A Primal-Dual Approximation Algorithm for the One-Warehouse Multiple-Retailer Problem

Lehilton Lelis Chaves Pedrosa
Institute of Computing, University of Campinas
Av. Albert Einstein, 1251 – Campinas - SP – 13083-852
lehilton@ic.unicamp.br

ABSTRACT

We consider the One-Warehouse Multiple-Retailer Problem (OWMR). In this problem, there is a set of retailers, and a single warehouse. At each period of a planning horizon, a retailer may face a demand for items, that must be satisfied with units currently held in its inventory. A retailer's inventory is replenished by making orders to the warehouse, and the warehouse's inventory is replenished by making orders to an external supplier. The goal is to obtain an inventory policy, such that the total cost of placing orders, and holding items in the stocks is minimized. There exists an LP-rounding approximation for OWMR, but it makes the assumption that warehouse and retailer holding cost functions are dependent. We give a primal-dual approximation for the case with more general and independent holding cost functions. Obtaining an approximation for OWMR using a primal-dual algorithm was an open problem.

KEYWORDS. approximation algorithm, one-warehouse multiple-retailer, primal-dual

Main Area: Combinatorial Optimization (*OC – Otimização Combinatória*)

1. Introduction

We consider the *One-Warehouse Multiple-Retailer Problem* (OWMR). In this problem, the goal is to determine a policy for the inventory of a set of retailers and a single warehouse. The retailers are replenished by making orders to the warehouse, that also maintains its inventory. The warehouse stock must contain at least as many items as there were requested by all the retailer orders made at each period. A stock, either of a retailer or of the warehouse, has no limit on the number of items it can hold at each given moment. Also, an order made by a retailer or by the warehouse has a fixed cost that does not depend on the number of ordered items. What stops retailers from storing large amounts of items in their stocks is the cost of holding items during large periods.

OWMR is a generalization of the widely studied *Joint Replenishment Problem* (JRP). The only difference is that, in JRP, the warehouse may not store items, or, equivalently, the warehouse holding cost is prohibitive. These problems are studied when one wants a permanent inventory policy, or when it is possible to estimate the demands of a planning horizon. In the first case, one is given the estimate rate between the number demanded items per unit of time, and the objective is to define how frequently orders are made (Roundy, 1985). We are interested in the latter case, when there is a discretized set of periods, and the number of demanded items for each retailer and in each time period is known (Levi et al., 2006, 2008; Stauffer et al., 2011).

The holding cost model varies, what may turn problems more or less difficult. Traditionally, the cost to hold one unit of an item can be given as a constant value per period that the unit is held, or as an additive function on the periods the item is held. Levi et al. (2006) considered a more general case, when the holding cost for each retailer demand is a nondecreasing function on the number of periods it is held. For OWMR, Levi et al. (2008) considered a holding cost model such that, for each retailer, the holding cost function is either monotonically nondecreasing or monotonically nonincreasing in the time that an item is delivered to the retailer. For any of such assumptions, it is well known that there are optimal solutions with *zero inventory ordering* (ZIO) policies for JRP (Levi et al., 2006) and for OWRM (Levi et al., 2008). A policy is said to be ZIO if, whenever an order for a retailer or for the warehouse is made, the corresponding inventory level is zero. A consequence is that the problem reduces to determining the time periods at which warehouse or retailer orders are made.

1.1. Related works

JRP has a 2-approximation based on a primal-dual approach by Levi et al. (2006). In this algorithm, the dual variables are increased according to what they called the "wave mechanism", rather than uniformly as in traditional dual ascent algorithms. For the more general OWMR, there is a 1.8-approximation through LP-rounding, based on a new random shift procedure introduced by Levi et al. (2008). They considered the particular case in which the holding cost either decreases, or increases as the fraction of time that the item is held on the warehouse increases. Also, Stauffer et al. (2011) considered a more general holding cost function for OWMR, and obtained a simple 2-approximation algorithm. The currently best known approximation for JRP has factor 1.791 (Bienkowski et al., 2014). For the special case that holding costs are either zero or infinity, that is also known as JRP with deadlines, there is a randomized 1.574-approximation (Bienkowski et al., 2013). An on-line variant of JRP has been considered by Buchbinder et al. (2008), who gave a deterministic 3-competitive algorithm, and showed a lower bound of 2.64 on the competitiveness.

There is an efficient polynomial-time approximation scheme (EPTAS) for JRP with stationary demands and linear holding costs, and a PTAS for the JRP with non-stationary demands, but with soft capacitated single item orders, where each retailer order has a capacity, but several retailer orders at the same time are permitted (Nonner e Sviridenko, 2013). The similar multistage assembly problem, that considers the problem of assembling items according to a bill of materials, and paying echelon holding costs has a 2-approximation based on the primal-dual approach by Levi et al. (2006). They also showed how to solve in polynomial time the particular case of JRP with

only one retailer, called the single-item lot-sizing problem. A capacitated version of the multi-item lot-sizing problem has been considered by Levi et al. (2007), when there is a hard capacity on the total number of items ordered. They showed that this problem is strongly NP-hard, and gave a 2-approximation based on the rounding of an LP relaxation that is similar to that of classical facility location problem.

1.2. Formal definition

In the OWMR problem, there is one warehouse, indexed by number zero, and N retailer locations, that are indexed by integers $1,\ldots,N$. Each retailer i faces a demand of d_{it} units over a finite planning horizon with T time steps $1,\ldots,T$, and such demand may be satisfied only with items that are currently in the retailer inventory. For a pair (i,t), let r,s be such that $r \leq s \leq t$. If one unit of item is ordered at time r in the warehouse, transported to the retailer i at time s, and delivered at time t, then it incurs holding cost h^{it}_{rs}. Every time that the warehouse receives an order, there is a setup ordering cost K_0, that is independent of the number of ordered items. Similarly, every time that a retailer receives the items from the warehouse, there is a setup ordering cost K_i, that is independent of the number of items. The objective is to satisfy all demands, minimizing the overall holding and ordering costs.

1.3. Our results

The 1.8-approximation for OWMR by Levi et al. (2008) assumes that each retailer satisfies one strict monotonicity property. Namely, they assume that each retailer is either a *J-retailer*, or a *W-retailer*. In a J-retailer, if the time that an item leaves the warehouse is delayed, so that the interval it is held in the warehouse stock is extended, then the total holding cost incurred by this item in the warehouse and in the retailer may only increase. In the opposite direction, in a W-retailer, the holding cost incurred by an item may only decrease if the delivery is delayed. Although this generalizes JRP, these restrictions create a dependency between warehouse and retailer holding costs that might be unnatural for many applications. Stauffer et al. (2011) considered a slightly more relaxed holding cost structure, but also with dependent warehouse and retailer costs. We consider a more general holding cost structure, that separates the holding cost incurred by warehouse and that incurred by retailers. We obtain an approximation algorithm based on the primal-dual approach. Obtaining a primal-dual algorithm for OWMR was an open problem (Levi et al., 2008).

In Section 2, we describe formally the holding cost structure. In Section 3, we present the linear programming formulation for OWMR. In Section 4, we describe the primal-dual algorithm. Finally, in Section 5, we show that the algorithm is a 5-approximation.

2. Holding cost model

Consider the demand of retailer i at time t. We refer to this simply as a *demand pair*, and write (i,t). Also, we say that this demand is served by the pair of orders $\lceil r,s \rfloor$ if the units of item that serve such demand were first ordered at time r by the warehouse, later ordered at time s by the retailer, and finally delivered at time t to the client. Each such item has been held in the warehouse inventory during the period $[r,s)$, and in the retailer inventory during the period $[s,t]$. Let h^{it}_{rs} be the total cost of holding one unit of demand (i,t) served by the pair $\lceil r,s \rfloor$. We define the per-unit holding cost function as $h^{it}_{rs} = f^{it}_{rs} + g^{it}_{s}$, where f^{it}_{rs} is the cost of holding this unit at the warehouse, and g^{it}_{s} is the cost of holding this unit at the retailer. The following properties are assumed:

1. If $r = s$, then $f^{it}_{rs} = 0$. This means that the warehouse holding cost is zero if the item was not held in the warehouse inventory.
2. If $[r,s] \subseteq [r',s']$, then $f^{it}_{rs} \leq f^{it}_{r's'}$. This means that storing an item in the warehouse inventory at an earlier time r', or removing it at a later time s' cannot decrease the cost.
3. If $s < s'$, then $g^{it}_{s} \geq g^{it}_{s'}$. This means that storing an item in the retailer inventory at a later time s' cannot increase the cost.

4. Each demand pair (i,t) has one of the following properties:
   - For every r,s,s', such that $r \leq s < s' \leq t$, we have $h^{it}_{rs} \leq h^{it}_{rs'}$. That is, h^{it}_{rs} is monotonically nondecreasing in s. This means that for demand pair (i,t) and a fixed warehouse order r, it is always more economical to hold the items in the retailer inventory, rather than in the warehouse inventory. In this case, we say that (i,t) is a J-demand.
   - For every r,s,s', such that $r \leq s < s' \leq t$, we have $h^{it}_{rs} \geq h^{it}_{rs'}$. That is, h^{it}_{rs} is monotonically nonincreasing in s. This means that for demand pair (i,t) and a fixed warehouse order r, it is always more economical to hold the items in the warehouse inventory, rather than in the retailer inventory. In this case, we say that (i,t) is a W-demand.
   - For every r_1,r_2,r_3, such that $r_1 \leq r_2 \leq r_3 \leq t$, we have $f^{it}_{r_1r_3} \leq f^{it}_{r_1r_2} + f^{it}_{r_2r_3}$. That is, f^{it}_{rs} is subadditive with respect to time intervals $[r_1,r_2)$ and $[r_2,r_3)$. This means that removing an item from the warehouse inventory, and putting it back immediately after cannot decrease the holding cost. In this case, we say that (i,t) is an S-demand.

We notice that the holding cost model studied by Levi et al. (2008) is a particular case of the holding cost defined above. In their setting, they assume that each retailer contains either only W-demands, or only J-demands. Notice that, although property 2 is not explicitly assumed by Levi et al. (2008), this property is implied by their monotonicity assumptions combined with the monge-property for W-demands. Differently from J or W-demands, being an S-demands is a property of function f^{it} only. This allows to model more general problems in which the warehouse and retailer holding cost functions are independent.

3. A linear programming formulation

OWMR admits a natural integer linear program formulation. For retailer i and time period s, we let y_{is} be a binary variable indicating whether an order for retailer i is placed at time s, and let x^{it}_{rs} be a binary variable indicating whether demand pair (i,t) is satisfied by an item ordered at time r at the warehouse, and transported to retailer i at time s. The relaxation is given below. For simplicity, we assume that all demands are unitary, that is, for each pair (i,t), either $d_{it} = 0$, or $d_{it} = 1$. This is without loss of generality, since one can scale the holding cost otherwise.

The linear programming relaxation of formulation is given in the following. The first set of constraints assures that each nonzero demand is satisfied a pair $\lceil r,s \rfloor$. The second and third sets guarantee that, if a demand pair (i,t) is satisfied by a pair $\lceil r,s \rfloor$, then retailer i places an order at time s, and warehouse places an order at time r.

$$
\begin{aligned}
\text{minimize} \quad & \sum_{r=1}^{T} y_{0r}K_0 + \sum_{i=1}^{N}\sum_{s=1}^{T} y_{is}K_i + \sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{r,s:r\leq s\leq t} x^{it}_{rs}h^{it}_{rs} \\
\text{subject to} \quad & \textstyle\sum_{r,s:r\leq s\leq t} x^{it}_{rs} \geq 1 && i \in [N], t \in [T], d_{it} > 0 \\
& \textstyle\sum_{r:r\leq s} x^{it}_{rs} \leq y_{is} && i \in [N], t \in [T], s \in [t] \\
& \textstyle\sum_{s:r\leq s\leq t} x^{it}_{rs} \leq y_{0r} && i \in [N], t \in [T], r \in [t] \\
& x^{it}_{rs}, y_{ir} \geq 0 && i \in [0,N], r \in [T], s \in [r,T], t \in [s,T]
\end{aligned}
$$

The dual program is given next.

$$
\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{N}\sum_{t=1}^{T} b^{it} \\
\text{subject to} \quad & b^{it} \leq h^{it}_{rs} + l^{it}_{s} + z^{it}_{r} && i \in [N], t \in [T], r \in [t], s \in [r,t] \\
& \textstyle\sum_{t=s}^{T} l^{it}_{s} \leq K_i && i \in [N], s \in [T] \\
& \textstyle\sum_{i=1}^{N}\sum_{t=r}^{T} z^{it}_{r} \leq K_0 && r \in [T] \\
& l^{it}_{s}, z^{it}_{r} \geq 0 && i \in [N], r \in [T], s \in [T], t \in [s,T]
\end{aligned}
$$

4. Primal-dual algorithm

Our primal-dual algorithm is inspired in the framework by Levi et al. (2006), based on a "wave" dual ascent algorithm. Rather than increasing dual variables uniformly, they increase the dual variables using the growing rate of the retailer holding cost of each demand, starting with the latest demands, and moving back to the first. However, for the case of OWMR, items can be held in the warehouse inventory, so the wave does not preserve some essential properties in the analysis by Levi et al. (2006). For OWMR, we use a new dual ascent algorithm, and a more complex pruning phase. More specifically, we use the following two-phase algorithm: in the first phase, we generalize the wave algorithm to deal with more complex holding cost models, and, in the second phase, we prune warehouse and retailer orders separately.

We summarize the algorithm below, and describe each step in the following subsections.

1. Run the dual ascent algorithm, and temporarily open warehouse and retailer orders;
2. Prune orders and open permanent orders:
   (a) Permanently open warehouse orders;
   (b) Permanently open retailer orders:
      i. Place a first round of retailer orders;
      ii. Fix unsatisfied demands with additional retailer orders;
3. Connect demands.

4.1. Dual ascent algorithm

We use the following dual ascent mechanism. We consider a notion of a wave. The wave starts at time T and, as time passes, moves backward in time continuously, until it reaches time 0. Initially, all dual variables are set to value zero, and we set the status of each demand pair to *unconnected*. As the wave moves, we increase the values of dual variables for unconnected demands according to certain rules. It might happen that, at some point, the a dual constraints become tight, and so we cannot keep increasing the dual variables without violating the constraints. In such a moment, we say that an event has occurred, and might either change the status of some demand pairs to *connected*, or start increasing other variables.

Denote the current position of the wave by τ. We say that a retailer order (i,s) is paid if the current values of dual variables are such that $\sum_{t=s}^{T} l_s^{it} = K_i$. For a retailer i, let $P_i(\tau)$ be the set of time steps s such that retailer order (i,s) is paid at time τ. At each moment τ, we set the variable b^{it} of each unconnected demand pair (i,t) as

$$b^{it} \leftarrow \min \ \{g_\tau^{it}\} \cup \{f_{\tau s}^{it} + g_s^{it} + l_s^{it} | s \in P_i(\tau)\}. \tag{1}$$

Notice that g_τ^{it} is only defined for integer values of τ. If τ is not integer, we define g_τ^{it} by linearly interpolating $g_{\lfloor\tau\rfloor}^{it}$ and $g_{\lceil\tau\rceil}^{it}$. Similarly, for any integer s, such that $\tau \leq s \leq t$, we define $f_{\tau s}^{it}$ by linearly interpolating $f_{\lfloor\tau\rfloor s}^{it}$ and $f_{\lceil\tau\rceil s}^{it}$. Also, we define $g_0^{it} = \infty$, and $f_{0s}^{it} = \infty$, for every demand pair (i,t) and $s \leq t$.

As the wave moves, we place temporary retailer and warehouse orders. We will say that such orders are "opened". The set of temporarily open warehouse orders is denoted by R, and, for each i, the set of temporarily open retailer orders is denoted by S_i. Initially, sets R and S_i contain no orders. Whenever a demand pair (i,t) is satisfied by a temporarily open pair of retailer and warehouse orders, we connect the demand to the pair of orders, and freeze the corresponding variables b^{it}, l_s^{it}, and z_{rs}^{it} for all r and s.

The algorithm starts at time $\tau = T$ that is decreased continuously until $\tau = 0$. While time passes, some of the following conditions may become satisfied, and the corresponding events are triggered. The events are processed in the order that they happen.

Event 1 For some demand (i,t), such that l_s^{it} has not started increasing, it holds $b^{it} = g_s^{it}$:

- start increasing l_s^{it} at the same rate of b^{it}.

Event 2 For some retailer order (i,s), such that $s \notin S_i$, it holds $\sum_{t=s}^{T} l_s^{it} = K_i$:

- add s do S_i,
- set $open(i,s) \leftarrow \tau$,
- for each $t \geq s$, stop increasing l_s^{it}.

Event 3 For some demand (i,t), retailer order (i,s), and warehouse order $(0,r)$, such that $r \leq s$, $s \in S_i$ and z_r^{it} has not started increasing, it holds $b^{it} = g_s^{it} + f_{rs}^{it} + l_s^{it}$:

- if $r \in R$ (warehouse order $(0,r)$ is temporarily open):
  - connect demand (i,t) to the pair of orders $\lceil r,s \rfloor$,
  - set $freeze(i,t) \leftarrow \tau$;
- if $r \notin R$ (warehouse order $(0,r)$ is not temporarily open):
  - start increasing z_r^{it} at the same rate of b^{it}.

Event 4 For some warehouse order $(0,r)$, such that $r \notin R$, it holds that $\sum_{i=1}^{N} \sum_{t=r}^{T} z_r^{it} = K_0$:

- add r to R,
- set $open(0,r) \leftarrow \tau$,
- for each demand (i,t) such that, for some s, variable z_{rs}^{it} has already started increasing:
  - connect demand (i,t) to the pair of orders $\lceil r,s \rfloor$,
  - set $freeze(i,t) \leftarrow \tau$.

We give the following interpretation for the dual ascent algorithm. For each demand (i,t), variable b^{it} corresponds to its budget. Equation (1) means that, at any moment τ, an unconnected demand (i,t) is willing to pay the minimum cost between the cost of being served by a pair of warehouse and retailer orders at the wave position, or being served by a warehouse order at the wave position and a previously paid retailer order at some time $s \in P_i(\tau)$. We remark that the value of variable b^{it} depends on the value of other dual variables l_s^{it}, for which $s \in P_i(\tau)$. The algorithm is well defined since, at time τ, $P_i(\tau)$ corresponds to the the set S_i, and thus each such a variable has already been frozen at a previous step of the algorithm.

4.2. Pruning phase

The shadow of a given order (i,s) is defined as the interval $[open(i,s),s]$. For any two warehouse orders in R with intersecting shadows, there is some demand pair that contributes to both orders. Therefore, we want to open a subset of permanent warehouse orders R', with no intersecting shadows. This is accomplished by a *pruning* procedure. However, two joint orders with intersecting shadows may include distinct retailer orders, and thus dropping one of such orders may leave demand pairs unsatisfied. For the JRP, the algorithm of Levi et al. (2006) simply moved any retailer order (i,s) from the dropped joint order to the position of the earliest permanent warehouse order that was placed in the shadow of (i,s). This strategy for fixing unsatisfied demand pairs does not work for W-retailers, since, contrary to the case of the JRP, W-retailer orders and warehouse orders are not necessarily synchronized, and thus it may happen that no warehouse order is placed in the retailer shadow.

In our algorithm, we open permanent warehouse orders and retailer orders in two steps. First, we permanently open only warehouse orders, ignoring the retailers, by a simple greedy algorithm to open orders with non-intersecting shadows. This guarantees that each demand pair contributes to at most one warehouse order. Then we open the orders of each retailer using a two-round procedure. In the first round, we permanently open retailer orders with non-intersecting shadows, so that a demand pair does not use the same budget to contribute to several orders. When this first round finishes, each demand that can pay for the holding cost of some permanently open pair of orders is considered satisfied. In the second round, we make sure that any remaining demand becomes satisfied. This is done by greedily placing a retailer order for some unsatisfied demand. By carefully selecting the next unsatisfied demand pair, we guarantee that each demand pays for at most two retailer orders opened in the second round.

(a) Opening permanent warehouse orders:

Let $R = \{r_1, \ldots, r_k\}$, with $r_1 < r_2 < \cdots < r_k$, be the set of all time steps at which a warehouse order was temporarily open. We will return a set R' of permanent warehouse orders. At each moment we keep the last time r' at which we placed a permanent warehouse order. Then we iterate over every $r \in R$, in increasing order of time, and add r to R' if the shadow of r does not intersect the shadow of any order included previously. More precisely, we run the following steps:

1. Set $r' \leftarrow 0$, and $R' \leftarrow \emptyset$.
2. For each $r \leftarrow r_1, \ldots, r_k$, if $open(0, r) > r'$, then
   - set $r' \leftarrow r$, and
   - update $R' \leftarrow R' \cup \{r\}$.

(b) Opening permanent retailer orders:

i. First round of retailer orders:

For each retailer i, we open a first round of retailer orders using the same algorithm used for opening warehouse orders. That is, let $S_i = \{s_1, \ldots, s_k\}$, with $s_1 < s_2 < \cdots < s_k$, be the set of all time steps at which retailer i has placed temporary orders. We obtain a set S'_i of permanent retailer orders, by greedily iterating over $s \in S_i$, in order of time, and adding non-intersecting retailer orders to S'_i.

Before going to the second round, we introduce some notation. For any demand (i,t), let $left(i,t)$ be the minimum time σ such that $b^{it} = g^{it}_{\sigma}$. Notice that σ may be fractional. Intuitively, $left(i,t)$ is the leftmost retailer order by which demand (i,t) may be served paying at most its budget. We use the following definition.

Definition 1. A demand (i,t) is said to be *satisfied* if there is a permanently open pair of orders $\lceil r, r \rfloor$ such that $h^{it}_{rr} \leq b^{it}$, or if there is no open warehouse order $r \in R'$ such that $r \in [left(i,t), t]$.

ii. Fixing unsatisfied demands:

For each retailer i, we open a second round of retailer orders S''_i to fix unsatisfied demands. We fix unsatisfied demands greedily. Let (i,t) be the demand pair with largest $left(i,t)$. Since (i,t) is unsatisfied, we know that there is at least one permanently open warehouse order at $r \in R'$ such that $r \in [left(i,t), t]$. Let r' be the least such time. We include a retailer order r' in S''_i. In this case, we say that demand (i,t) is the *initiator* of order (i,r'). Notice that after permanently opening the retailer order (i,r'), the demand at time t, as well as any unsatisfied demand at time t' with $t' \geq r'$, will become satisfied. We repeat this process until all demand pairs become satisfied. More precisely, for each retailer i we execute the following steps:

1. Set $S_i'' \leftarrow \emptyset$.
2. While there are unsatisfied demand pairs in retailer i:
   (a) let (i,t) be an unsatisfied demand pair with largest $left(i,t)$,
   (b) let $r' \leftarrow \min [left(i,t),t] \cap R'$,
   (c) update $S_i'' \leftarrow S_i'' \cup \{r'\}$.

4.3. Connecting demand pairs

At the last phase of the algorithm, each demand pair (i,t) is simply connected to one pair of orders $\lceil r,s \rfloor$ such that $r \in R'$, $s \in S' \cup S''$, and $r \leq s \leq t$. If there are many pairs that satisfy these properties, we choose the ordering pair with minimum h_{rs}^{it}.

5. Analysis

By the construction of the dual ascent algorithm, the set of variables (b,l,z) is a feasible solution for the dual problem. The incurred costs of the generated solution correspond to the cost of holding items for each demand and the cost of placing permanently open orders: the warehouse orders in R', the first round of retailer orders in S_i' for each i, and the second round of retailer orders in S_i'' for each i. We use the following charging scheme: for each demand pair (i,t) and $r' \in R'$, $z_{r'}^{it}$ is the contribution of (i,t) towards opening warehouse order r'. Similarly, for each $s' \in S_i'$, $l_{s'}^{it}$ is the contributions for retailer order s'.

For retailer orders in S_i'', we use a more complex charging scheme. For a given retailer order (i,s''), with $s'' \in S_i''$, let $(i,\hat{t})$ be the corresponding initiator. Since demand $(i,\hat{t})$ was connected by the dual ascent algorithm, there must exist at least one order in S_i to which $(i,\hat{t})$ can connect. Let $\hat{s}$ be one such order with the largest opening time, that is, an order that was first paid by the dual ascent algorithm. To pay for the order s'', we will use the same budget used to pay for retailer order $\hat{s}$. Notice that here we must use that fact that all retailer orders of item i have the same cost. The order $\hat{s}$ of which the contribution will be used to pay s'' is denoted by $N_i(s'') = \hat{s}$. The contribution of a given demand (i,t) towards s'' is thus $l_{\hat{s}}^{it}$. Finally, each demand pair pays for the holding cost of its own items.

We start with an auxiliary lemma.

Lemma 1. *For any demand pair (i,t), there exists a retailer order (i,s), with $s \in S_i$, such that $left(i,t) \leq open(i,s)$, and $s \leq t$.*

Proof. Let $\tau = freeze(i,t)$, and $\sigma = left(i,t)$. Since demand (i,t) has been connected, we know that there exists a retailer order at time $s \in S_i$ such that $\sigma \leq s$ and $\tau \leq open(i,s)$. Let $\bar{s}$ be the time of a retailer order in S_i such that $\sigma \leq \bar{s} \leq t$ with maximum $open(i,\bar{s})$. For the sake of contradiction, suppose that $open(i,\bar{s}) < \sigma$.

Consider the state of the dual ascent algorithm at the time $\tau' = open(i,\bar{s})$, just before any event is processed, and let b be the budget of (i,t) at such moment. At this moment, we know that $\bar{s} \notin S_i$. Also, according to equation (1), the budget of demand (i,t) is the cost to connect to warehouse order at τ' through some retailer order at s'. Since $\tau' < \sigma$, we obtain $g_{\tau'}^{it} > g_{\sigma}^{it} = b^{it} \geq b$. It follows that s' must be already paid, that is, $s' \in S_i$, and so $open(i,s') > \tau'$. But since $b^{it} \geq b \geq g_{s'}^{it}$, we have $\sigma \leq s'$. This is a contraction to the maximality of $\bar{s}$, and the lemma follows. □

In the following lemmas we bound the contribution of a given demand to each incurred cost of the solution. Each contribution of a demand pair (i,t) is bounded by a factor of its budget b^{it}, so the total contribution is a factor of the dual solution value, and the approximation will follow by the weak duality theorem. First we consider the contribution towards R'.

Lemma 2. *The contribution of demand (i,t) towards opening warehouse orders in R' is at most b^{it}.*

Proof. Let $r \in R$ be a warehouse order to which (i,t) gives a positive contribution, that is, $z_r^{it} > 0$. Since $z_r^{it} > 0$, there is a paid retailer order s at time $\tau = freeze(i,t)$ such that $b^{it} > g_s^{it} + f_{rs}^{it} + l_s^{it}$ by Event 3 of the dual ascent algorithm. Since s is already paid at time τ, we have $s \geq open(i,s) \geq \tau$. Thus we get $s \in P_i(\tau)$, and therefore $b^{it} \leq g_s^{it} + f_{\tau s}^{it} + l_s^{it}$, so $f_{\tau s}^{it} > f_{rs}^{it}$, and, by the monotonicity of f^{it}, this implies that $r \geq \tau$.

Let $r' \in R'$ be the largest time of a permanently open warehouse order such that $z_{r'}^{it} > 0$. We claim that $open(i,r') \leq freeze(i,t)$. Indeed, if at time $open(i,r')$, demand (i,t) is not connected yet, then it would be connected at this pair. It follows from the algorithm to open permanent warehouse orders that $R' \cap [freeze(i,t), r'] \subseteq R' \cap [open(i,r'), r'] = \{r'\}$. Since demand (i,t) does not contribute to warehouse orders $r'' \in R'$ with $r'' < freeze(i,t)$, it follows that (i,t) can only contribute to one warehouse order, namely r'. □

Now, we bound the contribution towards each set of permanently open retailer orders.

Lemma 3. *The contribution of demand (i,t) towards opening retailer orders in S_i' is at most b^{it}.*

Proof. Let $s_1, \ldots, s_k \in S_i'$, with $s_1 < \cdots < s_k$, be the retailer order times to which demand (i,t) gives a positive contribution, that is, for which $l_{s_j}^{it} > 0$ for each $j = 1, \ldots, k$. It follows from the algorithm to open the first round of retailer orders that the shadows of the retailer orders do not intersect, that is, for each $j = 2, \ldots, k$, we have $s_{j-1} < open(i,s_j)$.

Let τ_j be the time at which order (i,s_j) was opened, that is, $\tau_j = open(i,s_j)$, and let $b_{\tau_j}^{it}$ be the budget of demand pair (i,t) at this moment. By equation (1), $b_{\tau_j}^{it} \leq g_{\tau_j}^{it}$. Therefore, the contribution of (i,t) to some retailer order (i,s_j) is $l_{s_j}^{it} \leq b_{\tau_j}^{it} - g_{s_j}^{it} \leq g_{\tau_j}^{it} - g_{s_j}^{it}$. For any $j \geq 2$, we have $g_{\tau_j}^{it} \leq g_{s_{j-1}}^{it}$, since $s_{j-1} < open(i,s_j) = \tau_j$, and thus $l_{s_j}^{it} \leq g_{s_{j-1}}^{it} - g_{s_j}^{it}$. For $j = 1$, trivially $l_{s_1}^{it} \leq b^{it} - g_{s_1}^{it}$. Adding up all contributions, we obtain

$$\sum_{j=1}^{k} l_{s_1}^{it} \leq b^{it} - g_{s_1}^{it} + \sum_{j=2}^{k} (g_{s_{j-1}}^{it} - g_{s_j}^{it}) \leq b^{it}. \qquad \square$$

Afterwards, we use the following definition.

Definition 2. We say that a demand pair (i,t) is served by a twin ordering pair if there is an ordering pair $\lceil r, r \rfloor$ that is permanently open such that $left(i,t) \leq r \leq t$.

Next lemma bounds the contribution of a demand pair towards the retailer orders opened in the second round.

Lemma 4. *The contribution of demand (i,t) towards opening retailer orders in S_i'' is:*

- *at most $2b^{it}$, if (i,t) is served by a twin ordering pair, or*
- *at most b^{it}, if (i,t) is not served by a twin ordering pair.*

Proof. Let $s_1, \ldots, s_k \in S_i''$, with $s_1 < \cdots < s_k$, be the retailer order times to which demand (i,t) gives a positive contribution, that is, for which $l_{\hat{s}_j}^{it} > 0$, where $\hat{s}_j = N_i(s_j)$, for each $j = 1, \ldots, k$. Also, for each $j = 1, \ldots, k$, let (i,t_j) be the initiator of retailer order (i,s_j), and let $\tau_j = left(i,t_j)$.

Recall that, for every retailer order $s \in S_i''$, there is a warehouse order $s \in R'$. Since the algorithm to fix unsatisfied demands chooses the order of R' with minimum time in the interval $[\tau_j, t_j]$, it follows that there is no permanently open warehouse order in the interval $[\tau_j, s_j)$. Formally, for any j, $[\tau_j, s_j) \cap R' = \emptyset$.

We claim that $t_{j-1} < s_j$ for any $j = 2, \ldots, k$. Let $j > 1$. We have $s_{j-1} \in R'$, but $[\tau_j, s_j) \cap R' = \emptyset$, thus $s_{j-1} < \tau_j$, since otherwise we would get $s_{j-1} \geq s_j$. It follows that $\tau_{j-1} < \tau_j$. This

implies that demand pair (i,t_j) was picked before demand pair (i,t_{j-1}) by the algorithm. Consider the execution state of the algorithm just after (i,t_{j-1}) was picked (at step 2(a)). At this moment, we must have $s_j \in S_i''$, since (i,t_j) was picked at a previous iteration. Therefore, at this moment the ordering pair $\lceil s_j, s_j \rfloor$ was permanently open, and thus any demand pair (i,t') with $t' \geq s_j$ should be satisfied. Since (i,t_{j-1}) was picked, it was unsatisfied at this moment, and thus $t_{j-1} < s_j$.

By definition, $\hat{s}_j$ is the retailer order $s \in S_i$ such that $\tau_j \leq s \leq t_j$ with the largest $open(i,s)$. By Lemma 1, we know that there is at least one such s such that $\tau_j \leq open(i,s)$, and thus $\tau_j \leq open(i,\hat{s}_j)$. Let $j > 2$. Recall that $t_{j-2} < s_{j-1}$, and $s_{j-1} < \tau_j$. Hence $open(i,\hat{s}_{j-2}) \leq \hat{s}_{j-2} \leq t_{j-2} < s_{j-1} < \tau_j \leq open(i,\hat{s}_j) \leq \hat{s}_j$. We conclude that the shadows of any two retailer orders $(i,\hat{s}_{j_1})$ and $(i,\hat{s}_{j_2})$ with odd indices j_1 and j_2 do not intersect, that is, $[open(i,\hat{s}_{j_1}),\hat{s}_{j_1}] \cap [open(i,\hat{s}_{j_2}),\hat{s}_{j_2}] = \emptyset$. Using the same arguments of Lemma 3, we conclude that the total contribution of demand pair (i,t) towards opening retailer orders with odd indices is at most b^{it}. Analogously, the total contribution of demand pair (i,t) towards opening retailer orders with even indices is at most b^{it}.

If (i,t) is served by a twin ordering pair, then we are done. So, from now on, assume that this is not the case. Therefore, there is no permanently open pair of orders $\lceil r,r \rfloor$ with $r \in R'$ such that $r \in [left(i,t),t]$. We claim that $k \leq 1$, and thus the lemma will follow. For the sake of contradiction, suppose that $k > 1$. Since $s_k \in R'$, and $s_k \leq t$, we obtain $s_k < left(i,t)$, otherwise we would get $[left(i,t),t] \cap R' \neq \emptyset$. But then $\hat{s}_{k-1} \leq t_{k-1} < s_k < left(i,t)$. However, since (i,t) contributes to $\hat{s}_{k-1}$, we know that $b^{it} > g^{it}_{\hat{s}_{k-1}}$, and thus $left(i,t) < \hat{s}_{k-1}$. This is a contradiction, and we are done also in this case. □

Lemma 5. *Suppose that demand (i,t) is not served by a twin ordering pair. Then the holding cost of demand (i,t) is at most $2b^{it}$.*

Proof. Let $\tau = freeze(i,t)$ and $\sigma = left(i,t)$.

We claim that there is an open warehouse order $r' \in R'$, such that $\tau \leq r' \leq t$. Let $r \in R$ be the warehouse order to which (i,t) was temporarily connected, such that $r \leq t$. Clearly, $open(0,r) \geq \tau$, since the warehouse order was already open when (i,t) was connected. If $r \in R'$, then the claim holds. Otherwise, r was not permanently open by the algorithm, and thus there must exist some $r' \in R' \cap [open(0,r),r]$. Then $\tau \leq open(0,r) \leq r' \leq r \leq t$, and the claim holds also in this case.

Since (i,t) is satisfied (recall Definition 1), and is not served by a twin ordering pair, there is no permanently open warehouse order in the interval $[\sigma,t]$, that is, $[\sigma,t] \cap R' = \emptyset$. It follows that $g^{it}_{r'} > g^{it}_{\sigma} = b^{it}$, and thus $r' < \sigma$. At the freezing time τ, we know by equation (1) that the budget of demand pair (i,t) is such that there exists $s \in S_i$, with $s \leq t$, and $b^{it} = f^{it}_{\tau s} + g^{it}_s + l^{it}_s$. Since $b^{it} \geq g^{it}_s$, we know that $\sigma \leq s$,

We suppose that (i,t) is an S-demand. The other cases are analogous. By construction, there must is an order open in the first round of retailer in the interval $[open(i,s),s]$. Let $s' \in S'_i$ be the largest such order. If $\sigma \leq s'$, then $h^{it}_{r's'} = f^{it}_{r's'} + g^{it}_{s'} \leq f^{it}_{r's} + g^{it}_{\sigma} \leq 2b^{it}$, and we are done. So, we assume that $\sigma > s'$.

Consider the state of the dual ascent algorithm at time $\tau' = s'$, just before any event is processed, and let b be the budget of demand (i,t) at this moment. Clearly, we have $b \leq b^{it}$, because $s' \geq open(i,s) \geq \tau$. Since $s' < \sigma$, we get $b \leq b^{it} = g^{it}_{\sigma} < g^{it}_{s'}$. Thus, from equation (1), for time $\tau' = s'$, there must exist $\bar{s} \in P_i(\tau')$ such that $\bar{s} \leq t$ and $b = f^{it}_{s'\bar{s}} + g^{it}_{\bar{s}} + l^{it}_{\bar{s}}$. Since $\bar{s}$ was already paid at time τ', we get $open(i,\bar{s}) > s'$. Once again, there must exist an open retailer order $\bar{s}' \in S'_i \cap [open(i,\bar{s}),\bar{s}]$. Since $(s',s] \cap S'_i = \emptyset$ by the maximality of s', we conclude that $s < \bar{s}' \leq \bar{s}$.

Now we bound the holding cost of serving demand (i,t) by the ordering pair $\lceil r',\bar{s}' \rfloor$. We get $h^{it}_{r'\bar{s}'} = f^{it}_{r'\bar{s}'} + g^{it}_{\bar{s}'} \leq f^{it}_{\tau\bar{s}} + g^{it}_s \leq f^{it}_{s'\bar{s}} + f^{it}_{\tau s'} + g^{it}_s \leq f^{it}_{s'\bar{s}} + f^{it}_{\tau s} + g^{it}_s \leq b + b^{it} \leq 2b^{it}$, where the second inequality follows from the fact that (i,t) is an S-demand. □

Lemmas 4 and 5 imply the following corollary.

Corollary 1. *The contribution of demand (i,t) towards opening retailer orders in S_i'' plus the holding cost of demand (i,t) is at most $3b^{it}$.*

Proof. If (i,t) is not served by a twin ordering pair, the lemma follows directly from Lemmas 4 and 5. If (i,t) is served by a twin ordering pair $\lceil r,r\rfloor$, then we know that $h^{it}_{rr} = g^{it}_r \leq b^{it}$, since otherwise demand (i,t) would be unsatisfied, and so the corollary follows from Lemma 4. □

The following theorem is now immediate.

Theorem 1. *The primal-dual algorithm in Section 4 is a 5-approximation for the OWMR with independent retailer-warehouse holding costs.*

References

Bienkowski, M., Byrka, J., Chrobak, M., Dobbs, N., Nowicki, T., Sviridenko, M., Swirszcz, G., e Young, N. E. (2013). Approximation Algorithms for the Joint Replenishment Problem with Deadlines. In *Automata, Languages, and Programming*, p. 135–147.

Bienkowski, M., Byrka, J., Chrobak, M., Jeż, Ł., Nogneng, D., e Sgall, J. (2014). Better Approximation Bounds for the Joint Replenishment Problem. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, p. 42–54.

Buchbinder, N., Kimbrelt, T., Levi, R., Makarychev, K., e Sviridenko, M. (2008). Online make-to-order joint replenishment model: primal dual competitive algorithms. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, p. 952–961.

Levi, R., Lodi, A., e Sviridenko, M. (2007). Approximation Algorithms for the Multi-item Capacitated Lot-Sizing Problem Via Flow-Cover Inequalities. In *Integer Programming and Combinatorial Optimization*, p. 454–468.

Levi, R., Roundy, R. O., e Shmoys, D. B. (2006). Primal-Dual Algorithms for Deterministic Inventory Problems. *Mathematics of Operations Research*, 31(2):267–284.

Levi, R., Roundy, R. O., Shmoys, D. B., e Sviridenko, M. (2008). A Constant Approximation Algorithm for the One-Warehouse Multiretailer Problem. *Management Science*, 54(4):763–776.

Nonner, T. e Sviridenko, M. (2013). An Efficient Polynomial-Time Approximation Scheme for the Joint Replenishment Problem. In *Integer Programming and Combinatorial Optimization*, p. 314–323.

Roundy, R. (1985). 98%-effective integer-ratio lot-sizing for one-warehouse multi-retailer systems. *Management Science*, 31(11):1416–1430.

Stauffer, G., Massonnet, G., Rapine, C., e Gayon, J.-P. (2011). A simple and fast 2-approximation algorithm for the one-warehouse multi-retailers problem. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, p. 67–79.