Improved Algorithms for a Lot-Sizing Problem with Inventory Bounds and Backlogging

Hark-Chin Hwang

Department of Industrial Engineering, Chosun University
375 Seosuk-Dong, Dong-Gu, Gwangju 501-759, South Korea, hchwang@chosun.ac.kr

Wilco van den Heuvel
Econometric Institute and Erasmus Research Institute of Management, Erasmus University Rotterdam, P.O. Box 1738, 3000 DR Rotterdam, The Netherlands, wvandenheuvel@ese.eur.nl

March 9, 2010

Econometric Institute Report EI2010-17

Abstract

This paper considers a dynamic lot-sizing problem with storage capacity limitation in which backlogging is allowed. For general concave production and inventory costs, we present an $O(T^2)$ dynamic programming algorithm where T is the length of the planning horizon. Furthermore, for fixed-charge and nonspeculative costs, we provide $O(T \log T)$ and $O(T)$ algorithms, respectively. This paper therefore concludes that the time complexity to solve the bounded inventory lot-sizing problem with backlogging is the same as the complexity to solve the uncapacitated lot-sizing problem for the commonly used cost structures.

Keywords: Lot-sizing; Storage capacity; Inventory and Production; Algorithms

1. Introduction

In this paper we consider a dynamic lot-sizing problem with backlogging and a bound on the inventory level. In the problem, we are given the (deterministic) demands for a fixed number of periods. In each period, demand can be satisfied (i) by production in the period itself, (ii) from inventory built up in previous periods, or (iii) by backlogging in future periods. We assume that the warehouse has a finite capacity, which means that the on-hand inventory level is bounded from above in each period. The cost structures in our model all exhibit economies of scale as commonly used in the literature. They include a concave cost structure, a (general) fixed-charge cost structure and a fixed-charge cost structure without speculative motives.

Since the $O(T^2)$ algorithm for the uncapacitated lot-sizing problem of Wagner and Whitin [23], there have been great efforts to generalize the model and solve it efficiently. Zangwill [25] generalized the uncapacitated problem to consider backlogging and solved it in $O(T^3)$ time under concave costs. Utilizing the mo-

notonicity properties in the planning horizon theorem, Federgruen and Tzur [6] developed $O(T\log T)$ and $O(T)$ algorithms for fixed-charge and nonspeculative cost structures, respectively. The same results were obtained independently by Wagelmans et al. [22] and Aggarwal and Park [2]. The approaches used in Federgruen and Tzur [6] and Wagelmans et al. [22] are based on essentially the same geometric technique, which we call the *points-approach*, that maintains a lower convex hull (envelope) of given points. However, Aggarwal and Park [2] solve the lot-sizing problem by matrix searching algorithms utilizing the so-called Monge property. They presented an improved $O(T^2)$ backlogging algorithm over the algorithm of Zangwill [25] under concave costs. Van Hoesel et al. [21] extended the idea of the geometric technique in Wagelmans et al. [22] and presented a generalized approach for solving special dynamic programming problems. They also introduced a dual method of the points-approach, which we call the *lines-approach*, that constructs a lower convex envelope of given lines.

The lot-sizing problem with warehouse storage capacity (or bounded inventory) was first studied by Love [11] under general concave production and inventory holding/backlogging costs. Pochet and Wolsey [14], Atamtürk and Küçükyavuz [3] and Wolsey [24] performed a polyhedral study on the problem with fixed-charge costs. In case of nonspeculative costs, Pochet and Wolsey [14] provided linear programming extended formulations for the bounded inventory problem with and without production capacities. Atamtürk and Küçükyavuz [3] presented valid inequalities for the case that inventory holding cost has a fixed setup to deal with a situation of an outsourced warehouse. Wolsey [24] studied the bounded inventory problem in relation to the lot-sizing problem with production time windows. In this problem the production of demand k should happen during some time window $[a_k, b_k]$ and it must be satisfied in period b_k. Wolsey [24] showed that the bounded inventory problem is equivalent to the lot-sizing problem with non-inclusive production time windows. Similar to this result, Van den Heuvel and Wagelmans [19] proved the equivalence of the bounded inventory problem with the lot-sizing problem with a remanufacturing option (Richter and Sombrutzki [16]) and the lot-sizing problem with cumulative capacities (Sargut and Romeijn [17]). Consequently, the lot-sizing problem with inventory bounds is shown to be the same as those with non-inclusive time windows, a remanufacturing option or cumulative capacities.

We now review the solution algorithms developed for the bounded inventory problem. Love proposed an $O(T^3)$ algorithm for the problem with concave production and inventory holding/backlogging costs. Gutiérrez et al. [7, 8] designed another $O(T^3)$ algorithm, which is shown to be fast in practice. Under the fixed-charge cost structure, Toczylowski [18] developed an $O(T^2)$ algorithm by reducing multi-graph edges in a shortest path graph. For the same problem with a generalized fixed-charge holding cost structure, Atamtürk and Küçükyavuz [4] proposed an $O(T^2)$ algorithm.

The contribution of this paper is as follows. It has been an open question whether there exists a more efficient algorithm than the $O(T^3)$ algorithm of Love [11] for the bounded inventory lot-sizing problem with backlogging and a concave cost structure. We show that a slightly less general version of the problem can be solved in $O(T^2)$, by exploiting the Monge property inherent in the problem, as has been done in Aggarwal and Park [2] for the uncapacitated problem. Our model is slightly less general because only carried forward inventory is bounded, while Love [11] considers a bound on the amount of backlogged items as well. For the problem with fixed-charge and nonspeculative costs, we generalize the results of Gutiérrez et al. [9] and Liu [10] to the backlogging case and present $O(T\log T)$ and $O(T)$ algorithms based on the geometric technique of Van Hoesel et al. [21]. Part of this geometric technique consists of maintaining a lower convex hull of points and updating it with new points. Because in our problem we also need to delete points from the lower convex hull, we cannot apply this approach directly. To overcome this, especially for the problem with fixed-charge costs, we combine it with the results of Brodal and Jacob [5] who proposed an algorithm to (dynamically) maintain a convex hull of points. For the problem with nonspeculative costs, we apply the lines-approach and generalize it to a *line-segments-approach* to handle the deletion problem in maintaining lower envelopes of lines.

In conclusion, we show that the time complexity to solve the bounded inventory lot-sizing problem with backlogging is the same as to solve the uncapacitated lot-sizing problem. Together with the result of Van den Heuvel and Wagelmans [19], this means that the lot-sizing problems with bounded inventory, non-inclusive time windows, a remanufacturing option and cumulative capacities can all be solved in the same complexity as the uncapacitated lot-sizing problem. We finally note that Gutiérrez et al. [9] and Liu [10] recently developed an $O(T\log T)$ algorithm for the fixed-charge cost structure and an $O(T)$ algorithm for the nonspeculative cost structure, by using the results of Wagelmans et al. [22]. Although the recursive equations in Gutiérrez et al. [9] are correct, their proposed $O(T\log T)$ time implementation does not provide an optimal solution in general (as shown by the counterexample in Van den Heuvel et al. [20]). Furthermore, it also turns out that the algorithm of Liu [10] does not obtain an optimal solution in general within their claimed running time (Onal [12]).

The remainder of the paper is organized as follows. In Section 2 we formulate the problem and provide basic optimality properties. Section 3 deals with the concave cost case, while in Section 4 and Section 5 we develop algorithms to solve the fixed-charge and nonspeculative cost cases, respectively. The paper is concluded in Section 6.

2. Problem Formulation and Optimality Properties

Let T denote the length of planning horizon. For each period $t \in \{1, 2, \ldots, T\}$ we define the following notation.

- d_t: demand in t.
- u_t: *effective* storage capacity in t; without loss of generality, we assume that each storage capacity u_t satisfies $u_t \le d_t + u_{t+1}$. If not, we obtain this property by setting $u_t = \min\{u_{t'} + d_t + d_{t+1} + \cdots + d_{t'-1}: \ t < t' \le T\}$, *as this does not change the set of feasible solutions* (Toczylowski [18], Atamtürk and Küçükyavuz [3]). For feasibility, we also assume that $u_t \ge d_t$.
- x_t: production level in t. We assume that x_t units are all produced and available at the beginning of period t.
- I_t^+: on-hand inventory level at the end of t.
- I_t^-: backlog inventory level at the end of t.
- I_t: inventory level in t, defined as $I_t = I_t^+ - I_t^-$.
- $p_t(x_t)$: concave production cost function in t for the amount $x_t \ge 0$.
- $h_t(I_t^+)$: concave inventory cost for keeping $I_t^+ \ge 0$ units at the end of period t.
- $b_t(I_t^-)$: concave inventory cost for backlogging $I_t^- \ge 0$ units at the end of period t.

The cost functions $p_t(x_t)$, $h_t(x_t)$ and $b_t(x_t)$ are assumed to be concave with $p_t(0) = h_t(0) = b_t(0) = 0$. The model is said to have a fixed-charge cost structure if

$$p_t(x_t) = K_t + p_t \cdot x_t \text{ if } x_t > 0,\ h_t(x_t) = h_t \cdot x_t \text{ and } b_t(x_t) = b_t \cdot x_t,$$

where K_t is the setup cost, and p_t, h_t and b_t are the unit production, holding and backlogging costs in period t, respectively. Moreover, if it holds that $p_t - b_t \le p_{t+1} \le p_t + h_t$ for $t = 1, 2, \ldots, T-1$, the cost structure is called nonspeculative. For notational convenience, we let $v_{s,t} = v_s + v_{s+1} + \ldots + v_t$ if $s \le t$ and $v_{s,t} = 0$ if $s > t$, for any sequence of values $v_s, v_{s+1}, \ldots, v_t$. So $d_{s,t}$ represents the cumulative sum of demands from s through t.

The lot-sizing problem with storage capacity is modeled as follows:

$$\text{Min} \sum_{t=1}^{T} \left(p_t(x_t) + h_t(I_t^+) + b_t(I_t^-) \right)$$

Subject to

$$(I_{t-1}^+ - I_{t-1}^-) + x_t = d_t + (I_t^+ - I_t^-), \quad t = 1, \ldots, T, \tag{1}$$

$$I_{t-1} + x_t \le u_t, \quad t = 1, \ldots, T, \tag{2}$$

$$I_0^+ = I_0^- = I_T^+ = I_T^- = 0,$$

$$x_t \ge 0,\ I_t^+ \ge 0, I_t^- \ge 0, \quad t = 1, \ldots, T.$$

Constraint (1) represents the inventory balance equation and constraint (2) the storage capacity limits. Most research on the lot-sizing problem with storage limitation puts the storage constraint on the final inventory levels, i.e., $I_t \le u_t$ (Love [11]; Atamtürk and Küçükyavuz [3, 4]; Wolsey [24]), which means produced units are immediately shipped to customers with demand in period t without storage. In our model, we assume that each produced (or purchased) unit is always stored first before being shipped to a customer having demand in the current or a future period, resulting in the constraint on the initial inventory levels, i.e., $I_{t-1} + x_t \le u_t$. Note that if $I_{t-1}^+ > 0$, then $I_{t-1}^- = 0$, so that the constraint becomes $I_{t-1}^+ + x_t \le u_t$. For backlogged demands, however, we assume that production (or purchasing) fulfills them instantaneously without any storage, so that we do not have an upper bound on the level of backlogged inventory. In this case $I_{t-1}^+ = 0$ and $I_{t-1}^- > 0$, so that the constraint becomes $x_t - I_{t-1}^- \le u_t$ and hence there is no upper bound on the backlogging level. We note that the two problems with a bound on either the initial or the final inventory level are equivalent because of the balance equation (1) (Gutiérrez et al. [7]).

We now provide an explicit feasibility test scheme in regard to the storage limitation in each period s. Since the inventory and production in period s cannot supply more than u_s units, we need to identify the period's maximum coverage by the u_s units. We define $n(s)$ to be the latest period by which every demand $d_s, d_{s+1}, \ldots, d_{n(s)}$ can be fulfilled:

$$n(s) = \max\{t: d_{s,t} \le u_s,\ t = s, s+1, \ldots, T\},$$

where $n(0) = 0$. Now we consider the opposite case of $n(s)$. Given a period t, we are interested in the earliest period that can cover up the demand d_t, which is denoted as $m(t)$:

$$m(t) = \min\{s: d_{s,t} \le u_s,\ s = 1, 2, \ldots, t\},$$

where $m(0) = 0$.

A period s is called a *regeneration period* if $I_s = 0$, which has been used for the ZIO (zero-inventory) policy in uncapacitated problems. This policy states that if periods $s-1$ and t are consecutive regeneration periods, then a single production during $\{s, s+1, ..., t\}$ covers all the demands $d_s, d_{s+1}, \ldots, d_t$. If u_s units are available at the beginning of period s, i.e., $I_{s-1} + x_s = u_s$, we say that period s is a *warehouse period*. A period is called an *inventory period*, if it is either a regeneration or a warehouse period. Furthermore, a period s is called a *production period* if $x_s > 0$. Finally, production period s is called a *warehouse production period* if s is a warehouse period. We further introduce two important types of production periods:

- Production period s is called a *complete* (*production*) *period* if a regeneration period $t \geq s$ precedes any production periods after s; that is, no production period exists between production periods s and t. In this case, the initial inventory level plus the production quantity in period s exactly covers the demands during periods $\{s, s+1, \ldots, t\}$, that is $I_{s-1} + x_s = d_{s,t}$.
- Production period s is called a *successive* (*production*) *period* if a production period $t > s$ precedes any regeneration period; no regeneration period exists between the two periods s and t. As we shall see later, any successive period will be a warehouse production period.

Note that the ZIO policy applies to complete productions but does not for successive productions. We also note that any complete period is not a successive period, and vice versa. In developing the dynamic programming procedures, we will mostly use complete and successive periods. The term warehouse period will be used for explicitly designating the quantity up to the maximum level at the start of a period.

Most lot-sizing problems can be represented as a concave cost network flow problem from which optimality properties of the extreme point solution can be characterized. It is well known that the unsaturated flow corresponding to an extreme point solution contains no cycle (Zangwill [26]). Applying the no-cycle property to the bounded inventory problem, we have the following relationships between inventory and production periods (Love [11]):

(i) between any two consecutive inventory periods there is at most one production period,
(ii) between any two consecutive production periods there is at least one inventory period.

Considering the effective storage capacity assumption, these properties also read as follows.

Property 1 (Love [11], Gutiérrez et al. [8])**.** There exists an optimal schedule such that each period s satisfies $x_s(I_{s-1} + x_s - u_s)(I_{s-1} + x_s - d_{s,t}) = 0$ for some t, $s \le t \le T$.

Based on this property, Love [11] and Gutiérrez et al. [7, 8] were able to develop optimal solution procedures with complexity $O(T^3)$. Note that if $I_{s-1} + x_s = d_{s,t}$ for some t, $s \le t \le T$, then period s is a complete production period. Love's second property shows the existence of an inventory period between production periods but does not point out the inventory period explicitly. However, the effective storage capacity assumption with Property 1 makes it possible to specify the inventory (warehouse) period between production periods when the inventory period is not a regeneration period. In the following, we show that any successive production period has items up to its storage capacity. In other words, any successive production period is by definition an inventory (warehouse) period. This property can also be derived by applying the no-cycle property.

Property 2. There exists an optimal schedule such that $I_{s-1} + x_s = u_s$ for each successive period s.

Suppose that s is a production period and t is the first regeneration period after period s. If period s is successive (implying that s is a warehouse period), then a production period $i < t$, exists with $I_{i-1} \neq 0$. If this period i is complete, then there is no more productions during $\{i+1, i+2, \ldots, t\}$. Otherwise, there is another production period between i and t. Consequently, between consecutive regeneration periods, there are two production patterns: a single complete production occurs or a series of successive productions take place followed by a complete production. This observation together with Love's properties shows that inventory periods (regeneration periods) and production periods (successive production periods) play a structural role in extreme point solutions. We will solve the bounded inventory problem by decomposing a solution based on regeneration and successive production periods. The decomposition by regeneration periods will generate a new self-contained subproblem while the decomposition by successive production periods will allow us to determine production quantities. Since each successive production period is a warehouse period, our decomposition is essentially based on inventory periods as was done by Love [11]. In the following sections we will derive a dynamic programming algorithm that is based on the decomposition into regeneration periods and successive production periods. To that end, we define the following variables:

- $F(s)$ is the minimum cost of producing demands $d_s, d_{s+1}, \ldots, d_T$ under the situation that period $s-1$ is a regeneration period. We let $F(T+1) = 0$. Note that the optimal solution is given by $F(1)$.

- $G(s)$ is the minimum cost of producing demands $d_s, d_{s+1},\ldots, d_T$ under the condition that
  - (i) period s is a successive production period, and
  - (ii) production cost in period s is not included.

  We let $G(T) = \infty$.

Before proceeding to the next section, we present notations associated with holding and backlogging costs:

- $h(s, t)$: the holding cost for fulfilling demands $d_{s+1}, \ldots, d_t$ with $d_{s+1,t}$ units at the end of period s, i.e., $h(s, t) = \sum_{i=s}^{t-1} h_i(d_{i+1,t})$.
- $b(s, t)$: the backlogging cost for fulfilling demands $d_s, d_{s+1}, \ldots, d_{t-1}$ with $d_{s,t-1}$ units at the beginning of period t, i.e., $b(s, t) = \sum_{i=s}^{t-1} b_i(d_{s,i})$.
- $h'(s, t)$: the holding cost for supplying demands $d_{s+1}, \ldots, d_t$ when having u_s units in period s (so having $u_s - d_{s,t} > 0$ units at the end of period t), i.e., $h'(s, t) = \sum_{i=s}^{t} h_i(u_s - d_{s,i})$.

The term $h(s, t)$ describes the inventory carrying cost when period s is a complete period with its regeneration in period t. Similarly, $b(s, t)$ is the inventory backlogging cost when period $s-1$ is a regeneration period with period t the first production period after period s. The cost $h'(s, t-1)$ denotes the inventory carrying cost when period s is successive with its next production in period t. Note that in the definition of $h'(s, t-1)$, the demands $d_s, d_{s+1}, \ldots, d_{t-1}$ are fulfilled by the initial inventory and production in period s without supply by backlogging from period t, so $t \le n(s) + 1$. Now consider the case $t > n(s)+1$, which means that demands $d_s, \ldots, d_{n(s)}$ are all supplied from period s and demands $d_{n(s)+2}, \ldots, d_{t-1}$ are supplied from period t. The demand $d_{n(s)+1}$ may be supplied by both periods s and t. Since we know that u_s units are available at the beginning of period s, the total amount supplied from period t should be $d_{s,t-1} - u_s$. These $d_{s,t-1} - u_s$ units are supplied for demands $d_{n(s)+1}, d_{n(s)+2}, \ldots, d_{t-1}$ with backlogging cost $b'(s, t)$ defined as:

- $b'(s, t)$: the backlogging cost for supplying demands $d_{n(s)+1}, d_{n(s)+2}, \ldots, d_{t-1}$ using $d_{s,t-1} - u_s$ units at the beginning of period t, i.e., $b'(s, t) = \sum_{i=n(s)+1}^{t-1} b_i(d_{s,i} - u_s)$.

Hence, if $t > n(s)+1$, the total holding and backlogging cost during $\{s, s+1, \ldots, t-1\}$ equals

$$h'(s, n(s)) + b'(s, t). \tag{3}$$

Finally, note that using the recursion $h(s-1, t) = h(s, t) + h_{s-1}(d_{s,t})$, we can compute all values $h(s, t)$ in $O(T^2)$ time. A similar recursion holds for the computation of $h'(s, t)$, $b(s, t)$ and $b'(s, t)$. This means that all inventory and backlogging related costs can be computed in $O(T^2)$ time by preprocessing.

3. Concave Costs

In this section, we first describe an ordinary $O(T^3)$ dynamic programming algorithm for concave costs, and then show how it can be implemented in $O(T^2)$ time using the matrix-searching algorithm as in Aggarwal and Park [2].

3.1 An $O(T^3)$ Dynamic Programming Algorithm

To obtain the optimal solution $F(1)$, we will determine $F(s)$ and $G(s)$ by a backward dynamic programming algorithm iterating from $s = T$ to 1. In the remainder of this section we will further decompose the cost terms $F(s)$ and $G(s)$. The primary purpose of this is to make possible the efficient computation of $F(s)$ and $G(s)$, by exploiting the Monge property in the next subsection. First, let $F(s, i)$ denote the cost $F(s)$ under the restriction that the first production after regeneration period $s-1$ occurs at period i which is complete, $1 \le s \le i \le T$. Similarly, $G(s, i)$ denotes the cost under the constraint that the successive production period s has its next production in period i which is complete, $1 \le s < i \le T$. To determine the production quantity of a complete period, we need to further specify the regeneration period after the complete period. To this end, we introduce the cost terms $f_i(s, t)$, and $g_i(s, t)$:

- $f_i(s, t)$ is the minimum cost of producing demands $d_s, d_{s+1}, \ldots, d_T$, $1 \le s \le i \le t \le n(i)$, under the constraints that
  - (i) $s-1$ and t are consecutive regeneration periods,
  - (ii) period i is the only production period during $\{s, ..., t\}$, which means that period i is complete.
- $g_i(s, t)$ is the minimum cost of producing demands $d_s, d_{s+1}, \ldots, d_T$, $1 \le s < i \le t \le n(i)$, under the constraints that

(i) s is a successive period with its next production period i, which is complete,

(ii) the production cost in period s is not included,

(iii) period t is the first regeneration period after period i.

Similar to $G(s)$, the cost $g_i(s, t)$ also does not include the production cost of period s.

In the following we will show how the cost terms $F(s, i)$, $f_i(s, t)$, $G(s, i)$ and $g_i(s, t)$ are used for determining $F(s)$ and $G(s)$.

Computing $F(s)$. Since $I_{s-1} = 0$, there must be at least one production during $\{s, s+1, ..., T\}$. Let i be the first production period at or after period s. We further suppose that period i is complete. Then, in this case we have, by definition,

$$F(s) = F(s, i).$$

Next, suppose that period i is a successive production period. With the information that i is successive, the cost for satisfying demands d_i, d_{i+1}, …, d_T is $G(i)$. Then consider the cost during $\{s, s+1, ..., i-1\}$. Since we have no production during $\{s, s+1, ..., i-1\}$, the backlogging level at the end of period $i-1$ goes up to $d_{s,i-1}$ (i.e., $I_{i-1} = -d_{s,i-1}$), for which backlogging cost $b(s, i)$ occurs. Note that $I_{i-1} + x_i = u_i$ by Property 2, which means $x_i = u_i + d_{s,i-1}$. Since the cost $G(i)$ does not include the production cost $p_i(u_i + d_{s,i-1})$ of period i, we need to count it here. Thus, the cost $F(s)$ in this case is obtained as

$$F(s) = b(s, i) + p_i(u_i + d_{s,i-1}) + G(i). \tag{4}$$

Combining the two formulas for $F(s)$ with the initial condition of $F(T+1) = 0$, we have:

$$\begin{aligned} &F(T+1) = 0, \\ &F(s) = \min\begin{cases} F(s,i) : s \le i \le T, \\ b(s,i) + p_i(u_i + d_{s,i-1}) + G(i) : s \le i \le T. \end{cases} \end{aligned} \tag{5}$$

Computing $F(s, i)$ and $f_i(s, t)$. Since period i is complete in the computation of $F(s, i)$, there exists a regeneration period t before any other subsequent production period. Note that demands d_i, d_{i+1}, ..., d_t should be fulfilled by the inventory and production in period i, which is no greater than u_i. That is, it should hold that $d_{i,t} \le u_i$ or $t \le n(i)$. Then, for $1 \le s \le i \le T$, the cost $F(s, i)$ can be simply derived from $f_i(s,$

t) by the formula:

$$F(s, i) = \min\{f_i(s, t): i \le t \le n(i)\}. \tag{6}$$

We now explain how to compute $f_i(s, t)$. Since period t is a regeneration period, the cost during $\{t+1, t+2, ..., T\}$ is $F(t+1)$. The demands during $\{i+1, ..., t\}$ are supplied by carrying inventory in period i with cost $h(i, t)$. We consider the production quantity in period i. Since period $s-1$ is a regeneration period, the production in period i should supply demands $d_s, d_{s+1}, ..., d_t$, that is, $x_i = d_{s,t}$ with cost $p_i(d_{s,t})$. Note that the backlogging cost during $\{s, ..., i-1\}$ is $b(s, i)$. Hence, we have, for $1 \le s \le i \le t \le n(i) \le T$,

$$f_i(s, t) = b(s, i) + p_i(d_{s,t}) + h(i, t) + F(t+1). \tag{7}$$

Computing $G(s)$. Let i be the next production period of s. Suppose that period i is complete. Then, in this case we have, by definition,

$$G(s) = G(s, i).$$

We next consider the case that period i is successive. Note that $I_{i-1} = (I_{s-1} + x_s) - d_{s,i-1}$. Since period s is successive (and so a warehouse period), it supplies u_s units ($I_{s-1} + x_s = u_s$), which means that $I_{i-1} = u_s - d_{s,i-1}$. This with the fact that period i is also successive implies that $x_i = u_i - u_s + d_{s,i-1}$, incurring production cost of $p_i(u_i - u_s + d_{s,i-1})$. The cost during $\{s+1, ..., i-1\}$ depends on whether the available amount in period s can fully supply demands $d_{s+1}, ..., d_{i-1}$, which is the case where $i \le n(s) + 1$. If $i \le n(s) + 1$, then the cost is given by $h'(s, i-1)$. Otherwise if $i > n(s) + 1$, the cost is $h'(s, n(s)) + b'(s, t)$ as shown in (3). Thus, in this case, we obtain

$$G(s) = \min \begin{cases} h'(s, i-1) + p_i(u_i - u_s + d_{s,i-1}) + G(i), & \text{if } s < i \le n(s) + 1, \\ h'(s, n(s)) + b'(s, i) + p_i(u_i - u_s + d_{s,i-1}) + G(i), & \text{otherwise.} \end{cases} \tag{8}$$

With the initial condition of $G(T) = \infty$, we have a complete formula for $G(s)$ as follows:

$$G(T)=\infty,$$
$$G(s)=\min\begin{cases} G(s,i): s<i\le T, \\ h'(s,i-1)+p_i(u_i-u_s+d_{s,i-1})+G(i): s<i\le n(s)+1. \\ h'(s,n(s))+b'(s,i)+p_i(u_i-u_s+d_{s,i-1})+G(i): n(s)+1<i\le T. \end{cases} \tag{9}$$

Computing $G(s, i)$ and $g_i(s, t)$. We first take the computation of $G(s, i)$ into account where period i is complete. Let t be the first regeneration period at or after period i. Note that demands d_i, d_{i+1}, ..., d_t should be fulfilled by the inventory and production in period i, which is no greater than u_i. That is, it should hold that $d_{i,t} \le u_i$ or $t \le n(i)$. Thus, for $1 \le s < i \le T$, we have

$$G(s, i) = \min\{g_i(s, t): i \le t \le n(i)\}.$$

We finally explain the formula for $g_i(s, t)$. As in (7), we can see that the cost during $\{i+1, ..., T\}$ is $h(i, t) + F(t+1)$. Since $(I_{s-1}+x_s) + x_i = d_{s,t}$ and $I_{s-1}+x_s = u_s$, we have $x_i = d_{s,t} - u_s$ incurring production cost of $p_i(d_{s,t} - u_s)$. The holding and backlogging cost during $\{s+1, ..., i-1\}$ depends on $i \le n(s) +1$ or $i > n(s) +1$ as in (9). Hence, for $1 \le s < i \le t \le n(i) \le T$, $g_i(s, t)$ can be obtained by

$$g_i(s,t)=\min\begin{cases} h'(s,i-1)+p_i(d_{s,t}-u_s)+h(i,t)+F(t+1): s<i\le n(s)+1. \\ h'(s,n(s))+b'(s,i)+p_i(d_{s,t}-u_s)+h(i,t)+F(t+1): n(s)+1<i\le T. \end{cases} \tag{10}$$

Note that all values $f_i(s, t)$, $1 \le s \le i \le t \le n(i)$, and $g_i(s, t)$, $1 \le s < i \le t \le n(i)$, are computed in $O(T^3)$ time for all $i = 1, 2, \ldots, T$. Since the computations of $F(s)$ by (5) and $G(s)$ by (9) take $O(T^2)$, the bottleneck is the computation of $f_i(s, t)$ and $g_i(s, t)$.

3.2 An Improved $O(T^2)$ Algorithm for Concave Costs

In this section, we present efficient implementations of $f_i(s, t)$ and $g_i(s, t)$ based on the Monge property. Aggarwal and Park [2] show that an $n \times n$ matrix $e = \{e(s, t)\}$ is inverse-Monge if each value $e(s, t)$ can be represented as

$$e(s, t) = a_s + a_t' + p(y_t' - y_s),$$

where a_s, a_t', y_s and y_t' are constants with $y_1 \le y_2 \le \cdots \le y_n$ and $y_1' \le y_2' \le \cdots \le y_n'$, and p is a concave function. It is well-known that the matrix-searching algorithm of Aggarwal et al. [1] can obtain the column or row minima of the inverse-Monge matrix e in time $O(n)$; that is, all the values $\min\{e(s, t): t = 1, 2, \ldots, n\}$ for $s = 1, 2, \ldots, n$ can be computed in $O(n)$ time. Using this algorithm Aggarwal and Park [2] were able to solve the uncapacitated lot-sizing problem with backlogging in $O(T^2)$ time, improving on the $O(T^3)$ algorithm of Zangwill [25]. From now on we fix i and view f_i as a $T \times T$ matrix with its elements $f_i(s, t)$. In particular, we are interested in the submatrix $f_i(s, t)$ with indices (s, t) satisfying $1 \le s \le i$ and $i \le t \le n(i)$.

Lemma 1. The submatrix $\{f_i(s, t):1 \le s \le i, i \le t \le n(i)\}$ is inverse-Monge for $i = 1, 2, \ldots, T$.

Proof. For a given period i, consider the element $f_i(s, t) = b(s, i) + p_i(d_{s,t}) + h(i, t) + F(t+1)$. The term $b(s, i)$ only depends on period s, while the term $h(i, t) + F(t+1)$ only depends on period t. However, the production cost term $p_i(d_{s,t})$ relies on both periods s and t. By rewriting $d_{s,t} = d_{1,t} - d_{1,s-1}$ and by noting that $d_{1,1} \le d_{1,2} \le \cdots \le d_{1,t}$, we conclude that the submatrix $\{f_i(s, t): 1 \le s \le i, i \le t \le n(i)\}$ is inverse-Monge. □

Suppose that each value $F(t+1)$ is preprocessed for $i \le t \le n(i)$. Furthermore, suppose that the terms $h(s, t)$, $b(s, t)$, $b'(s, t)$, $h'(s, t)$ and $d_{s,t}$ are all preprocessed in $O(T^2)$ time for $1 \le s \le t \le T$. Then, each cost term $f_i(s, t)$ is evaluated in constant time by (7). By the fact that the submatrix of $\{f_i(s, t)\}$ is inverse-Monge, we can obtain its corresponding row (or column) minima in $O(T)$ time. That is, every value of $\min\{f_i(s, t): i \le t \le n(i)\}$ for $s = 1, 2, \ldots, i$ is computed in $O(T)$. Note that these minima are exactly the values $F(s, i)$ in (6) for $s = 1, 2, \ldots, i$ and they are used to calculate (5).

Similar to $f_i(s, t)$, we will also determine $g_i(s, t)$ by using the matrix-searching algorithm. Again, we fix the period i. Then, $g_i(s, t)$ is only defined for $m(i-1) \le s < i$ and $i \le t \le n(i)$. So, for fixed i, formula (10) can be rewritten as

$$g_i(s,t) = \min \begin{cases} h'(s,i-1) + p_i(d_{s,t} - u_s) + h(i,t) + F(t+1) : m(i-1) \le s < i,\ i \le t \le n(i), \\ h'(s,n(s)) + b'(s,i) + p_i(d_{s,t} - u_s) + h(i,t) + F(t+1) : 1 \le s < m(i-1),\ i \le t \le n(i). \end{cases}$$

Let $g_i'(s, t) = h'(s, i-1) + p_i(d_{s,t} - u_s) + h(i, t) + F(t+1)$ and $g_i''(s, t) = h'(s, n(s)) + b'(s, i) + p_i(d_{s,t} - u_s) + h(i, t) + F(t+1)$, so that $g_i(s, t) = \min\{g_i'(s, t), g_i''(s, t)\}$. Then, using analogous arguments as in Lemma 1, we can prove the following.

Lemma 2. Both the submatrices $\{g_i'(s, t): m(i-1) \le s < i,\ i \le t \le n(i)\}$ and $\{g_i''(s, t): 1 \le s < m(i-1),\ i \le t \le n(i)\}$ are inverse-Monge for $i = 1, 2, \ldots, T$.

Let $G'(s, i) = \min\{g_i'(s, t): i \le t \le n(i)\}$ for each s with $m(i-1) \le s < i$, and $G''(s, i) = \min\{g_i''(s, t): i \le t \le n(i)\}$ for each s with $1 \le s < m(i-1)$. Then all the values $G'(s, i)$ for $m(i-1) \le s < i$, and $G''(s, i)$ for $1 \le s < m(i-1)$ are obtained in $O(T)$ by applying the matrix-searching algorithm. Now we present the whole procedure for computing an optimal solution. The algorithm proceeds by complete periods $i = T, T-1, \ldots, 1$. At stage i, suppose that we are given the following values:

- $F(t)$ for $i+1 \le t \le T+1$ and $G(t)$ for $i+1 \le t \le T$,
- $F(s, t)$ for $1 \le s \le t$, $i+1 \le t \le T$, and $G(s, t)$ for $1 \le s \le t-1$, $i+1 \le t \le T$.

Then the following steps are performed at stage i:

Step 1. Compute $G(i)$ by (9).

Step 2. Compute the row minima of the submatrix $\{f_i(s, t): 1 \le s \le i,\ i \le t \le n(i)\}$ by the matrix-searching algorithm, obtaining the values $F(1, i), F(2, i), \ldots, F(i, i)$ in (6).

Step 3. Compute $F(i)$ by (5).

Step 4. Compute the row minima of the submatrix $\{g_i'(s, t): m(i-1) \le s < i,\ i \le t \le n(i)\}$ and $\{g_i''(s, t): 1 \le s < m(i-1),\ i \le t \le n(i)\}$ by the matrix-searching algorithm, obtaining the values $G''(s, i)$ for $1 \le s < m(i-1)$ and $G'(s, i)$ for $m(i-1) \le s < i$. Set $G(s, i) = G''(s, i)$ for $1 \le s < m(i-1)$ and $G(s, i) = G'(s, i)$ for $m(i-1) \le s < i$.

Note that the Steps 1–4 in each stage are executed in $O(T)$ time and hence the overall complexity of the algorithm is $O(T^2)$.

Theorem 1: The lot-sizing problem with inventory bounds, backlogging and concave costs can be solved in $O(T^2)$ time.

4. Fixed-charge Costs

In this section we consider the problem under a fixed-charge cost structure, which means that

$$p_t(x_t) = K_t + p_t \cdot x_t \text{ if } x_t > 0,\ h_t(x_t) = h_t \cdot x_t \text{ and } b_t(x_t) = b_t \cdot x_t,$$

where K_t is the setup cost, and p_t, h_t and b_t are unit production, holding and backlogging costs in period t, respectively. For the nonspeculative cost structure it is further assumed that $p_t - b_t \le p_{t+1} \le p_t + h_t$ for $t = 1, 2, \ldots, T-1$. We first describe an $O(T^2)$ recursive procedure, and then show how it can be implemented in $O(T \log T)$ and $O(T)$ time for fixed-charge and nonspeculative costs, respectively.

4.1 An $O(T^2)$ Dynamic Programming Algorithm

We will also use the cost terms $F(s)$ and $G(s)$ as in the concave cost case to solve the fixed-charge and nonspeculative cost problems. However, for efficient implementations under these specific cost structures, we will utilize a new cost term $f(i)$ for each complete production period i instead of $f_i(s, t)$ and $g_i(s, t)$ in the previous section:

- $f(i)$ denotes the minimum cost of supplying demands $d_i, d_{i+1}, \ldots, d_T$ under the constraints that
  - (i) i is a complete production period,
  - (ii) the setup cost K_i in period i is not included.

Since $f(i)$ satisfies the demands $d_i, d_{i+1}, \ldots, d_T$ by only productions during $\{i, i+1, ..., T\}$, we can solve the problem with the assumption that $I_{i-1} = 0$. We also note that $f(i)$ describes the cost from period i through the final period T while $f_i(s, t)$ and $g_i(s, t)$ consider the costs taking into account a regeneration period t. We first develop the formula for $f(i)$ followed by the ones for $F(s)$ and $G(s)$.

Computing $f(i)$. Suppose that period t is the first regeneration period at or after period i. Then the cost during $\{i, i+1, \ldots, t\}$ is $p_i d_{i,t} + h(i, t)$ and that during $\{t+1, t+2, \ldots, T\}$ is $F(t+1)$. Thus, the formula for $f(i)$ is given as

$$f(i) = \min\{p_i d_{i,t} + h(i, t) + F(t+1): i \le t \le n(i)\}. \qquad (11)$$

Computing $F(s)$. Let i be the first production period at or after period s. We further suppose that period i is complete. Then, demands $d_s, d_{s+1}, \ldots, d_{i-1}$ are satisfied by backlogging from production in period i. Since the unit production cost in period i is p_i, it takes $b(s, i) + p_i d_{s,i-1}$ to fulfill the demands $d_s, d_{s+1}, \ldots,$

d_{i-1}. Note that the cost during $\{i, i+1, \ldots, T\}$ is $f(i)$ since period i is complete. Because $f(i)$ does not include the setup cost K_i, the formula for $F(s)$ in this case, denoted by $F_1(s)$, is given as

$$F_1(s) = \min\{b(s, i) + K_i + p_i d_{s,i-1} + f(i): s \le i \le T\}. \tag{12}$$

On the other hand, if period i is successive, then the cost is the same as in (4), which we denote by $F_2(s)$,

$$F_2(s) = \min\{b(s, i) + K_i + p_i(u_i + d_{s,i-1}) + G(i): s \le i \le T\}. \tag{13}$$

So the complete formula for $F(s)$ is given by:

$$\begin{aligned} &F(T+1) = 0, \\ &F(s) = \min\{F_1(s), F_2(s)\}. \end{aligned} \tag{14}$$

Computing $G(s)$. Let i be the next production period after period $s \le T$. We first consider the case that period i is complete. We further suppose that $i \le n(s)+1$. Then demands $d_s, d_{s+1}, \ldots, d_{i-1}$ are all supplied by the u_s units available in period s, with holding cost $h'(s, i-1)$, leaving $I_{i-1} = u_s - d_{s,i-1}$ units at the end of period $i-1$. If we know the regeneration period $t \ge i$, then the production quantity in period i equals $x_i = d_{s,t} - u_s$, so that

$$G(s) = h'(s, i-1) + K_i + p_i(d_{s,t} - u_s) + h(i, t) + F(t+1).$$

Because this formula requires additional information on regeneration period t, we do not achieve the aim of efficient computation immediately. Observe now that the formula for $G(s)$ can be rewritten as:

$$G(s) = h'(s, i-1) + K_i + p_i(d_{s,i-1} - u_s) + p_i(d_{i,t}) + h(i, t) + F(t+1).$$

In this revised formula, we see that the last term $p_i(d_{i,t}) + h(i, t) + F(t+1)$ coincides with the cost $f(i)$ having a regeneration period in t. Thus, we are allowed to let

$$G(s) = h'(s, i-1) + K_i + p_i(d_{s,i-1} - u_s) + f(i).$$

Note that the quantity $d_{s,i-1} - u_s$ can be negative since we assumed that $i \le n(s) + 1$. So, when $d_{s,i-1} - u_s$ is negative, subtracting the cost for the I_{i-1} ($= u_s - d_{s,i-1}$) units in advance enables us to use $f(i)$, which has no inventory at the beginning of period i. Using this formula, however, may not result in an optimal solution, since it is valid only when the production quantity in period i, in the optimal schedule corresponding to $f(i)$, is no less than $u_s - d_{s,i-1}$. Therefore, we define $\phi_{f(i)}$ as the production quantity in period i in the schedule corresponding to $f(i)$ (i.e., $\phi_{f(i)} = d_{i,t}$ for some t). So, we need to ensure $\phi_{f(i)} \ge u_s - d_{s,i-1}$. In conclusion, the appropriate formula in this case, denoted by $G_1(s)$, is given as

$$G_1(s) = \min\{h'(s, i-1) + K_i + p_i(d_{s,i-1} - u_s) + f(i): \phi_{f(i)} \ge u_s - d_{s,i-1},\ s < i \le n(s)+1\}. \tag{15}$$

We can easily see that the case $\phi_{f(i)} < u_s - d_{s,i-1}$ is not feasible. Namely, the production quantity in period i equals $\phi_{f(i)} - (u_s - d_{s,i-1})$, which is negative in this case. Gutiérrez et al. [9] prove the existence of such period i with $\phi_{f(i)} \ge u_s - d_{s,i-1}$. In fact, period $i = n(s) + 1$ satisfies this condition.

We next consider the case that period i is complete with $i > n(s)+1$. In this case, $d_{s,i-1} - u_s$ is nonnegative. So, we can apply the following formula safely.

$$G_2(s) = \min\{h'(s, n(s)) + b'(s, i) + K_i + p_i(d_{s,i-1} - u_s) + f(i): n(s)+1 < i \le T\}. \tag{16}$$

When period i is successive, we can apply formula (8), which is denoted separately as follows.

$$G_3(s) = \min\{h'(s, i-1) + K_i + p_i(u_i - u_s + d_{s,i-1}) + G(i): s < i \le n(s) + 1\}, \text{ and} \tag{17}$$

$$G_4(s) = \min\{h'(s, n(s)) + b'(s, i) + K_i + p_i(u_i - u_s + d_{s,i-1}) + G(i): n(s)+1 < i \le T\}. \tag{18}$$

In summary, $G(s)$ is computed as follows:

$$\begin{aligned} &G(T) = \infty, \\ &G(s) = \min\{G_1(s), G_2(s), G_3(s), G_4(s)\}. \end{aligned} \tag{19}$$

Note that the costs $f(s)$, $F_1(s)$, $F_2(s)$, and $G_j(s)$, $j = 1, \ldots, 4$, for all $s = T, T-1, \ldots, 1$, are obtained in $O(T^2)$ time. Thus we also obtain $F(s)$ by (14) and $G(s)$ by (19) for all $s = T, T-1, \ldots, 1$ in $O(T^2)$ time.

We will now show how to improve the running time of the algorithm by using known results from the literature. As was done in Section 3.2, we might improve the computational complexity by applying the algorithms of Aggarwal and Park [2] for fixed-charge and nonspeculative costs. However, their algorithms are complex and it seems not easy to adapt them to the bounded inventory problem with fixed-charge and nonspeculative costs. Instead, we will utilize the geometric technique of Wagelmans et al. [22], a points-approach which is formalized in Van Hoesel et al. [21]. The geometric technique is used to solve dynamic programming problems of the following form:

$$\begin{aligned} &\psi(n+1) = 0, \\ &\psi(i) = A_i + \min\{\psi(j) + B_j + C_i(D_i - D_j): i < j \le n+1\}, \end{aligned} \tag{20}$$

where A_i, B_i, C_i and D_i are known constants for $1 \le i \le n$.

We now briefly review the main ideas of the geometric technique. An important part of the algorithm is to update a lower convex hull (envelope) of points. To be more precise, at stage i we are given the values $\psi(i+1)$, $\psi(i+2)$, ..., $\psi(n+1)$ to determine the subsequent value $\psi(i)$. Associated with these values, we have a set S_i consisting of the points $(D_{i+1}, \psi(i+1) + B_{i+1})$, $(D_{i+2}, \psi(i+2) + B_{i+2})$, ..., $(D_{n+1}, \psi(n+1) + B_{n+1})$ in a two dimensional plane, where $D_{n+1} = B_{n+1} = 0$. We are interested in the lower part of the convex hull of S_i, also called the lower convex hull, which we denote by H_i. Namely, given the extreme points of H_i, we can evaluate the minimum value of $\min\{\psi(j) + B_j + C_i(D_i - D_j): i < j \le n+1\}$ by finding the line with slope C_i tangent to the lower convex hull, say $\lambda(x) = \psi(j) + B_j + C_i(x - D_j)$ for some j, $i < j \le n+1$, and then computing $\lambda(D_i)$. This can be done in $O(\log n)$ time by binary search, which is an improvement on a straightforward linear approach. We let $eval(C_i, D_i, H_i)$ be the value $\lambda(D_i)$. Hence, the new value of $\psi(i)$ is equal to $A_i + eval(C_i, D_i, H_i)$, leading to a new point $(D_i, \psi(i) + B_i)$.

Then for the set $S_{i-1} = S_i \cup \{(D_i, \psi(i) + B_i)\}$ in the subsequent stage, we need to update the convex hull H_i to obtain H_{i-1}. If D_i is monotone, say increasing, the lower convex hull can be maintained by a stack as a data structure. However, if such monotonicity does not exist for D_i, we need to employ a more complicated data structure like a height balanced or 2-3 tree for efficiency (Van Hoesel et al. [21]). In case of the uncapacitated lot-sizing problem, the constants D_i correspond to the cumulative sum of demands from period i to period T ($d_{i,T}$), which is increasing as period i decreases. Because of this property, updating the lower convex hull takes linear time in total. Hence, Wagelmans et al. [22] were able to solve the unca-

pacitated problem in $O(n\log n)$ time using a stack. Even when D_i is not monotone, problem (20) can still be solved in $O(n\log n)$ time using a 2-3 tree data structure (Van Hoesel et al. [21]).

Unfortunately, the geometric technique cannot be directly applied to the dynamic programming formulation for the bounded inventory problem. As we shall see later, the main reason is that points must be deleted from the lower hull because periods might become infeasible over stages. Hence, we need a deletion operation to update the lower hull. Note that updating the lower hull associated with problem (20) only deals with the insertion of points. Therefore, we need a more general approach than the approach described above.

In the area of computational geometry a significant amount of research has been performed in constructing and maintaining the convex hull of a set of points. The static version of the problem focuses on constructing a convex hull for a given set of points (so all points are known in advance), while the dynamic version concerns updating a convex hull each time when a new point is added to the set or a point is deleted from the set. The maintenance of the lower hull for solving the dynamic program (20) can be considered as a dynamic version of the convex hull problem with insertion operations only. Furthermore, in the context of computational geometry, the problem (20) can be referred to as a dynamic problem with extreme-point query (tangential line query) in a given direction (slope) C_i. Such dynamic problems repeatedly evaluate a given query in each update when a point is added to or removed from a convex hull. The complexity of a dynamic algorithm is determined by the query time and the insertion and deletion times.

Overmars and van Leeuwen [13] show that given a convex hull of n points, the (dynamic) insertion and deletion of a single point can be done in $O(\log^2 n)$ time in worst case (see also Preparata and Shamos [15]). Hence, the overall maintenance of the n points with a basic query like extreme-point query in a given direction takes $O(n\log^2 n)$ time. Brodal and Jacob [5] further improved the algorithm. Their algorithm performs the dynamic maintenance of n points and the basic queries in $O(n\log n)$ time, which means that a single deletion or insertion can be done in $O(\log n)$ amortized time. This result will be used in the next section.

4.2 An Improved $O(T\log T)$ Algorithm for Fixed-charge Costs

The complexity reduction requires efficient computation of the elementary inventory cost $h(s, t)$, $b(s, t)$, $h'(s, t)$ and $b'(s, t)$. From the standard technique of cost decomposition (Aggarwal and Park [2], Van Hoe-

sel et al. [21]), we can rewrite $h(s, t)$ to a more simplified form, $h(s, t) = -h(1, s) + h_{1,s-1} \cdot d_{1,s} + h(1, t) - h_{1,s-1} \cdot d_{1,t}$, which is represented in a more compact form as $h(s, t) = c_s + c_t - h_{1,s-1} \cdot d_{1,t}$ where c_s and c_t are constants only depending on periods s and t, respectively (in this case, $c_s = -h(1, s) + h_{1,s-1} \cdot d_{1,s}$ and $c_t = h(1, t)$). Since the terms c_s and c_t, $h_{1,s-1}$ and $d_{1,t}$ are obtained in linear time, we can compute $h(s, t)$ in $O(1)$ time for a given s and t. Similarly, we can represent the other three costs in the following forms:

$$b(s, t) = c_s + c_t - d_{1,s-1} b_{1,t-1},$$
$$h'(s, t) = c_s + c_t + (d_{1,s-1} + u_s) h_{1,t}, \text{ and}$$
$$b'(s, t) = c_s + c_t - (d_{1,s-1} + u_s) b_{1,t-1},$$

where c_s (c_t) again are constants only depending on period s (period t). From now on, we assume all the inventory related costs are preprocessed in $O(T)$ time.

We will slightly change the recursion formulas of $f(\cdot)$, $F_j(\cdot)$ and $G_j(\cdot)$ in the previous section to define their appropriate functions for lower convex hulls. Based on the lower hulls, we will provide a systematic procedure to compute the values $f(\cdot)$, $F_j(\cdot)$ and $G_j(\cdot)$. We start with $f(i)$ in (11).

Lower convex hull for $f(i)$. Because $h(i, t) = c_i + c_t - h_{1,i-1} \cdot d_{1,t}$ and $d_{i,t} = d_{1,t} - d_{1,i-1}$, we can change $f(i)$ in (11) into the following form:

$$f(i) = A_i + \min\{F(t+1) + B_t - (p_i - h_{1,i-1})(d_{1,i-1} - d_{1,t}) : i \le t \le n(i)\}, \tag{21}$$

where A_i and B_t are appropriate constants relying only on periods i and t, respectively, which can be computed in $O(1)$ time when needed. Furthermore, let $H_f(i, n(i))$ be the lower hull of the points in $\{(d_{1,t}, F(t+1) + B_t) : i \le t \le n(i)\}$. Using the hull $H_f(i, n(i))$, we can obtain the value $f(i)$ by the following equation

$$f(i) = A_i + eval(-(p_i - h_{1,i-1}), d_{1,i-1}, H_f(i, n(i))). \tag{22}$$

Lower convex hulls for $F_1(s)$ and $F_2(s)$. By the equation $b(s, i) = c_s + c_i - d_{1,s-1} b_{1,i-1}$ and the fact that $d_{s,i-1} = d_{1,i-1} - d_{1,s-1}$ we can rewrite $F_1(s)$ in (12) as

$$F_1(s) = A_s' + \min\{f(i) + B_i' + d_{1,s-1}(p_s + b_{1,s-1} - (p_i + b_{1,i-1})) : s \le i \le T\},$$

and $F_2(s)$ in (13) as

$$F_2(s) = A_s'' + \min\{G(i) + B_i'' + d_{1,s-1}(p_s + b_{1,s-1} - (p_i + b_{1,i-1})): s \le i \le T\},$$

where the constants A_s', B_i', A_s'' and B_i'' can all be obtained in $O(1)$ when needed. Then, we define the corresponding hulls $H_{F_1}(s, T)$ and $H_{F_2}(s, T)$ of $F_1(s)$ and $F_2(s)$ for the sets of points

$$\{(p_i + b_{1,i-1}, f(i) + B_i'): s \le i \le T\} \text{ and } \{(p_i + b_{1,i-1}, G(i) + B_i''): s \le i \le T\}.$$

Given the hulls $H_{F_1}(s, T)$ and $H_{F_2}(s, T)$, we get the value $F(s)$ by

$$\begin{aligned} &F(T+1) = 0, \\ &F(s) = \min\{A_s' + eval(d_{1,s-1}, p_s + b_{1,s-1}, H_{F_1}(s, T)), A_s'' + eval(d_{1,s-1}, p_s + b_{1,s-1}, H_{F_2}(s, T))\}. \end{aligned} \tag{23}$$

Note that the horizontal axis of the hull $H_{F_1}(s, T)$ (and of $H_{F_2}(s, T)$) corresponds to the cost $p_i + b_{1,i-1}$, while in other lot-sizing papers the horizontal axis corresponds to cumulative demands. This explains why we are able to improve on the $O(T^2)$ algorithm of Liu [10], who also applies the geometric technique to solve the fixed-charge problem.

Lower convex hull for $G_1(s)$. By the formula $h'(s, i-1) = c_s + c_{i-1} + (d_{1,s-1} + u_s)h_{1,i-1}$, and the fact that $d_{s,i-1} - u_s = d_{1,i-1} - d_{1,s-1} - u_s$, we can rewrite $G_1(s)$ in (15) as follows:

$$G_1(s) = A_s^1 + \min\{f(i) + B_i^1 + (u_s + d_{1,s-1})(p_s - h_{1,s-1} - (p_i - h_{1,i-1})): \phi_{f(i)} \ge u_s - d_{s,i-1}, s < i \le n(s)+1\},$$

where A_s^1 and B_i^1 are constants depending only on periods s and i, respectively. Suppose that we have already computed $f(T)$, $f(T-1)$, ..., $f(s+1)$ and hence know the values $\phi_{f(T)}$, $\phi_{f(T-1)}$, ..., $\phi_{f(s+1)}$. We define π_s as the feasible set of periods with respect to the successive period s, i.e., $\pi_s = \{i: \phi_{f(i)} \ge u_s - d_{s,i-1}, s < i \le n(s)+1\}$. We use $H_{G_1}(\pi_s)$ to denote the lower hull of the points in $\{(p_i - h_{1,i-1}, f(i) + B_i^1): i \in \pi_s\}$.

Lower convex hulls for $G_2(s)$–$G_4(s)$. The formulas for the costs $G_2(s)$–$G_4(s)$ in (16)–(18) can be rewritten as

$$G_2(s) = A_s^2 + \min\{f(i) + B_i^2 + (u_s + d_{1,s-1})(\, p_s + b_{1,s-1} - (p_i + b_{1,i-1})) : n(s) + 1 < i \le T\},$$
$$G_3(s) = A_s^3 + \min\{G(i) + B_i^3 + (u_s + d_{1,s-1})(\, p_s - h_{1,s-1} - (p_i - h_{1,i-1})) : s < i \le n(s) + 1\},$$
$$G_4(s) = A_s^4 + \min\{G(i) + B_i^4 + (u_s + d_{1,s-1})(\, p_s + b_{1,s-1} - (p_i + b_{1,i-1})) : n(s)+1 < i \le T\},$$

where A_s^j and B_i^j (j = 2, 3, 4) are constants depending only on periods s and i, respectively, which can be computed in constant time when needed. We use $H_{G_2}(n(s)+2, T)$, $H_{G_3}(s+1, n(s)+1)$, and $H_{G_4}(n(s)+2, T)$ to denote the hulls for the recursions $G_2(s)$, $G_3(s)$ and $G_4(s)$, respectively.

Given the lower hulls $H_{G_1}(\cdot) - H_{G_4}(\cdot)$, we can compute $G(s)$ as follows:

$$G(T) = \infty,$$
$$G(s) = \min \begin{cases} A_s^1 + eval(u_s + d_{1,s-1},\, p_s - h_{1,s-1}, H_{G_1}(\pi_s)), \\ A_s^2 + eval(u_s + d_{1,s-1},\, p_s + b_{1,s-1}, H_{G_2}(n(s)+2, T)), \\ A_s^3 + eval(u_s + d_{1,s-1},\, p_s - h_{1,s-1}, H_{G_3}(s+1, n(s)+1)), \\ A_s^4 + eval(u_s + d_{1,s-1},\, p_s + b_{1,s-1}, H_{G_4}(n(s)+2, T)). \end{cases} \tag{24}$$

Now we provide the formal steps to compute values $f(\cdot)$, $F_j(\cdot)$ and $G_j(\cdot)$ systematically. The algorithm consists of T stages, each requiring at most $O(\log T)$ time. At stage s, suppose that we are given the following values and hulls:

- $f(t)$, $F(t)$, $G(t)$ for $t = s+1, \ldots, T$, and
- $H_f(s, n(s))$, $H_{F_j}(s+1, T)$, $j = 1, 2$, $H_{G_1}(\pi_s)$, $H_{G_3}(s+1, n(s)+1)$, $H_{G_j}(n(s)+2, T)$, $j = 2, 4$.

Then the following steps are executed at stage s:

Step 1. Evaluate $f(s)$ and $G(s)$ by (22) and (24), respectively.

Step 2. With the values $f(s)$ and $G(s)$, construct hulls $H_{F_j}(s, T)$ from $H_{F_j}(s+1, T)$, j = 1, 2. Then, compute $F(s)$ by formula (23) based on the hulls $H_{F_j}(s, T)$, $j = 1, 2$.

Step 3. Construct hulls $H_f(s-1, n(s-1))$, $H_{G_3}(s, n(s-1)+1)$, $H_{G_j}(n(s-1)+2, T)$, j = 2, 4. Obtain the hull $H_{G_1}(\pi_{s-1})$ from $H_{G_1}(\pi_s)$.

We now consider the complexity of the algorithm. The computation of the functional values $f(s)$, $G(s)$ and $F(s)$ in the Steps 1 and 2 of stage s can be done in $O(\log T)$ if the appropriate lower hulls are given. Thus, we focus on the construction of the hulls. Note that most lower hulls consist of points corresponding to an interval of periods except for the hull $H_{G_1}(\pi_s)$. In this case the set π_s is not necessarily an interval of periods. The construction of a hull with points from an interval of periods can be easily implemented. For instance, the creation of $H_f(i-1, n(i-1))$ from $H_f(i, n(i))$ can be carried out by deleting the points $\{(d_{1,t}, F(t+1) + B_t): n(i-1) < t \le n(i)\}$ and then inserting the (single) point $(d_{1,i-1}, F(i) + B_{i-1})$ into $H_f(i, n(i))$. During the execution of the stages from T to 1, each point is added (or deleted) at most once. Every step can be done in $O(\log T)$ amortized time by using the algorithm of Brodal and Jacob [5], implying that the total computing time in association with the hulls $H_f(i, n(i))$ is $O(T\log T)$. Similarly, the constructions of the hulls corresponding to the costs $F_1(\cdot)$, $F_2(\cdot)$ and $G_2(\cdot)$–$G_4(\cdot)$ take at most $O(T\log T)$ time.

We finally consider the time complexity with regard to the hull $H_{G_1}(\cdot)$ of the cost $G_1(\cdot)$. Since the set π_s $(= \{i: \phi_{f(i)} \ge u_s - d_{s,i-1}, s < i \le n(s)+1\})$ may not be an interval of periods, we should be careful in adding and deleting points in the transition from $H_{G_1}(\pi_s)$ to $H_{G_1}(\pi_{s-1})$. We assume that the set π_s of the hull $H_{G_1}(\pi_s)$ is ordered by period number so that we can easily remove points from the set π_s that violate the second condition of π_{s-1}, $s-1 < i \le n(s-1)+1$. For the addition and deletion in association with the first condition $\phi_{f(i)} \ge u_s - d_{s,i-1}$, we define π_s' as a set of infeasible periods i with respect to period s, i.e. periods i with $\phi_{f(i)} < u_s - d_{s,i-1}$ for each $i = s, s+1, ..., T$. The set π_s' is ordered by non-increasing values of $\phi_{f(i)}$. Then, we remove each period i with $\phi_{f(i)} \ge u_{s-1} - d_{s-1,i-1}$ from the set π_s', and if $s-1 < i \le n(s-1)+1$, insert it into π_s. Note that this will happen at most once for every period i. Finally, for period $s-1$, if $\phi_{f(s)} < u_{s-1} - d_{s-1}$, then we insert s into the set π_s' with value $\phi_{f(s)}$ to obtain π_{s-1}'. Otherwise, we insert s into the set π_s to obtain π_{s-1}. We note that during the process of making π_{s-1} from π_s, we also obtain the new lower hull $H_{G_1}(\pi_{s-1})$. Since all the addition and deletion operations can be executed in $O(T\log T)$, the time to construct $H_{G_1}(\cdot)$ is at most $O(T\log T)$. We, therefore, conclude that the algorithm consisting of Steps 1–3 can be implemented in $O(T\log T)$ time.

Theorem 2: The lot-sizing problem with inventory bounds, backlogging and fixed-charge costs can be solved in $O(T\log T)$ time.

5. Nonspeculative Costs

This section first refines the formulas for the fixed-charge costs in the previous section to appropriate forms for the nonspeculative costs. In spite of the simplified formulas, it seems still not easy to obtain an improved algorithm over the $O(T\log T)$ algorithm for the fixed-charge costs if we use the points-approach. This also suggests that the lines-approach, a dual method of the points-approach, which maintains a lower envelope of lines, does not lead to an efficient algorithm. However, with a slight generalization of the lines-approach, which we will refer to as the *line-segments-approach*, we can get an efficient $O(T)$ algorithm for the nonspeculative costs. With the refined formulas in Section 5.1, we shortly introduce the lines-approach in Section 5.2 to better understand the line-segments-approach, which will be described in Section 5.3.

5.1 Adjustment of the Fixed-charge Procedure for Nonspeculative Costs

From the characteristics of the nonspeculative cost structure, $p_t - b_t \le p_{t+1} \le p_t + h_t$ for each $t = 1, 2, \ldots, T-1$, we have a useful structural property, allowing for a more efficient implementation.

Property 4. Under a nonspeculative cost structure, there exists an optimal schedule such that if t is a production period, then $I_{t-1}^{+} = I_t^{-} = 0$.

This property implies that if we have a production in period t, there is no *in-flow* supply from productions of other periods and so the demand d_t is fulfilled by its own production. Suppose that period s is a successive period with its next production period at i. Since $I_{i-1}^{+} = 0$ by Property 4, the u_s units at the beginning of period s ($I_{s-1} + x_s = u_s$) cannot completely satisfy demands up to period $i-1$. This implies that $u_s < d_{s,i-1}$, and hence $n(s)+1 < i \le T$. Hence, the cost $G_1(s)$ of formula (15) and the cost $G_3(s)$ of formula (17) are not needed so that the formula (24) for $G(s)$ is modified to

$$G(T) = \infty,$$
$$G(s) = \min \begin{cases} A_s^2 + eval(u_s + d_{1,s-1}, p_s + b_{1,s-1}, H_{G_2}(n(s)+2, T)), \\ A_s^4 + eval(u_s + d_{1,s-1}, p_s + b_{1,s-1}, H_{G_4}(n(s)+2, T)). \end{cases}$$

During the construction of convex hulls $H_{G_2}(\cdot)$ and $H_{G_4}(\cdot)$, points are only added; no points are deleted. Since $u_s + d_{1,s-1}$ and $p_s + b_{1,s-1}$ are decreasing as s decreases due to the nonspeculative cost assumption, we can use a stack to maintain the lower convex hulls. For the same reason, we can also use a stack for keeping the extreme points of the lower hulls $H_{F_1}(\cdot)$ and $H_{F_2}(\cdot)$. Thus, we can compute all the values regard-

ing the lower hulls $H_{G_2}(\cdot)$, $H_{G_4}(\cdot)$, $H_{F_1}(\cdot)$ and $H_{F_2}(\cdot)$ in $O(T)$ if we utilize the results in Wagelmans et al. [22].

Finally, we consider the construction of the convex hull $H_f(\cdot)$ for the cost $f(\cdot)$. In this case we should take into account the deletion of points from the hull, since the set of points associated with $H_f(i, n(i))$ consists of the interval of periods $[i, n(i)]$ where $n(i)$ might not be T. Because of this deletion operation on $H_f(i, n(i))$, it seems not easy to not improve the geometric algorithm based on the points-approach. Although this deletion operation is considered in Liu [10], it turns out that his algorithm is incorrect. It can be shown that, after the deletion operation and updating the left part of the hull, it may not be convex anymore (Onal [12]). This means that the algorithm does not find an optimal solution in general.

5.2 Geometric Technique Based on the Lines-approach

The geometric technique based on the lines-approach maintains a lower envelope of lines instead of points. Given a set of linear functions $\lambda_t(x) = y_t + r_t x$ with intercept y_t and slope r_t for $t \in \pi$ where π is a set of finite indices (periods), the envelope is given by the minimum of the lines: $env(x|\pi) = \min\{\lambda_t(x): t \in \pi\}$. Note that the $env(x|\pi)$ is a piecewise linear concave function which is described by at most $|\pi| - 1$ *breakpoints* and where $|\pi|$ denotes the number of elements of π. To use the lines-approach, we adjust formula (21) in an appropriate form as follows:

$$f(i) = A_i + \min\{F(t+1) + B_t + d_{1,t}(p_i - h_{1,i-1}): i \le t \le n(i)\},$$

where A_i and B_t denote constants depending only on periods i and t, respectively. Suppose that we are given values $F(t+1)$ for $t = T, T-1, ..., i$. The linear functions $\lambda_t(x)$ in association with $f(i)$ are then defined as $\lambda_t(x) = F(t+1) + B_t + d_{1,t}x$, referred to as the *line of period t*, for $i \le t \le n(i)$. The envelope of these lines is given as

$$env_f(x|i, n(i)) = \min\{F(t+1) + B_t + d_{1,t}x: i \le t \le n(i)\}.$$

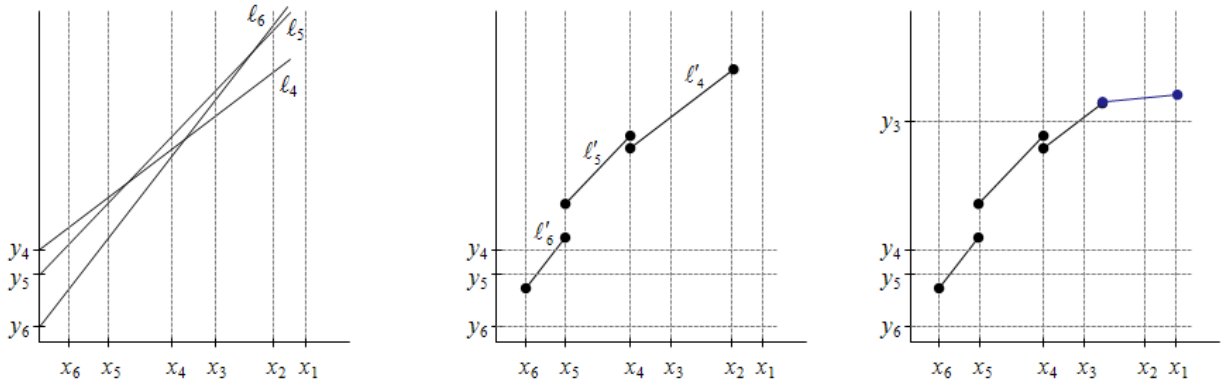
We note that the index set π of the envelope is given here by the interval $[i, n(i)]$ where we removed the brackets in the above formula for notational convenience. Using the envelope function $env_f(\cdot)$, we can obtain the value $f(i)$ by the following equation

$$f(i) = A_i + env_f(p_i - h_{1,i-1}|i, n(i)).$$

As period index t decreases, we note that the slope $d_{1,t}$ of the period decreases since $d_{1,t-1} \le d_{1,t}$ while the query value $p_t - h_{1,t-1}$ of the period increases due to the nonspeculative cost assumption of $p_t \le p_{t-1} + h_{t-1}$. Clearly, if $n(i) = T$, then we can solve the recursion for $f(i)$ in $O(T)$ using the result in Van Hoesel et al. [21].

5.3 An $O(T)$ Algorithm for Nonspeculative Costs

The ordinary lines-approach does not apply to the inventory bounded problem in which $n(i)$ is not equal to T, which requires the deletion of lines from an envelope. Consider a 6-period problem for which the first three lines λ_6, λ_5 and λ_4 are given as in Figure 1(a) where the intercept $F(t+1) + B_t$ of line λ_t is denoted as y_t and the query value $p_t - h_{1,t-1}$ is denoted as x_t. If period 6 is reachable from period 4 (i.e., $6 \le n(4)$), we can find $env_f(x_4|4, n(4))$ using line λ_6: $env_f(x_4|4, n(4)) = y_6 + d_{1,6} \cdot x_4$ because line λ_6 is the lowest for query value x_4. On the other hand, if $6 > n(4)$, we cannot use the envelope but we have to delete the line λ_6 from the envelope. This removal causes that latent breakpoints may be exposed, which seems to prevent a linear time implementation. In Figure 1(a), the removal of line λ_6 exposes a breakpoint between lines λ_5 and λ_4. To avoid such difficulty, we use a line-segments-approach, a generalized version of the lines-approach, which will allow us to focus on insertion operations only (and hence no deletion operation).



(a) (b) (c)

Figure 1. Envelope of line segments.

Note that line λ_6 is used only for evaluating $f(i)$ for each i such that $6 \in [i, n(i)]$. In other words, line λ_6 is used only for $f(i)$ such that $i \in [m(6), 6]$. This means that the valid domain of line λ_6 in terms of x-coordinate is the interval $[p_6 - h_{1,5}, p_{m(6)} - h_{1,m(6)-1}]$. Hence from now on we are not interested in the entire

domain of a line but in its valid line segment. Figure 1(b) shows the segment of line λ_6 when the valid domain is $[x_6, x_5]$, which is the case when $m(6) = 5$. Furthermore, the segments of lines λ_5 and λ_4 are also illustrated for the cases when $m(5) = 4$ and $m(4) = 2$. For each line λ_t we define its valid line segment ℓ'_t by $F(t+1) + B_t + d_{1,t}x$ where $x \in [p_t - h_{1,t-1}, p_{m(t)} - h_{1,m(t)-1}]$. Then we can find $f(i)$ by using the lower envelope of these line segments so that we let

$$env_f(x|i, n(i)) = \min\{ \ell'_t(x) : i \le t \le n(i)\} \text{ for } x \in [p_i - h_{1,i-1}, p_{m(i)} - h_{1,m(i)-1}].$$

It should be noted that the envelope is not necessarily continuous (and concave), However, the envelope is piecewise linear, non-decreasing and left continuous on its domain. Moreover, the slopes of the pieces are non-increasing on the domain when starting from the left. Furthermore, every piece starts and ends at (i) a query value, or (ii) at the intersection point of two line segments. For instance, the envelope in Figure 1(b) has three pieces represented by intervals $[x_6, x_5]$, $[x_5, x_4]$ and $[x_4, x_2]$ and every piece starts and ends at a query value. However, the envelope of Figure 1(c), obtained after the insertion of line segment ℓ'_3, has an interval $[x_4, x_1]$ consisting of two pieces, where the first piece ends and the second piece starts at the intersection of the line segments ℓ'_4 and ℓ'_3.

Suppose that we are given $env_f(x|i, n(i))$ with its intervals corresponding to the pieces. We deal with the insertion of a line segment ℓ'_{i-1} of period $i - 1$ with its domain $[x', x'']$. Recall that x' and x'' correspond to $p_{i-1} - h_{1,i-2}$ and $p_{m(i-1)} - h_{1,m(i-1)-1}$, respectively. Let $[x_i', x_i'']$ be the domain of the last piece in the envelope. Note that $x_i'' \le x''$ as $p_i - h_{1,i-1}$ is non-decreasing as i decreases. Now there are two possibilities: line segment ℓ'_{i-1} does or does not intersect the last piece. In the first case, (part of) line segment ℓ'_{i-1} is inserted in the lower envelope and we are done. Note that ℓ'_{i-1} will not intersect any other piece as the envelope is non-decreasing and the slopes are non-increasing on the relevant domain. In the latter case, the last piece can be removed from the envelope and the next piece needs to be considered. Again, line segment ℓ'_{i-1} does or does not intersect this piece and we either insert (part of) line segment ℓ'_{i-1} or we remove the piece. We continue in this way until (part of) the segment ℓ'_{i-1} is inserted. Note that every piece is removed only once. So the construction of the envelopes can be done in linear time when using a stack to maintain the envelope. Hence, we can obtain $env_f(x|i-1, n(i-1))$ from $env_f(x|i, n(i))$ in amortized constant time.

Furthermore, the evaluation of $env_f(x'|i-1, n(i-1))$ for the query value $x' = p_i - h_{1,i-1}$ can be done in amortized constant time as well. Namely, for the evaluation of a query value x', we start at the left most piece and move to the right until we find the piece that contains x'. In the next iteration we start from this piece and again we move to the right until we find the piece that contains the query value. It not difficult to see that this can be done in linear time, when a stack is used to maintain the lower envelope. Hence, we can evaluate every $f(i)$ in $O(T)$ time.

Following steps 1–3 in Section 4.2 and combining the computations of $env_f(\cdot)$ using the line-segments-approach with those of the convex hulls $H_{G_2}(\cdot)$, $H_{G_4}(\cdot)$, $H_{F_1}(\cdot)$ and $H_{F_2}(\cdot)$ using the points-approach, we can solve the problem with nonspeculative costs in $O(T)$ time.

Theorem 3: The lot-sizing problem with inventory bounds, backlogging and nonspeculative costs can be solved in $O(T)$ time.

6. Conclusion

In this paper we solved the bounded inventory lot-sizing problem with backlogging for three different cost structures by applying well-known matrix-searching and geometric techniques for the uncapacitated lot-sizing problem. By identifying the Monge property under a concave cost structure, we were able to develop an $O(T^2)$ time algorithm based on the matrix-searching technique. For the fixed-charge problem we presented an $O(T \log T)$ algorithm using the points-approach, a geometric technique of maintaining a lower convex envelope of points. In addition, we provided an $O(T)$ algorithm for the nonspeculative problem using a line-segments-approach, a geometric technique that maintains a lower convex envelope of line segments.

We can conclude that the inventory bounded lot-sizing problem is not harder than the traditional uncapacitated lot-sizing problem from a complexity point of view. This may suggest that more generalizations of the uncapacitated lot-sizing problem can be solved efficiently when the storage constraint is imposed. One such problem is the lot-sizing problem with constant production capacity, which can be solved in polynomial time if we have no storage limitation. Wolsey [24] provides an $O(T^3)$ time algorithm for the lot-sizing problem with production and storage capacity limitation under a fixed-charge cost structure. It is interesting to investigate whether there are algorithms with the same complexity as those for the capacitated-lot-sizing problems for the various cost structures. Finally, we need to mention that our problem is a special case of the original bounded inventory problem of Love [11], since Love considered not only a

bound on the on-hand inventory level but also a bound on the backlogging level. Although our problem is restricted to the inventory level, the results in this paper may possibly be extended to the more general problem.

Acknowledgements

We appreciate the help of Dr. Riko Jacob regarding our questions on the dynamic convex hull algorithms. Furthermore, we thank Mehmet Onal for pointing out the mistake in Liu [10].

References

[1] A. Aggarwal, M.M. Klawe, S. Moran, P.W. Shor and R. Wilber, Geometric applications of a matrix-searching algorithm. Algorithmatica 2 (1987), 195-208.

[2] A. Aggarwal and J.K. Park, Improved algorithms for economic lot-size problems. Operations Research 41 (1993), 549-571.

[3] A. Atamtürk and S. Küçükyavuz, Lot sizing with inventory bounds and fixed costs: Polyhedral study and computation, Operations Research 53 (2005), 711-730.

[4] A. Atamtürk and S. Küçükyavuz, An $O(n^2)$ algorithm for lot sizing with inventory bounds and fixed costs. Operations Research Letters 36 (2008), 297-299.

[5] G.S. Brodal and R. Jacob, Dynamic planar convex hull, in: Proc. 43rd Annual Symp. on Foundations of Computer Science, 2002, pp. 617-626.

[6] A. Federgruen and M. Tzur, A simple forward algorithm to solve general dynamic lot-sizing models with n periods in $O(n \log n)$ or $O(n)$ Time. Management Science 37 (1991), 909–925.

[7] J. Gutiérrez, A. Sedeño-Noda, M. Colebrook and J. Sicilia, A new characterization for the dynamic lot size problem with bounded inventory. Computers & Operation Research 30 (2003), 383-395.

[8] J. Gutiérrez, A. Sedeño-Noda, M. Colebrook and J. Sicilia, A polynomial algorithm for the production/ordering planning problem with limited storage. Computers & Operation Research 34 (2007), 934-937.

[9] J. Gutiérrez, A. Sedeño-Noda, M. Colebrook and J. Sicilia (2008) An efficient approach for solving the lot-sizing problem with time-varying storage capacities. European Journal of Operational Research 189, 682-693.

[10] T. Liu, Economic lot sizing problem with inventory bounds. European Journal of Operational Research 185 (2008), 204-215.

[11] S.F. Love, Bounded replenishment and inventory models with piecewise concave costs. Management Science 20 (1973), 313-318.

[12] M. Onal, Private communication (2010).

[13] M.H. Overmars and J. van Leeuwen, Maintenance of configurations in the plane, Journal of Computer and System Sciences 23 (1981), 166-204.

[14] Y. Pochet and L.A. Wolsey, Polyhedra for lot-sizing with Wagner-Whitin costs. Mathematical Programming 67 (1994), 297-323.

[15] F.P. Preparata and M.I. Shamos, □*Computational geometry, an introduction*□, Springer-Verlag, New York, 1985.

[16] K. Richter and M. Sombrutzki, Remanufacturing planning for the reverse Wagner/Whitin models, European Journal of Operational Research 121 (2000), 304–315.

[17] F.Z. Sargut and H.E. Romeijn, Lot-sizing with non-stationary cumulative capacities, Operations Research Letters 35 (2007), 549–557.

[18] E. Toczylowski, An $O(T^2)$ algorithm for the lot-sizing problem with limited inventory levels. IEEE Symposium on Emerging Technologies & Factory Automation 3, 1995, pp. 78–85.

[19] W. Van den Heuvel and A.P.M. Wagelmans, Four equivalent lot-sizing models. Operations Research Letters 36 (2008), 465–470.

[20] W. Van den Heuvel, J. Gutierrez and H.-C. Hwang, A note on "An efficient approach for solving the lot-sizing problem with time-varying storage capacities". (in preparation), 2010.

[21] C.P.M. Van Hoesel, A.P.M. Wagelmans and B. Moerman, Using geometric techniques to improve dynamic programming algorithms for the economic lot-sizing problem and extensions. European Journal of Operational Research 75 (1994), 312–331.

[22] A.P.M. Wagelmans, S. Van Hoesel and A. Kolen, Economic lot sizing: an $O(n \log n)$ algorithm that runs in linear time in the Wagner-Whitin case. Operations Research 40 (1992), S145–S156.

[23] H.M. Wagner and T.M. Whitin, Dynamic version of the economic lot-size model. Management Science 5 (1958), 89–96.

[24] L.A. Wolsey, Lot-sizing with production and delivery time windows. Mathematical Programming Series A 107 (2006), 471-489.

[25] W.I. Zangwill, A deterministic multi-period replenishment scheduling model with backlogging *Management Science* 13 (1966), 105–119.

[26] W.I. Zangwill, Minimum concave cost flows in certain networks *Management Science* 14 (1968), 429-450.