \sum_{e \in \mathbf{d}^1 \mid q(e) \geq m/2} \frac{m}{2} \frac{c(e)}{q(e)} + \sum_{h=1,2,\dots,\frac{m}{4}} \sum_{e \in f(\bar{a}_h) \mid q(e)=h} c(e) \leq L(G, \tau) \end{aligned}$$

For the first equality, we split the second sum into an  $h = m/2$  part and an  $h = 1, \dots, m/4$  part. In the first inequality we used the way  $\mathbf{d}^1$  and  $\mathbf{d}^2$  are determined in Algorithm 1, in the second inequality we used that  $f(\bar{a}_m) \subseteq \mathbf{d}$  and in the last inequality we used the inductive hypothesis, concluding the proof.

We are now ready for the main theorem.

**Theorem 4.** *There is a 6-approximation for MIN-MAX RFTT on trees.*

*Proof.* We first round all turnover times down to powers of 2, which loses a factor of 2 in the optimal solution. We then use Algorithm 1 to construct  $f(\cdot)$  and  $g(\cdot)$  thus determining the set of vertices  $J_k$  to be visited on day  $k$ . By Lemma 3 this defines a feasible schedule.

If we then take  $T(\bar{k}_{\tau_{max}})$  as in Lemma 4, we get a tree that spans  $J_k$ . Moreover the weight of  $T(\bar{k}_{\tau_{max}})$  is at most  $\frac{3}{2}OPT$ : the contribution of  $P$  is at most  $\frac{1}{2}OPT$ , since we need to reach any client at least on some day (and drive back), while the contribution of  $\bigcup_{h=1,\dots,\tau_{max}} f(\bar{k}_h)$  is at most  $L(G, \tau) \leq OPT$ , which can be seen by applying Lemma 5 for  $m = 2\tau_{max}$ . Lastly, since we need a tour around  $T(\bar{k}_{\tau_{max}})$ , we lose another factor 2. This gives the approximation factor of 6.

It remains to show that Algorithm 1 runs in polynomial time, and that we can find a polynomial representation for the schedule. For the first claim, note that in each recursive call to the algorithm, the following equality holds  $|\mathbf{d}^1 \cup \mathbf{d}^2| = |\mathbf{d}| - 1$ ; hence the algorithm terminates after at most  $2|E(G)|$  calls.

For the second claim, the crucial observation is that we only need to store the entries of  $g$  for  $a$  and  $m$  such that  $g(\bar{a}_m)$  is nonempty. Since at most one entry is defined in every call to the algorithm, and we can simply check if  $k \equiv a \pmod{m}$  for all stored entries, the claim, and the theorem, follow.

### 3.2 MIN-AVG on the Line

As an even more special underlying metric, we might consider the MIN-AVG problem on the line (on a path). For the MIN-MAX version this case is trivial, but for the MIN-AVG version its complexity is unclear: we do not know whether it is in NP, although we expect it to be NP-hard.

On the positive side we can show that the problem is not strongly NP-hard.

**Theorem 5.** *MIN-AVG on the line can be solved in pseudopolynomial time.*

We give a DP that finds an optimal schedule in polynomial time for any instance with polynomially bounded turnover times. Since we are minimizing the average, it is easy to see that we can reduce this problem to two times the MIN-AVG problem on the half-line (a path with the depot in one of the leaves). On the half-line each vertex  $i$  has a distance  $d_i \in \mathbb{N}$  from the origin. Suppose vertices are numbered such that  $d_1 \leq \dots \leq d_n$ . We present a pseudopolynomial time dynamic programming algorithm for this problem, based on the following observations.

First of all, we note that on any tour visiting vertex  $j$  automatically visits every vertex  $i < j$ . As in the tree case, we therefore assume that  $\tau_i \leq \tau_j$  for  $i < j$ . Thus, after visiting  $j$ , all  $i \leq j$  have a remaining turnover time of  $\tau_i$ . For the dynamic program to work, we guess  $L$ , the day on which vertex  $n$  is visited for the first time and try all guesses between 1 and  $\tau_n$ .

The dynamic program now works as follows. Suppose we are given the optimal solution for vertices  $1, \dots, i - 1$  when only considering the days  $1, \dots, k$ . Now we want to include  $i$  in the optimal solution for the first  $k$  days. If  $k < \min\{\tau_i, L\}$ , it is not necessary to visit  $i$  during the first  $k$  days, and hence it is optimal to take the optimal solution for the first  $i - 1$  vertices and  $k$  days. Otherwise, we need to visit  $i$  on some day  $\ell$  in  $\{1, \dots, \min\{\tau_i, L\}\}$ . Before day  $\ell$ , we only need to visit the vertices  $1, \dots, i - 1$ . Thus, we take the optimal  $\ell - 1$  tours for visiting the first  $i - 1$  vertices in the first  $\ell - 1$  days. After day  $\ell$ , all vertices have the same remaining turnover time as they had at time zero. Hence, we can take the optimal tours for the first  $i$  vertices and  $k - \ell$  days.

Let  $\phi_L(i, k) :=$  the minimum cost of the first  $k$  tours visiting vertices  $1, \dots, i$ . We initialize  $\phi_L(0, k) = \phi_L(i, 0) = 0$  and we use the recursion:

$$\phi_L(i, k) = \begin{cases} \phi_L(i - 1, k), & \text{if } k < \min\{\tau_i, L\} \\ \min_{\ell=1, \dots, \min\{\tau_i, L\}} \phi_L(i - 1, \ell - 1) + d_i + \phi_L(i, k - \ell), & \text{else} \end{cases}$$

The optimal solution is the schedule that corresponds to the value  $L \in \{1, \dots, \tau_n\}$  minimizing  $\phi_L(n, L)/L$ . Note that the algorithm runs in time  $O(n\tau_n^3)$ , implying the theorem.

### 4 Approximation on General Graphs

We will now present logarithmic approximations for both objectives. Note that an  $O(\log \tau_{max})$ -approximation is readily achieved; simply treat the sets of clients with equal turnover time as independent instances. For MIN-AVG, the problem with equal turnover times is simply TSP, for the MIN-MAX we get a problem sometimes called the  $k$ -TSP, for which a  $\frac{5}{2}$  approximation is known [15]. Since by rounding to powers of 2, we ensure there are  $O(\log(\tau_{max}))$  different turnover times, we get Theorem 6.

**Theorem 6.** MIN-MAX and MIN-AVG RFTT have an  $O(\log \tau_{max})$ -approximation.

*Proof.* By Lemma 1 we may assume every  $\tau_i$  is a power of 2 so that there are at most  $\log \tau_{max}$  different turnover times. We simply treat the sets of vertices with the same turnover time as separate instances and concatenate the solutions. Our result then follows from the fact that for all these instances a constant factor approximation is available. In the case of the MIN-MAX objective we get the  $k$ -TSP problem, where  $k$  is equal to the turnover time of the vertices in the instance. In the case of MIN-AVG, we need to minimize the sum over all  $k$  tours. But since all turnover times are equal there is no advantage to visiting vertices on different days, hence we recover a simple TSP problem.

In the case of MIN-MAX it is relatively simple to adapt this idea for an  $O(\log n)$ -approximation by appropriately reassigning clients to lower turnover times, as per Theorem 7.

**Theorem 7.** MIN-MAX RFTT has an  $O(\log n)$ -approximation.

*Proof.* We start by assuming that every turnover time is a power of 2. Next, we split up the instance into two new instances. To this end we first define a turnover time  $k$  to be *saturated* if  $|\{j \in V | \tau_j = k\}| \geq k$ . In the first instance we retain the set of vertices  $V_1$  with saturated turnover times, and in the second all vertices  $V_2$  with unsaturated turnover times. Now if all turnover times are saturated, then  $\tau_{max} = O(n)$  and we can find a  $O(\log n)$ -approximation using Theorem 6. So what remains is to find a  $O(\log n)$ -approximation for the second instance.

Since no turnover time is saturated, it is easy to see that we can partition the vertices in  $V_2$  into  $\lceil \log n \rceil$  sets  $W_1, W_2, W_4, \dots, W_{2^{\lceil \log n \rceil}}$ , such that  $|W_i| \leq i$ , and such that  $\tau_j \geq i$  for all  $j \in W_i$ . For example we could first add all vertices  $j$  with  $\tau_j = i$  to  $W_i$  for  $i \leq \lceil \log n \rceil$ , and then arbitrarily distribute vertices  $j$  with  $\tau_j > \lceil \log n \rceil$  among the sets that have space. We now produce a schedule by visiting all clients in any set  $W_k$  on different days. This is feasible and implies that at most  $\log n$  clients are visited on a given day, which leads to  $O(\log n)$ -approximation factor, as required.

The approach of Theorem 6 does not trivially extend to the MIN-AVG case. However, we may combine our result on trees with the FRT tree embeddings [13], to get a randomized  $O(\log n)$ -approximation.

A more direct, and deterministic  $O(\log n)$ -approximation is possible as well. In particular, we use the simple heuristic of visiting each client on every day that is a multiple of its turnover time, when turnover times are powers of 2. We call such a schedule a *synchronized* solution, and show that gives a logarithmic approximation.

The proof of this approximation factor, which is not trivial, works by show that a synchronized schedule is no more costly than a *non-decreasing* schedule, in which all tours are routed along a tree with turnover times non-decreasing from the root. We then show how to transform any schedule to a non-decreasing one, losing a logarithmic factor in the process. As a byproduct we show that the analysis is tight, and that a non-decreasing schedule must be  $\Omega(\log n)$  times more costly than OPT in the worst case. The deterministic proof of Theorem 8 can be found in the appendix of the full paper [4].

**Theorem 8.** MIN-AVG RFTT has an  $O(\log n)$ -approximation.

*Proof.* We will apply the FRT tree embedding [13] of the initial instance and then use the 2-approximation for tree metrics to obtain the final solution. Given the instance  $(G, \tau)$ , let  $T$  be a random tree produced by the tree metric approximation with  $O(\log n)$  distortion. Then  $d_G(u, v) \leq d_T(u, v)$  and  $E[d_T(u, v)] \leq O(\log n)d_G(u, v)$ . Let  $S$  be the solution produced by the 2-approximation for MIN-AVG RFTT on the tree metric  $T$ . Then  $E[S] \leq 2E[OPT(T, \tau)] \leq O(\log n)E[OPT(G, \tau)]$  by linearity of expectation on the sum over the edges.

It is an open question whether there exists a constant factor approximation algorithm for the general case. We observe that the approach of first finding a tree spanning all vertices and then using the algorithm of Sect. 3 is unsuccessful. In fact there exist instances of the problem on a graph  $G$  with  $n$  vertices, such that if we limit our attention to tours that for each day use only edges of a spanning tree of  $G$  then the obtained solution is  $\Omega(\log n)$  approximated. This implies that we need some new ideas, in order to improve the  $O(\log n)$  approximation of the previous theorem.

## 5 Conclusion

In this paper, we considered replenishment problems with fixed turnover times, a natural inventory-routing problem that has not been studied before. We formally defined the RFTT problem and considered the objectives MIN-AVG and MIN-MAX. For the MIN-AVG RFTT, we showed that it is at least as hard as the intractable PINWHEEL SCHEDULING PROBLEM on series-parallel graphs and we gave a 2-approximation for trees. For the MIN-MAX objective we showed NP-hardness on stars and gave a 6-approximation for tree metrics. We also presented a DP that

solved the MIN-AVG RFTT in pseudopolynomial time on line graphs. Finally, we gave a  $O(\log n)$ -approximation for the MIN-MAX objective on general metrics.

The results that we present should be considered as a first step in this area and many problems remain open. An intriguing open problem is the complexity of the of RFTT on a tree. Namely, for MIN-AVG variant we conjecture that the problem is hard, and we ask whether the simple 2-approximation we provide can be improved. For the MIN-MAX variant it is open whether the problem is APX-hard and whether we can improve the 6-approximation.

Next to replenishing locations with routing aspects as we studied in this paper, scheduling problems modeling maintenance or security control of systems, form a class of problems to which this model naturally applies. It would be interesting to study such fixed turnover time problems in combination with scheduling. Would this combination allow for more definitive results?

**Acknowledgments.** The research of YJ and RR is supported in part by the U. S. National Science Foundation under award numbers CCF-1527032 and CCF-1655442. The research of MvE was done while he was employed by Vrije Universiteit Amsterdam.

## References

1. Baruah, S., Goossens, J.: Scheduling real-time tasks: algorithms and complexity. In: Handbook of Scheduling: Algorithms, Models, and Performance Analysis. CRC Press, Boca Raton (2003)
2. Baruah, S., Rosier, L., Tulchinsky, I., Varvel, D.: The complexity of periodic maintenance. In: Proceedings of the International Computer Symposium, pp. 315–320 (1990)
3. Bonifaci, V., Marchetti-Spaccamela, A.: Feasibility analysis of sporadic real-time multiprocessor task systems. *Algorithmica* **63**(4), 763–780 (2012)
4. Bosman, T., van Ee, M., Jiao, Y., Marchetti-Spaccamela, A., Ravi, R., Stougie, L.: Approximation algorithms for replenishment problems with fixed turnover times. arXiv preprint [arXiv:1712.05218](https://arxiv.org/abs/1712.05218) (2017)
5. Calabro, C., Impagliazzo, R., Kabanets, V., Paturi, R.: The complexity of unique  $k$ -SAT: an isolation lemma for  $k$ -CNFs. *J. Comput. Syst. Sci.* **74**(3), 386–393 (2008)
6. Chan, M.Y., Chin, F.Y.L.: General schedulers for the pinwheel problem based on double-integer reduction. *IEEE Trans. Comput.* **41**(6), 755–768 (1992)
7. Chan, M.Y., Chin, F.Y.L.: Schedulers for larger classes of pinwheel instances. *Algorithmica* **9**(5), 425–462 (1993)
8. Coelho, L.C., Cordeau, J.-F., Laporte, G.: Thirty years of inventory routing. *Transp. Sci.* **48**(1), 1–19 (2013)
9. Coene, S., Spieksma, F.C.R., Woeginger, G.J.: Charlemagne’s challenge: the periodic latency problem. *Oper. Res.* **59**(3), 674–683 (2011)
10. Dell, H., Husfeldt, T., Marx, D., Taslaman, N., Wahlén, M.: Exponential time complexity of the permanent and the tutte polynomial. *ACM Trans. Algorithms* **10**(4), 21:1–21:32 (2014)
11. Eisenbrand, F., Hähnle, N., Niemeier, M., Skutella, M., Verschae, J., Wiese, A.: Scheduling periodic tasks in a hard real-time environment. In: Abramsky, S., Gavioille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010. LNCS, vol. 6198, pp. 299–311. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-14165-2\\_26](https://doi.org/10.1007/978-3-642-14165-2_26)

12. Ekberg, P., Yi, W.: Schedulability analysis of a graph-based task model for mixed-criticality systems. *Real-Time Syst.* **52**(1), 1–37 (2016)
13. Fakcharoenphol, J., Rao, S., Talwar, K.: A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Syst. Sci.* **69**(3), 485–497 (2004)
14. Fishburn, P.C., Lagarias, J.C.: Pinwheel scheduling: achievable densities. *Algorithmica* **34**(1), 14–38 (2002)
15. Frederickson, G.N., Hecht, M.S., Kim, C.E.: Approximation algorithms for some routing problems. In: *Proceedings of the 17th International Symposium on Foundations of Computer Science*, pp. 216–227 (1976)
16. Gąsieniec, L., Klasing, R., Levkopoulos, C., Lingas, A., Min, J., Radzik, T.: Bamboo garden trimming problem (perpetual maintenance of machines with different attendance urgency factors). In: Steffen, B., Baier, C., van den Brand, M., Eder, J., Hinchey, M., Margaria, T. (eds.) *SOFSEM 2017. LNCS*, vol. 10139, pp. 229–240. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-51963-0\\_18](https://doi.org/10.1007/978-3-319-51963-0_18)
17. Holte, R., Mok, A., Rosier, L., Tulchinsky, I., Varvel, D.: The pinwheel: a real-time scheduling problem. In: *Proceedings of the 22th Annual Hawaii International Conference on System Sciences*, vol. 2, pp. 693–702 (1989)
18. Jacobs, T., Longo, S.: A new perspective on the windows scheduling problem. arXiv preprint [arXiv:1410.7237](https://arxiv.org/abs/1410.7237) (2014)
19. Mok, A., Rosier, L., Tulchinsky, I., Varvel, D.: Algorithms and complexity of the periodic maintenance problem. *Microprocess. Microprogr.* **27**(1–5), 657–664 (1989)