Improved Approximation Algorithms for Inventory Problems

Thomas Bosman[1] and Neil Olver[2]*

[1] Booking.com, Amsterdam, The Netherlands. tbosman@gmail.com.
[2] Department of Mathematics, London School of Economics and Political Science, London, UK. N.Olver@lse.ac.uk.

Abstract. We give new approximation algorithms for the submodular joint replenishment problem and the inventory routing problem, using an iterative rounding approach. In both problems, we are given a set of N items and a discrete time horizon of T days in which given demands for the items must be satisfied. Ordering a set of items incurs a cost according to a set function, with properties depending on the problem under consideration. Demand for an item at time t can be satisfied by an order on any day prior to t, but a holding cost is charged for storing the items during the intermediate period; the goal is to minimize the sum of the ordering and holding cost.
Our approximation factor for both problems is $O(\log \log \min(N, T))$; this improves exponentially on the previous best results.

Keywords: approximation algorithms · iterative rounding · inventory.

1 Introduction

The inventory problem studied in this paper captures a number of related models studied in the supply chain literature. One of the simplest is the dynamic economic lot size model [14]. Here we have varying demand for a single item over T time units. Demand at time t or later can be satisfied by an order at time t (but not vice versa). For each day, there is a per-unit cost for holding the items in storage; there is also a fixed setup cost for ordering any positive quantity of the item, which is the same for each day. We want to decide on how many items to order on each day so as to minimize the total ordering and holding cost.

The *joint replenishment problem (JRP)* generalizes this problem to multiple items. We now have a unique holding cost for each day and each item, and a per item setup cost for ordering any quantity of that item. Furthermore, there is a general setup cost for ordering any items at all. This setup cost structure is called *additive*. While having limited expressive power in comparison to some of the more complex generalizations that have been studied, the additive joint replenishment problem is long known to be NP-hard [1]. This problem has attracted considerable attention from the theory community in the past, resulting

* Supported by Dutch Science Foundation (NWO) Vidi grant 016.Vidi.189.087.

in a line of progressively stronger approximation algorithms [12,11], the best of which gives an approximation ratio of 1.791 [2].

A more general version of this problem uses an ordering cost structure in which the setup cost is a submodular function of the items ordered. This model, introduced by Cheung et al. [6], is called the *submodular joint replenishment problem (SJRP)* and captures both the additive cost structure as well as other sensible models. In the same work, the authors give constant factor approximation algorithms for special cases of submodular cost functions, such as tree cost, laminar cost (i.e., coverage functions where the sets form a laminar family) and cardinality cost (where the cost is a concave function of the number of distinct items). For the general case, they provide an $O(\log NT)$ approximation algorithm.

In the *inventory routing problem (IRP)* setup costs are routing costs in a metric space. There is a root node and every item corresponds to a point in the metric. The setup cost for a given item set is then given by the length of the shortest tour visiting the depot and every item in the set. The usual interpretation of the model is that the root node represents a central depot and every other point in the metric represents a warehouse to be supplied from the depot. (To streamline terminology with the joint replenishment problem, we will keep using the term "item" rather than "location".)

The IRP has been extensively studied in the past three decades (see [7] for a recent survey), but primarily from a computational perspective. But very little is known about its approximability. Fukunaga et al. [9] presented a constant factor approximation under the restriction that orders for a given item must be scheduled *periodically*. This restriction appears to significantly simplify the construction of an approximation algorithm, as prior to this work the best known polynomial time approximation algorithms gave (incomparable) $O(\log N)$ [9] and $O(\log T)$ [13] performance guarantees.

Nagarajan and Shi [13] break the logarithmic barrier for both IRP and SJRP, under the condition that holding costs grow as a fixed degree monomial. This is a very natural restriction; in particular it captures the case where holding an item incurs a fixed rate per unit per day, depending only on the item. Building on their approach, we improve exponentially on their $O(\log T/\log\log T)$ approximation factor. We also provide some general techniques to turn (sub)logarithmic approximation algorithms in terms of T into equivalent algorithms in terms of N; and we are able to obtain results without restriction on the holding costs. Our main contributions are summarized in the following theorems.

Theorem 1. *There is a polynomial time $O(\log\log\min(N,T))$-approximation algorithm for the inventory routing problem.*

Theorem 2. *There is a polynomial time $O(\log\log\min(N,T))$-approximation algorithm for the submodular joint replenishment problem.*

We also mention the works on submodular partitioning problems [4,5,8]. In these problems, a ground set V must be partitioned across k different sets to minimize a submodular cost function. They use rounding of a relaxation based on the Lovász extension to unify and improve several prior results. Their approach inspired our use of the Lovász extension in the rounding algorithm for SJRP.

2 Preliminaries, model and technical overview

We use log for the base 2 logarithm. We write $[k]$ for $\{1, \dots, k\}$, and $[k, \ell]$ for $\{k, k+1, \dots, \ell\}$, for any integers $k \leq \ell$.

The general framework of the inventory problems we investigate is defined by a set of items V of size N, ordering cost function $f : 2^V \to \mathbb{R}_{\geq 0}$ and a time horizon $[T] = \{1, \dots, T\}$. We will assume throughout this paper that f is monotone and subadditive, with $f(\emptyset) = 0$. We will typically refer to the atomic time units as *days*.

For each item $v \in V$ and day t, there is a demand $d_{vt} \geq 0$. The collection of item-day pairs for which there is positive demand is denoted $D := \{(v,t) : d_{vt} > 0\}$. Demand for day t can be satisfied on or before day t. If we satisfy demand for item i on day t using an order on some day $s < t$, we need to store the items in the intervening days, and we pay a holding cost of h^v_{st} per unit we store. The magnitude of the demand only plays a role in the holding cost; the ordering cost is determined by the unique items ordered and is independent of how many units are ordered.

Given these inputs, we need to place an order for items to be delivered on each day so as to minimize the total ordering cost plus the holding cost. Since the cost of delivering an item does not depend on the size of the order and we want to store items as briefly as possible, it is always optimal to deliver just enough units of an item to satisfy demand until the next order for that item is scheduled. Hence, once we decide which items to order on which days, the optimal schedule is completely determined.

The inventory routing problem is the special case where we have a metric on V, some distinguished root node $r \in V$, and the ordering cost $f(S)$ of a set $S \subseteq V$ is the minimum possible length of a tree containing $S \cup \{r\}$. Here we differ from the usual definition, where $f(S)$ is defined to be the length of a shortest tour on $S \cup \{r\}$; but as is well known, these two definitions differ only by a factor of at most 2, which will not concern us. The submodular joint replenishment problem is the special case where f is submodular (in addition to the required properties already listed).

An integer programming formulation for this problem is given in (1). Here the variable y^S_t indicates whether item set S is ordered on day t, and x^v_{st} indicates whether the demand for item v on day t is satisfied by an order on day s.

$$\begin{array}{ll} \text{minimize} & \sum_{t\in[T]} \sum_{S\subseteq V} f(S) y^S_t + \sum_{t\in[T]} \sum_{v\in V} \sum_{s\leq t} d_{vt} h^v_{st} x^v_{st} \\ \text{subject to} & x^v_{st} \leq \sum_{S: v\in S} y^S_s \qquad \forall (v,t) \in D, s \leq t \\ & \sum_{s\leq t} x^v_{st} = 1 \qquad \forall (v,t) \in D \\ & y^S_t, x^v_{st} \in \{0,1\} \qquad \forall v \in V, s \leq t, S \subseteq V \end{array} \tag{1}$$

Let (LP) denote the LP relaxation obtained by replacing the integrality constraints of ILP (1) by nonnegativity constraints; this LP has an exponential number of variables. To efficiently solve (LP), it suffices to provide an efficient separation oracle for the dual. This can be done for both SJRP and IRP (in the latter case, in an approximate sense); see [13].

Nagarajan and Shi [13] show that in order to round (LP), it suffices to round an associated covering problem. Essentially, given a solution $(\hat{x}, \hat{y})$ to (LP), we require that each demand $(v, t) \in D$ is served within an interval $[s'(v, t), t]$, where $s'(v, t)$ is the median of the distribution $(\hat{x}^v_{st})_{s \leq t}$. Serving (v, t) anywhere within this interval will incur cost at most twice what the fractional solution pays; and moreover, they show that enforcing this restriction cannot make the optimal solution more than a constant factor more expensive. The holding costs can then be dropped from the objective function. All in all, we obtain an instance of the following *subadditive cover over time* problem: for each item $v \in V$ we are given an associated set of *demand windows* $\mathcal{W}_v \subseteq \{[s, t] : s \leq t \in [T]\}$. We must choose a subset $S_t \subseteq V$ for each day $t \in [T]$ such that every item $v \in V$ is covered in each of its demand windows—that is, $v \in S_r$ for some $r \in [s, t]$, for each $[s, t] \in \mathcal{W}_v$. The goal is to find a feasible solution minimizing the total cost $\sum_{t \in [T]} f(S_t)$.

We also associate the canonical LP (2) with the subadditive cover over time problem.

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_{t \in [T]} \sum_{S \subseteq V} f(S) y^S_t \\ \text{subject to} \quad & \textstyle\sum_{r \in [s,t]} \sum_{S : v \in S} y^S_r \geq 1 \qquad \forall [s, t] \in \mathcal{W}_v, \quad \forall v \in V \\ & y^S_t \geq 0 \qquad \forall t \in [T], S \subseteq V \end{aligned} \tag{2}$$

Our goal, given a fractional solution to this LP, is to round it to an integral solution. Note that the instance of subadditive cover over time is constructed from a solution $(\hat{x}, \hat{y})$ to (LP) in such a way that $\hat{y}$ is already a feasible fractional solution to (2).

We now come to our first contribution. We show that this reduction can be taken much further: we can reduce to covering problems where the set of intervals have a very special structure. This structure, which we call *left aligned*, shares many of the benefits of a laminar family. For example, they have a natural notion of depth, which is always logarithmically bounded by T; the approximation factors of our algorithms are essentially logarithmic in this depth. We describe this reduction, which is rather general and applies identically to both SJRP and IRP, in Section 3. We also show, again generally, how the time horizon T can be polynomially bounded in terms of the number of items N.

So to obtain our main theorems, it suffices to give $O(\log \log T)$-factor approximation algorithms for these well-structured covering variants of SJRP and IRP. Here the approaches diverge; we give rather different algorithms for these two problems, albeit based on iterative rounding [10] in both cases.

For IRP, the algorithm uses randomized iterative rounding [3] of a certain path-based relaxation. We can show that after sampling every path in the support of the relaxation $O(\log \log T)$ times, we can remove a constant fraction of the edges and reorder the remaining paths such that we retain a feasible solution. Details are given in Section 4.

For SJRP, by contrast, the iterative rounding approach is naturally deterministic in nature. Instead of randomly rounding item sets in the support of a relaxation, we carefully try to pick a set for each day such that we win a constant fraction of its cost back in the subsequent reduction of the cost of the relaxation. If such a set cannot be found, we show that we can shrink the time horizon T by merging some adjacent time units (or put differently, we are able to remove the bottom "leaf" layer of the left-aligned family); we can then recurse on the smaller instance. We discuss this further in Section 5.

3 Reducing to structured covering problems

The results of this section will not use any properties of the ordering cost function f that differ between IRP and SJRP. All that we will need, other than the general properties of f assumed at the start of Section 2, is that we are given an (approximately) optimal solution to the LP relaxation of (1).

Let $\mathcal{D} = \{[k2^i+1, (k+1)2^i] : i, k \in \mathbb{Z}_{\geq 0}\}$ denote the family of dyadic intervals over the nonnegative integers; the value of i for one of these intervals in $\mathcal{D}$ we call the *level* of that interval.

Definition 1. A family of intervals $\mathcal{F} \subseteq \{[s,t] : s, t \in \mathbb{Z}_{\geq 0}\}$ is called

- *left aligned* if for all $[s,t] \in \mathcal{F}$ there exists $[s,t'] \in \mathcal{D}$ with $t' \geq t$,
- *right aligned* if for all $[s,t] \in \mathcal{F}$ there exists $[s',t] \in \mathcal{D}$ with $s' \leq s$.

The *level* of an interval $[s,t] \in \mathcal{F}$ is the level of the minimal interval of $\mathcal{D}$ containing $[s,t]$.

We will call an instance of subadditive cover over time *left (right) aligned* if $\bigcup_{v \in V} \mathcal{W}_v$ is left (right) aligned.

Theorem 3. *At the loss of a constant factor, we can reduce an instance of the subadditive cover over time problem to a pair of instances, one left aligned and the other right aligned.*

The proof is given in the appendix. Right-aligned instances can be handled identically to left-aligned ones (by simply reversing the time indexing in the instance), so we consider only left-aligned instances in the sequel.

3.1 Bounding the time horizon, and further simplifications

Due to space constraints, we defer the proof of the following to the full version.

Theorem 4. *At the loss of a constant factor we can reduce a left-aligned subadditive cover over time problem to a polynomial-sized collection of left-aligned subadditive cover over time problems with time horizons equal to N^2.*

This theorem could already be used to improve the dependence of the Nagarajan-Shi algorithm from $O\left(\frac{\log T}{\log\log T}\right)$ to $O\left(\frac{\log \min\{T,N\}}{\log\log \min\{T,N\}}\right)$. It also allows us to make the simplifying assumption that each item has positive demand on exactly one day, by making a copy of an item for each day in which it has a positive demand (with $T \le N^2$, this only increases the number of items by a polynomial factor). Finally, we also assume that $T = 2^{2^k}$ for some $k \in \mathbb{N}$; if not, simply round up. Call an instance with all these properties (including being left aligned) *nice*; we will assume throughout the remainder of the paper that we are working with a nice instance.

4 Steiner tree over time

So in order to prove Theorem 1, we need to give an $O(\log\log T)$-approximation algorithm to nice instances of subadditive cover over time, for the appropriate class of order functions f. More precisely, V is the set of nodes of a semimetric space with distance function $c : V \times V \to \mathbb{R}_{\ge 0}$; a root node $r \in V$ is specified, and for all other nodes $v \in V \setminus \{r\}$, a time window $F_v = [a_v, b_v] \subseteq [T]$ is given. Since $f(S)$ denotes the cost of a cheapest tree containing $S \cup \{r\}$, we will consider a solution to be described by a collection of trees $(\mathcal{T}_t)_{t\in T}$, all containing the root. To be feasible, each node $v \ne r$ must be contained in $\mathcal{T}_t$ for some $t \in F_v$. The cost of a tree $\mathcal{T}$ (i.e., the sum of the length of its edges) is denoted $c(\mathcal{T})$; the objective is to minimize the total cost $\sum_t c(\mathcal{T}_t)$. We will call this the *Steiner tree over time problem.*

The main part of our result works by iteratively massaging a specific type of fractional solution until it becomes integral. We now describe this type of solution.

We let $\mathcal{P}$ denote the collection of directed paths in V. For each such directed path $P \in \mathcal{P}$, let $P \odot_t v$ signify that P connects v to $\mathcal{T}_t$, i.e., $P \odot_t v$ if there is a directed subpath on P from v to a node in the tree $\mathcal{T}_t$ containing the root on day t. If $v \in \mathcal{T}_t$ we let $P \odot_t v$ hold for all P by convention.

Definition 2. A *fractional path solution* (FPS) is a pair $(\mathcal{T}, w)$, giving for each day t a tree $\mathcal{T}_t$ rooted at r and weights $w_t : \mathcal{P} \to [0,1]$, with the property that

$$\textstyle\sum_{t\in F_v} \sum_{P\in\mathcal{P}: P\odot_t v} w_t(P) \ge 1 \qquad \forall v \in V \setminus \{r\}.$$

The cost of the fractional path solution is given by the sum of the cost of the trees and the cost of the paths weighted by w:

$$\textstyle\sum_{t\in[T]} \sum_{P\in\mathcal{P}} w_t(P)c(P) + \sum_{t\in[T]} c(\mathcal{T}_t).$$

Note that a fractional path solution with $w_t(P) = 0$ for all P and t corresponds to a feasible integral solution to the Steiner tree over time problem. Moreover, we can start with a solution y to (2) and convert it to a fractional path solution at the loss of a constant (or more directly, we can solve a compact LP corresponding to fractional path solution). To do this, start by initializing all trees $\mathcal{T}_t$ to contain only the root. Then for each day t and set S in $\mathrm{supp}(y_t)$, construct a minimum spanning tree on $S \cup \{r\}$, and use this to define a path P to r that contains S and has cost at most twice the cost of this spanning tree (simply shortcut the doubled tree). Add P to the solution with weight $w_t(P) = y_t^S$.

Hence, we focus on turning a fractional path solution into one where all path weights are zero, without losing too much in the cost of the solution.

We need some preliminary notation and definitions. Given a directed path P, we use $V(P)$ and $E(P)$ to denote the node set and edge set, respectively. We similarly define $V(\mathcal{T})$ and $E(\mathcal{T})$ for a tree $\mathcal{T}$. The head and tail of P are denoted $\mathrm{head}(P)$ and $\mathrm{tail}(P)$ respectively. Given a tree $\mathcal{T}$ and path P with $\mathrm{head}(P)$ in $\mathcal{T}$, *adding* P to $\mathcal{T}$ (which we may write as $\mathcal{T} + P$) results in a spanning tree of the union of $\mathcal{T}$ and P. In particular, $\mathcal{T} + P$ is a tree, costing no more than $c(\mathcal{T}) + c(P)$, and spanning $V(\mathcal{T}) \cup V(P)$. We associate with each node v a *level* $\ell(v)$ in the natural way, namely as the level of F_v in the left aligned family $\bigcup_{v \in V \setminus \{r\}} F_v$.

The rounding algorithm will consist of a number of iterations. Each iteration will increase the size of the integral part $(\mathcal{T}_t)_{t \in [T]}$, while reducing the size of the fractional part $(w_t)_{t \in [T]}$, until the solution is entirely integral. We will ensure that the cost increase in the integral part is an $O(\log\log T)$ factor times the cost decrease in the fractional part, which clearly yields the required approximation guarantee.

Each iteration of the rounding scheme involves two steps. The first step is, for each $t \in [T]$, to independently sample the paths according to the weights $w_t(P)$, upscaled by a factor $K \log\log T$; K is a fixed constant we will choose later. These paths are added (one by one) to $\mathcal{T}_t$. The total cost increase due to this step is $O(C \log\log T)$, where $C = \sum_{t \in [T]} \sum_{P \in \mathcal{P}} w_t(P) c(P)$ is the total cost of the fractional part. Our goal will now be to adjust the fractional paths in a way that reduces the total fractional cost by a constant factor with high probability, while maintaining feasibility. This will clearly lead to our desired approximation ratio: each iteration we pay $O(\log\log T)$ times the decrease in the total fractional cost, and once the total fractional cost reaches zero, we have an integral solution.

The main operation that the algorithm will perform in order to achieve this is a "split and shift" operation. Let $(\mathcal{T}, w)$ be the fractional path solution after the above sampling step. Let P be some path in $\mathrm{supp}(w_t)$. Our goal will be to

– **(split)** remove some edges from P, resulting in a collection of subpaths $P_1, P_2, \ldots, P_q$, which may *not* have their heads in $\mathcal{T}_t$; and then
– **(shift)** for each one of these paths P_j, assign its weight to some day t_j, in such a way that now $\mathrm{head}(P_j)$ is in $\mathcal{T}_{t_j}$, and t_j is still in the time windows of all the nodes in P_j.

This would ensure feasibility, while reducing the fractional cost by $w(P)$ times the total cost of the removed edges. If each edge is removed with constant probability, we obtain the required cost decrease.

In order to make this work, we need some control of the interaction of the different time windows of the nodes on a given path. The most important fact, immediate from the left aligned structure, is the following.

Lemma 1. *If the time windows of v and w overlap, with $\ell(w) \geq \ell(v)$, then $a_w \leq a_v$.*

This means that if we consider a path P' with head v' (which might be a subpath of a path in $\operatorname{supp}(w_t)$ say) where $\ell(v')$ is minimal amongst all nodes in P', then any $t' \in F_{v'}$ with $t' \leq t$ will be in F_v for all $v \in V(P')$.

The following definition will aid us in shifting always to earlier days, so that the above can be applied.

Definition 3. Given a fractional path solution $(\mathcal{T}, w)$, for each $v \in V \setminus \{r\}$, let m_v be maximal such that

$$\textstyle\sum_{t=m_v}^{b_v} \sum_{P \in \mathcal{P}: P \odot_t v} w_t(P) \geq \frac{1}{2}.$$

Then for any $v \in V$, we call $[a_v, m_v]$ the *sow* phase of v and $[m_v, b_v]$ the *reap* phase of v. *(Note that the sow and reap phases both contain m_v.)*

Let S_v and R_v denote the sow and reap phases of $v \in V$, at the start of this iteration. We first double all weights; let $w' = 2w$. This ensures that

$$\sum_{t \in R_v} \sum_{P \in \mathcal{P}: P \odot_t v} w'_t(P) \geq 1 \quad \text{and} \quad \sum_{t \in S_v} \sum_{P \in \mathcal{P}: P \odot_t v} w'_t(P) \geq 1. \tag{3}$$

Our goal will be to show that every edge on a path can be removed with probability $\frac{3}{4}$; this counteracts the doubling of the weights and ensures that the expected fractional cost at the end of the iteration is at most half its initial value.

Consider each day $t \in [T]$ and path $P \in \operatorname{supp}(w'_t)$ separately. Let us say that a node $v \in V(P)$ is *serviced* by P if $t \in R_v$. Assume that $\operatorname{tail}(P)$ is serviced by P; otherwise replace P in w'_t by the subpath of P from the first serviced node, retaining feasibility by (3). We say that a node v serviced by P has *germinated* if it lies in $V(\mathcal{T}_t)$ for some $t \in S_v$. Let $v_1, v_2, \ldots, v_k$ be the nodes serviced by P, in order from the tail to the head of P; so $v_1 = \operatorname{tail}(P)$ and $v_k = \operatorname{head}(P)$. We consider each node v_j (with $j < k$) in turn, and check if (i) v_j has germinated, and (ii) $\ell(v_j)$ is minimal amongst $\ell(v_1), \ell(v_2), \ldots, \ell(v_j)$. If this is the case, we proceed as follows:

- Let $P^{(1)}$, $P^{(2)}$ and $P^{(3)}$ be the subpaths of P from $\operatorname{tail}(P)$ to v_j, from v_j to v_{j+1}, and from v_{j+1} to $\operatorname{head}(P)$, respectively.
- Shift $P^{(1)}$ to a day t_j witnessing that v_j germinated, and delete all edges of $P^{(2)}$. Thus, we modify w' by increasing both $w'_{t_j}(P^{(1)})$ and $w'_t(P^{(3)})$ by $w'_t(P)$, and then setting $w'_t(P)$ to zero.

Since $t_j \in S_{v_j}$, and $t \in R_{v_j}$, we know that $t_j \leq t$; thus by the condition (ii) and Lemma 1, t_j lies in the sow phases of $v_1, v_2, \ldots, v_j$. Thus, this modification retains feasibility of $(\mathcal{T}, w')$. We then repeat this process, continuing with path $P^{(3)}$ instead of P (resulting in a new sequence of nodes serviced by $P^{(3)}$; $v_1, \dots, v_j$ will not be part of this sequence). The iteration ends when this process has been completed for all fractional paths on all days.

As observed, $(\mathcal{T}, w')$ is still feasible at the end of this process. All that remains is to show that indeed each edge is removed with the desired $\frac{3}{4}$ probability.

So fix a day $t \in [T]$ and a path $P \in \text{supp}(w_t)$ (thus, a path in the support of the solution before this process began). As before, let $v_1, \dots, v_k$ be the ordered sequence of nodes serviced by P. Fix now also an edge $e \in E(P)$, and let v_j be the last node in the sequence that lies in the subpath of P from the tail of P to the tail of e. We have the following characterization of when e can be deleted.

Lemma 2. *Edge e will be deleted in the described split-and-shift procedure if the following condition holds:*

For each layer i, the last (furthest from tail(P)*) node from $v_1, \dots, v_j$ with layer at most i has germinated.*

The proof can be found in the appendix.

The probability that e is deleted is now easily controlled. Fix a level i, and consider the last node u from $v_1, \dots, v_j$ of level at most i. Since

$$\sum_{t \in S_u} \sum_{P \in \mathcal{P} : P \odot_t u} w_t(P) \geq \tfrac{1}{2},$$

standard calculations imply that the probability that u has not germinated is at most $e^{-K \log\log T/2} = \epsilon / \log T$, where $\epsilon = \log(-K/2)$. A union bound over all levels, and an appropriate choice of K, gives us the desired result (with high probability).

To complete the proof of Theorem 1, we should observe that the number of iterations is polynomial (in expectation). This is fairly clear, and we omit the details in this extended abstract.

5 Submodular cover over time

Here we consider the subadditive cover over time problem where in addition to the previously required properties (in particular, that f is monotone with $f(\emptyset) = 0$), f is submodular. We assume throughout that we have a nice instance, and use F_v to denote the single time window for $v \in V$.

First, we observe that the LP relaxation (2) has an equivalent convex formulation in terms of the Lovász extension $\hat{f}$ of f.

$$\begin{array}{lll} \min & \sum_{t \in [T]} \hat{f}(x^t) & \\ \text{s.t.} & \sum_{t \in F_v} x_v^t = 1 & \forall v \in V \\ & x \geq 0 & \end{array} \tag{4}$$

For $x \in [0,1]^V$ and $\theta \in [0,1]$, we define the *level set* $L_\theta(x) = \{v \in V : x_v \geq \theta\}$. Then $\hat{f}(x) = \mathbb{E}_\theta[f(L_\theta(x))]$, where $\theta \sim \text{Uniform}(0,1)$. Define the truncation $x^{|\theta}$ by $x_v^{|\theta} = \min\{x_v, \theta\}$.

Definition 4. Given $\theta \in [0,1]$ and $x \in [0,1]^V$, we say that the set $L_\theta(x)$ is *α-supported* if:

$$\hat{f}(x) - \hat{f}(x^{|\theta}) \geq \alpha f(L_\theta(x)). \tag{5}$$

We will provide some intuition for this definition later, but first we describe the algorithm.

Input: A solution x to (4).
Output: A solution $(S_t)_{t\in[T]}$ to the submodular cover over time problem.

```
1: $S_t \leftarrow \emptyset$ for all $t \in [T]$.
2: for $i = 1, \ldots, \log T$ do
3:     for $t \in [T]$ do
4:         if there exists $\theta \in [0,1]$ such that $L_\theta(x^t)$ is $\frac{1}{32 \log\log T}$-supported then
5:             Choose such a $\theta$ and set $S_t \leftarrow S_t \cup L_\theta(x^t)$, $x^t \leftarrow x^{t|\theta}$.
6:         else
7:             $S_t \leftarrow S_t \cup L_1(x^t)$.
8:     Merge time periods by setting, for all $t \in \{k2^i : k = 0,1,2,\ldots\}$,
           $x^{t+1} \leftarrow x^{t+1} + x^{t+2^{i-1}+1}$ and $x^{t+2^{i-1}+1} \leftarrow 0$.
9: return $(S_t)_{t\in[T]}$.
```

Feasibility. The only steps where the coverage $\sum_{t\in F_v} x_v^t$ for an item v could possibly decrease are steps 5 and 8. In step 5, v is added to S^t, so this is clearly no problem. In step 8, we are shifting weight from some time $t+2^{i-1}+1$ to $t+1$. This cannot leave the time window of v unless $\ell(v) < i$. But if v has not been covered by the end of iteration $\ell(v)$, all of the fractional coverage for v will have been merged into the left endpoint of its time window $t' = \min F_v$. This means that $L_\theta(x^{t'})$ contains v for any θ, ensuring v will be added to $S_{t'}$ in step 3 of iteration $\ell(v)+1$.

Cost analysis. This is where the key insights lie. The main driver is the following technical lemma(the proof is postponed to the appendix).

Lemma 3. *For any $x \in [0,1]^V$ and $\alpha \in (0,1]$, either there exists $\theta \in [0,1]$ such that $L_\theta(x)$ is $\frac{\alpha}{32}$-supported, or otherwise $2^{1/\alpha} f(L_1(x)) \leq \hat{f}(x)$.*

The intuition for this lemma is that if no θ with the desired property exists, it can be shown that $f(L_\theta(x))$ decreases quickly everywhere, and consequently $f(L_1(x))$ is small compared to $\hat{f}(x) = \mathbb{E}_\theta[f(L_\theta(x))]$.

Consider now some iteration i of the algorithm, and a particular choice of t. If step 5 is executed, then $\hat{f}(x^t)$ decreases by at least a $\frac{1}{32\log\log T}$ fraction of $f(L_\theta(x^t))$, which is an upper bound on the cost increase of the current solution by subadditivity. Otherwise, by Lemma 3 (with $\alpha = \frac{1}{\log\log T}$), $f(L_1(x^t)) \leq \hat{f}(x^t)/\log T$. The total cost of sets chosen in step 7 in a single iteration is thus at most $\sum_{t\in[T]} \hat{f}(x^t)/\log T$. So over all $\log T$ iterations, this incurs a total cost not more than the original objective value of the relaxation.

Again, to complete the proof of Theorem 2, we should argue that the algorithm runs in polynomial time; we postpone the straightforward details to the full version.

References

1. E. Arkin, D. Joneja, and R. Roundy. Computational complexity of uncapacitated multi-echelon production planning problems. *Operations Research Letters*, 8(2):61–66, 1989.
2. M. Bienkowski, J. Byrka, M. Chrobak, Ł. Jeż, D. Nogneng, and J. Sgall. Better approximation bounds for the joint replenishment problem. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 42–54, 2014.
3. J. Byrka, F. Grandoni, T. Rothvoß, and L. Sanità. Steiner tree approximation via iterative randomized rounding. *Journal of the ACM*, 60(1):6:1–6:33, 2013.
4. C. Chekuri and A. Ene. Approximation algorithms for submodular multiway partition. In *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 807–816, 2011.
5. C. Chekuri and A. Ene. Submodular cost allocation problem and applications. In *Proceedings of the International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 354–366, 2011.
6. M. Cheung, A. N. Elmachtoub, R. Levi, and D. B. Shmoys. The submodular joint replenishment problem. *Mathematical Programming*, 158(1-2):207–233, 2016.
7. L. C. Coelho, J.-F. Cordeau, and G. Laporte. Thirty years of inventory routing. *Transportation Science*, 48(1):1–19, 2013.
8. A. Ene, J. Vondrák, and Y. Wu. Local distribution and the symmetry gap: Approximability of multiway partitioning problems. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 306–325, 2013.
9. T. Fukunaga, A. Nikzad, and R. Ravi. Deliver or hold: approximation algorithms for the periodic inventory routing problem. In *Proceedings of APPROX/RANDOM*, 2014.
10. K. Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica*, 21(1):39–60, 2001.
11. R. Levi, R. Roundy, D. Shmoys, and M. Sviridenko. A constant approximation algorithm for the one-warehouse multiretailer problem. *Management Science*, 54(4):763–776, 2008.
12. R. Levi, R. O. Roundy, and D. B. Shmoys. Primal-dual algorithms for deterministic inventory problems. *Mathematics of Operations Research*, 31(2):267–284, 2006.
13. V. Nagarajan and C. Shi. Approximation algorithms for inventory problems with submodular or routing costs. *Mathematical Programming*, 160(1-2):225–244, 2016.
14. H. M. Wagner and T. M. Whitin. Dynamic version of the economic lot size model. *Management Science*, 5(1):89–96, 1958.

A Some omitted proofs

Proof (Theorem 3). Let y be a solution to (2). We will first generate two new instances of the subadditive cover over time problem, one being left aligned and the other right aligned.

Given an interval $[s, t]$, define the *right-aligned part* $R([s,t])$ and the *left-aligned part* $L([s,t])$ by

$$R([s,t]) = [s, k2^i] \quad \text{and} \quad L([s,t]) = [k2^i + 1, t],$$

where i, k are integers such that $k2^i \in [s,t]$ and i is maximal. If $k2^i = t$, then $L([s,t]) = \emptyset$, and if $k2^i + 1 = s$ then $R([s,t]) = \emptyset$ by convention. It is clear from this definition that $\{L([s,t]) : v \in V, [s,t] \in \mathcal{W}_v\}$ forms a left-aligned family, and similarly the right-aligned parts form a right-aligned family.

Any LP solution must cover every item by at least half in either the right-aligned or left-aligned part of its demand window. For each $v \in V$ and demand window $[s,t] \in \mathcal{W}_v$, if $L([s,t])$ receives half a unit of coverage under y, add $L([s,t])$ as a time window for v in the left-aligned instance; otherwise put $R([s,t])$ in the right-aligned instance.

It is immediate from the way in which we constructed the two instances that $2y$ is a feasible solution to each. Hence the combined cost of the optimal solutions to the LP relaxations of the generated instances is at most 4 times that of the original instance. Furthermore, we can translate integral solutions to the left and right aligned instances back to one for the original instance by adding them together, which does not increase the cost by subadditivity of f. □

Proof (Lemma 2). We proceed by induction on j, the number of serviced nodes on the subpath of P from the tail of P until before edge e. The claim is clearly true if $j = 1$, since the condition ensures that v_1 germinated, in which case all edges from v_1 to v_2 will be deleted by the procedure. (The claim is trivial if $j = 0$, in which case edge e is always deleted).

Suppose $j > 1$, and that that the condition holds. First, by considering level $\ell(v_j)$, it follows that v_j germinated. Next, consider level $i = \ell(v_j) - 1$. If none of $v_1, v_2, \ldots, v_j$ have level i or less, then the procedure will clearly remove the edges between v_j and v_{j+1}, irrespective of what edges have already been removed from P. Otherwise, let q be chosen maximally from $\{1, 2, \ldots, j-1\}$ so that $\ell(v_q) \leq i$. Then the condition of the lemma holds for an edge between v_q and v_{q+1}; hence by our inductive assumption, these edges were removed by the split-and-shift procedure. So at the point that the current edge e is being considered for removal, the subpath of P remaining contains only the services nodes $v_{q+1}, \ldots, v_k$. Since v_j has the smallest level amongst $v_{q+1}, \ldots, v_j$ and has germinated, e will be removed. This completes the induction. □

Proof (Lemma 3). Let $k \in \mathbb{N}$ be such that $\frac{1}{16}\alpha \leq \frac{1}{k} \leq \frac{1}{8}\alpha$. Note that this implies $k \geq 8$. Let $z = f(L_1(x))$.

Claim. *If $2^{1/\alpha} z > \hat{f}(x)$, there exists $m \in \mathbb{N}$ such that*

$$f(L_{\frac{k-m}{k}}(x)) < 2^m z. \tag{6}$$

Before we prove the claim, let's see that it implies the lemma. Suppose that $2^{1/\alpha} f(L_1(x)) > \hat{f}(x)$, since otherwise we are done. The condition of the claim then holds, so take the smallest m that satisfies (6), and let $\theta = \frac{k-m}{k}$. We claim that

$$\frac{\alpha}{32} f(L_\theta(x)) \leq \hat{f}(x) - \hat{f}(x^{|\theta}).$$

To see this we first rewrite the right hand side as an integral.

$$\begin{aligned} \hat{f}(x) - \hat{f}(x^{|\theta}) &= \int_0^1 f(L_\eta(x))\,d\eta - \int_0^1 f(L_\eta(x^{|\theta}))\,d\eta = \qquad (7) \\ &= \int_0^1 f(L_\eta(x))\,d\eta - \int_0^\theta f(L_\eta(x))\,d\eta = \int_\theta^1 f(L_\eta(x))\,d\eta. \end{aligned}$$

Recall that $f(L_\eta(x))$ is monotonically decreasing and that $m \geq 1$ so that $\theta + \frac{1}{k} = \frac{k-m+1}{k} \leq 1$. Then

$$\int_\theta^1 f(L_\eta(x))\,d\eta \geq \int_\theta^{\theta+\frac{1}{k}} f(L_\eta(x))\,d\eta \geq \frac{1}{k} f(L_{\theta+\frac{1}{k}}(x)). \qquad (8)$$

Finally, we use the fact that m is minimal, which implies that $f(L_{\frac{k-m+1}{k}}(x)) \geq 2^{m-1}z$, together with (7) and (8):

$$\hat{f}(x) - \hat{f}(x^{|\theta}) \geq \tfrac{1}{k} 2^{m-1} z = \tfrac{1}{2k} 2^m z > \tfrac{\alpha}{32} f(L_\theta(x)). \qquad (9)$$

In the final inequality of (9) we use that the fact that we chose m to satisfy $2^m z > f(L_{\frac{k-m}{k}}(x))$ and $\frac{1}{k} \geq \frac{1}{16}\alpha$.

Now we proceed to prove the claim. Suppose for contradiction that the condition of the claim holds but no m satisfies inequality (6). Then, in particular it must hold that $f(L_{\frac{1}{k}}(x)) \geq 2^{k-1}z$ and therefore we obtain

$$\hat{f}(x) \geq \int_0^{\frac{1}{k}} f(L_\eta(x))\,d\eta \geq \frac{1}{k} f(L_{\frac{1}{k}}(x)) \geq \frac{1}{k} 2^{k-1} z.$$

Since $k \geq 8$, $\frac{1}{k} 2^{k-1} z \geq 2^{k/2} z$. Since also $\frac{1}{k} \leq \frac{1}{8}\alpha$, we deduce

$$\hat{f}(x) \geq 2^{k/2} z \geq 2^{4/\alpha} z \geq 2^{1/\alpha} z,$$

contradicting that $2^{1/\alpha} z > \hat{f}(x)$. This proves the claim, and hence the lemma. $\square$