mathematics

Article

Effective Algorithms for the Economic Lot-Sizing Problem with Bounded Inventory and Linear Fixed-Charge Cost Structure

José M. Gutiérrez *, Beatriz Abdul-Jalbar, Joaquín Sicilia and Inmaculada Rodríguez-Martín

Departamento Matemáticas, Estadística e Investigación Operativa, Universidad de La Laguna, 38200 Santa Cruz de Tenerife, Spain; babdul@ull.edu.es (B.A.-J.); jsicilia@ull.es (J.S.); irguez@ull.edu.es (I.R.-M.)
* Correspondence: jmgrrez@ull.es

Abstract: Efficient algorithms for the economic lot-sizing problem with storage capacity are proposed. On the one hand, for the cost structure consisting of general linear holding and ordering costs and fixed setup costs, an $\mathcal{O}(T^2)$ dynamic programming algorithm is introduced, where T is the number of time periods. The new approach induces an accurate partition of the planning horizon, discarding most of the infeasible solutions. Moreover, although there are several algorithms based on dynamic programming in the literature also running in quadratic time, even considering more general cost structures and assumptions, the new solution uses a geometric technique to speed up the algorithm for a class of subproblems generated by dynamic programming, which can now be solved in linearithmic time. To be precise, the computational results show that the average occurrence percentage of this class of subproblems ranges between 13% and 45%, depending on both the total number of periods and the percentage of storage capacity availability. Furthermore, this percentage significantly increases from 13% to 35% as the capacity availability decreases. This reveals that the usage of the geometric technique is predominant under restrictive storage capacities. Specifically, when the percentage of capacity availability is below 50%, the average running times are on average 100 times faster than those when this percentage is above 50%. On the other hand, an $\mathcal{O}(T)$ on-line array searching method in Monge arrays can be used when the costs are non-speculative costs.

Keywords: lot-sizing; dynamic programming; geometric technique; Monge arrays

Citation: Gutiérrez, J.M.; Abdul-Jalbar, B.; Sicilia, J.; Rodríguez-Martín, I. Effective Algorithms for the Economic Lot-Sizing Problem with Bounded Inventory and Linear Fixed-Charge Cost Structure. *Mathematics* **2021**, *9*, 689. https://doi.org/10.3390/math9060689

Academic Editor: Armin Fügenschuh

Received: 2 February 2021
Accepted: 16 March 2021
Published: 23 March 2021

1. Introduction

The economic lot-sizing problem is a classical deterministic inventory problem which was independently introduced by Wagner and Whitin (abbreviated W-W) [1] and Manne [2] as an extension of the economic order quantity (EOQ) model to cases with time-varying parameters. Accordingly, the goal for this problem consists of determining an optimal procurement schedule that satisfies the demands of time for each of the T time periods in which the planning horizon is divided. The cost structure defined in [1] is the sum of the fixed charge replenishment costs and the linear holding costs for each period. It is a well-known result that among the optimal solutions for the dynamic version of the EOQ, there is at least one satisfying the zero inventory ordering (ZIO) policy (i.e., an order is placed only if the inventory is depleted, which can be used to derive polynomially efficient solutions). Later, Veinott [3] proved that ZIO policies could be likewise used to derive optimal plans when the cost structure was concave in general. Since then, a significant number of contributions have been published in the literature considering diverse extensions of this simple model (refer to Brahimi et al. [4] and Wolsey [5] for more comprehensive surveys and to [6–10] for more recent contributions).

One of these extensions introduces bounds on the inventory levels as a consequence of the storage capacities. This version constitutes a subfamily of lot-sizing problems usually referred to as either the economic lot-sizing problem with bounded inventory (ELSPBI) or the economic lot-sizing problem with storage capacities (ELSPSC), and unfortunately, it has attracted poor attention from researchers. Throughout, the acronym ELSPBI instead of ELSPSC is used to refer to this family of problems. The importance of the ELSPBI lies primarily in its applicability in the context of material requirements planning (MRP) to improve productivity for businesses and industries. In this scenario, the decision-maker is interested in scheduling the production requirements of subassemblies (components, packaging, spare parts and raw materials involved in the manufacturing process) at the different levels within a multi-echelon system, after the firm (organization) determines or estimates the demand of finished products. Since there is normally a significant number of subassemblies involved in the manufacturing processes (e.g., automobile, pharmaceutic and aerospace industries, among others), which do not usually share the same storage area, efficient algorithms for the ELSPBI are essential to reduce the computing times needed to obtain optimal replenishment plans.

The first contribution devoted to the ELSPBI is credited to Love [11], who allowed shortages in the model and introduced a characterization of the optimal plans to derive an $\mathcal{O}(T^3)$ algorithm based on dynamic programming (the big O notation $\mathcal{O}(g(n))$ denotes the asymtotic behavior of an algorithm, and it is used to classify them according to how their run time or space requirements grow as the input size n grows. Accordingly, $f(n) = \mathcal{O}(g(n))$ if there exist $M, n_0 \in \mathbb{N}$ such that $f(n) \leq Mg(n)$ for all $n \geq n_0$. Thus, $|f|$ is bounded above by g (up to a constant factor) asymptotically.). Gutiérrez et al. in [12] and [13] addressed the problem, providing a new characterization and hence a new algorithm to identify optimal plans for the model with and without backorders, respectively. Even though the theoretical time complexity of the algorithm by Love could not be reduced, the computational experiment in [12] revealed that the running times were almost 30% faster than those reported by Love's method.

In the last few decades, several time complexity improvements have been presented. For instance, Toczylowski [14] transformed the ELSPBI into an equivalent shortest-path problem in a multigraph and developed an $\mathcal{O}(T^2)$ algorithm to construct the shortest path solution. Nevertheless, even though the new approach shares the same asymptotic running time, it uses an $\mathcal{O}(T \log T)$ geometric technique inspired by the algorithm developed by Hershberger [15] to handle the class of subproblems referred to as *elementary nonsingular plans* with descending edges, defined in [14]. Basically, a subproblem belongs to this class when its initial period is successor to a breaking period (a period where the maximum quantity of items to be stored corresponds to the physical storage capacity in that period) and its incoming inventory level is positive.

Sedeño-Noda et al. [16] devised an $\mathcal{O}(T \log T)$ algorithm for a case without backorders or setup costs and linear ordering and holding costs. More recently, Atamtürk and Kücükyavuz [17] developed an $\mathcal{O}(T^2)$ algorithm for a case with linear and fixed holding and ordering costs and without backlogging. Their algorithm is based on the definition of a *block*, which is simply a sequence of consecutive periods satisfying certain features of the inventory levels. However, the new method computes the objective value of those blocks with a positive initial inventory level by using an $\mathcal{O}(T \log T)$ geometric method, whereas the algorithm in [17] takes $\mathcal{O}(T^2)$ time.

Taking advantage of the geometrical technique in Wagelmans et al. [18], Liu [19] and Gutiérrez et al. [20] proposed, respectively, $\mathcal{O}(T^2)$ and $\mathcal{O}(T \log T)$ algorithms assuming linear order and holding costs and setup costs, as well as backorders in the former model. Moreover, Liu [19] devised an $\mathcal{O}(T)$ method for the Wagner and Whitin (W-W) cost structure (non-speculative motives). However, Önal et al. [21] and van den Heuvel et al. [22] demonstrated independently that both the algorithm by Liu and the algorithm by Gutiérrez et al. might fail to find an optimal solution in general.

Hwang and van den Heuvel [23] addressed a general concave cost structure for procurement and holding and backlogging, and they presented a dynamic programming algorithm that runs in $\mathcal{O}(T^2)$ by using matrix-searching in a Monge array. As noted above, the algorithm proposed in this paper is also based on dynamic programming and also runs in $\mathcal{O}(T^2)$ amortized time, but it uses an $\mathcal{O}(T \log T)$ geometric method for a class of subproblems. Concretely, this class of subproblems is related to the concept of *warehouse point* in [23]. According to the authors, a period is said to be a warehouse point when there are inventory units available at the beginning of that period.

Most recently, Wolsey [24] derived a versatile and tight model based on a shortest path formulation that was also able to account for stock fixed costs, stock overloads and backlogging. Since the unit flow polyhedron resulting from the constraints was totally unimodular, the extreme points were binary. The author also showed that there was an $\mathcal{O}(T^2)$ algorithm that solved the problem.

The following list summarizes the most significant contributions to the ELSPBI in the literature (all of them, including the algorithm proposed in this paper, run in quadratic amortized time) along with the corresponding nomenclature of the subproblems, for which the new algorithm outperforms the existing methods. This improvement lies mainly in the use of the $\mathcal{O}(T \log T)$ geometric technique instead of using a straightforward $\mathcal{O}(T^2)$ implementation from the dynamic programming, as in those papers.

Reference	**Subproblem Nomenclature**
E. Toczylowski [14]	Elementary nonsingular plan
A. Atamtürk, S. Kücükyavuz [17]	Block with positive initial inventory level
H.C. Hwang, W. van den Heuvel [23]	Warehouse point

The new approach is based on what we have called *breaking periods* (periods when the inventory level could reach its physical storage capacity), which induces a partition of the planning horizon so that each part (subpartition) can be computed in quadratic time. However, the main contribution in this paper is that when a subpartition (subproblem) fulfils the same assumptions of either an elementary nonsingular plan as in [14] or a block as in [17] with a positive initial inventory level, or the subpartition starts with a warehouse point as in [23], the subproblem can be solved in $\mathcal{O}(T \log T)$ time by means of a geometric technique, rather than $\mathcal{O}(T^2)$ as in the previous papers. This kind of subpartition plays a relevant role in the computation of an optimal plan, since the lower the storage capacity, the greater the number of occurrences of these subproblems. Specifically, our computational experiment shows that the average occurrence percentage of this class of subproblems (i.e., the average usage rate of the geometric technique) ranges between 13% and 45%, depending on both the total number of periods and the percentage of storage capacity availability. Furthermore, this percentage increases significantly from 13% to 35% as the percentage of storage capacity availability decreases. Moreover, the computational experiment also shows that when T is fixed and the percentage of capacity availability is below 50%, the average running times are on average 100 times faster than those reported when this percentage is above 50%, due to the intensive use of the geometric technique. Furthermore, the new algorithm discards a large fraction of (infeasible) subproblems generated by the dynamic programming by defining upper bounds on each interval of periods.

For the case with a non-speculative cost structure, Hwang and van den Heuvel [23] claimed that the problem could be solved in $\mathcal{O}(T)$ by employing a geometric technique. However, in this paper, an on-line array-searching method in Monge arrays is proposed to obtain, in linear time, an optimal plan considering the same cost structure.

The natural extension of the single-item ELSPBI also considering the bounds on the production and replenishment quantity at each period has been addressed recently by Akbalik et al. [25]. Concretely, for the single-item CLSP-IB with a constant production capacity, time-dependent inventory bounds and concave costs, the authors showed that the problem could be solved in time $\mathcal{O}(T^4)$. They proposed an $\mathcal{O}(T^3)$ time dynamic programming algorithm for non-speculative costs.

The goal of this paper is to develop efficient algorithms for the ELSPBI with linear holding and ordering costs and set-up (fixed-charge) costs in each period. Firstly, for the general linear cost structure, the original problem is decomposed into subproblems using the notion of breaking periods. Then, some properties are introduced to effectively define recurrence equations, thereby obtaining a more precise description of the feasible set. Secondly, when the cost structure is non-speculative, it is shown that there exists an optimal plan satisfying the zero inventory ordering (ZIO) policy. Moreover, the array containing the optimal costs of all subproblems is Monge, and hence an on-line array-searching algorithm can be used to construct an optimal plan in $\mathcal{O}(T)$.

The rest of the article is organized as follows. In Section 2, the problem formulations are defined, and the procedure to decompose the original problem into subproblems is also introduced. The recurrence equations restricted to the subproblems generated by the decomposition method, as well as the procedure to implement such equations, are presented in Section 3. Furthermore, in Section 4, the dynamic programming algorithm for the ELSPBI is sketched, and a numerical example to illustrate the solution is provided. The case with the Wagner and Whitin cost structure is analyzed in Section 5. In Section 6, a computational experiment to validate the complexity and the performance of the new method is carried out. Finally, some final remarks are discussed in Section 7.

2. Problem Formulations and Breaking Periods

The ELSPBI concerns the scheduling of an optimal replenishment plan at a minimum cost that satisfies the demand in each period on time without exceeding the storage capacity. It is assumed that both the replenishment and the withdrawal of items are instantaneously completed at the beginning of each period at the warehouse.

Before formulating the ELSPBI, the following notation is introduced. Let $H = \{1, \ldots, T\}$ be the set of time periods. Throughout, let d_t, S_t and $D_t = \sum_{j=t}^{T} d_j$, $t \in H$ denote the demand and physical storage capacity in period t and the cumulated demand from period t to period T, respectively. Thus, the cumulated demand between two periods t and t' can be easily computed as $D_{t,t'} = D_t - D_{t'+1}$, where $D_{T+1} = 0$. Moreover, the cost structure is assumed to be the sum of a setup (fixed charge) cost $f_t > 0$ and linear replenishment and holding costs p_t and h_t through the planning horizon. Initially, no assumption is made about the speculative nature of the linear costs (i.e., it is assumed to have a general linear cost structure). The case with a non-speculative cost structure is discussed later. Additionally, the decision variables representing the order quantity in period t and its corresponding binary variable are denoted by x_t and y_t ($y_t = 0$ if $x_t = 0$; $y_t = 1$ otherwise), respectively. Likewise, the variable I_t denotes the inventory level at the end of the period t. The list below summarizes the parameters and variables for each period $t \in H$ used to formulate the problem. Moreover, the list also provides the notation of the subproblems to be solved by dynamic programming and the data structures used in the implementation of the algorithm.

Parameters	
T	periods of the planning horizon
$H = \{1, \ldots, T\}$	set of indices of time periods
d_t	demand for period t
$D_t = \sum_{j=t}^{T} d_j$	cumulated demand from period t to period T
$D_{t,t'} = D_t - D_{t'+1}$	cumulated demand between periods $t \leq t' \in H$
f_t	setup (fixed charge) cost in period t
p_t	unit replenishment cost in period t
h_t	unit holding cost in period t
$c_t = p_t + \sum_{j=t}^{T} h_j$	marginal cost of ordering one unit of an item in period t and holding it up to period T
W_t	maximum feasible storage usage in period t
l_t	the greatest period whose demand can be fully met from an order placed in period t

Variables	
x_t	Order or replenishment quantity in period t
y_t	binary variable such that $y_t = 0$ if $x_t = 0$; $y_t = 1$ otherwise
I_t	inventory level at the end of period t
I_t^k	inventory point (period) $k \in \{0,1\}$: $I_t^0 = 0$ and $I_t^1 = W_t - d_t$
Problems	
$g(i,j,k,l)$	minimum cost of the subproblem involving periods from i to $j-1$ with initial and final stock levels I_{i-1}^k and I_{j-1}^l and where at most only one replenishment point (period) is allowed
$G(i,k)$	minimum cost of the subproblem starting in period i with incoming inventory I_{i-1}^k and final inventory $I_T^0 = 0$
Data Structures	
B	sorted list of breaking periods ($t \in H : W_t = S_t$) which induce a partition of the planning horizon into subpartitions (subproblems)
$Pr[t]$	array containing the number of the subproblem to which each period t belongs
F_n	set of line segments related to the subproblem starting in (breaking) period $t_n + 1$
$L_{F_n}(x)$	lower envelope of the set F_n computed by an adaptation of the algorithm in [15]
$\mathcal{L}_n$	sorted list of tuples (x,t) where the first component denotes the x-coordinate of a breakpoint β in L_{F_n} and the second term stands for the line segment in L_{F_n} containing β
$\tau[i,j,k,l]$	replenishment point (period) minimizing $g(i,j,k,l)$ (i.e., $\tau[i,j,k,l] = \text{argmin} g(i,j,k,l)$)
$\mathcal{T}$	matrix containing the replenishment points (periods) $\tau[i,j,k,l]$ for all $1 \le i < j \le T+1$ and $k,l \in \{0,1\}$
$succ[i,k]$	array containing two-tuples (j,l) where j is the inventory point (period) successor to period i and $l \in \{0,1\}$ denotes the inventory level at the end of $j-1$

Accordingly, the first formulation for the ELSPBI is as follows.

$$
\begin{array}{llll}
\text{(Formulation I)} & \min & \sum_{t=1}^{T} (f_t y_t + p_t x_t + h_t I_t) & \\
 & s.t. & & \\
 & & I_t + d_t = x_t + I_{t-1} & t \in H \\
 & & I_{t-1} + x_t \le S_t & t \in H \\
 & & x_t - y_t D_t \le 0 & t \in H \\
 & & I_0 = I_T = 0 & \\
 & & x_t, I_t \ge 0;\ y_t \in \{0,1\} & t \in H
\end{array}
$$

However, taking into account that $I_t = \sum_{j=1}^{t} (x_j - d_j) = \sum_{j=1}^{t} x_j - D_{1,t}$ for $t \in H$ and denoted by $c_t = p_t + \sum_{j=t}^{T} h_j$, for the marginal cost of ordering one unit of an item in period t and holding it up to period T, the following formulation is obtained as an extension of that in [18].

(Formulation II)	min	$\sum_{t=1}^{T}(f_t y_t + c_t x_t) - \sum_{t=1}^{T} h_t D_{1,t}$	
	$s.t.$		
		$\sum_{t=1}^{T} x_t = D_{1,T}$	
		$\sum_{j=1}^{t} x_t \geq D_{1,t}$	$t = 1, \ldots, T-1$
		$\sum_{j=1}^{t} x_t \leq W_t + D_{1,t-1}$	$t = 1, \ldots, T$
		$x_t - y_t D_t \leq 0$	$t = 1, \ldots, T$
		$x_t \geq 0;\ y_t \in \{0,1\}$	$t = 1, \ldots, T$

The parameter W_t stands for the maximum feasible storage usage in period t rather than the physical storage capacity S_t, which can be computed as $W_t = \min\{W_{t+1} + d_t, S_t\}$, $t = T-1, \ldots, 1$, with $W_T = d_T$. It is noted that $W_t - d_t \leq W_{t+1}$ for all $t \in \{1, \ldots, T-1\}$ or, in general, $W_t - D_{t,t'} \leq W_{t'+1}$ and $W_t - D_{t,t'} \leq W_{t+1} - D_{t+1,t}$ for any $t' \geq t$. Moreover, let $l_t = \max\{j \in H : W_t \geq D_{t,j} \wedge W_t < D_{t,j+1}\}$ be the greatest period whose demand can be fully met from an order placed in period t. Setting $l_{T+1} = l_T = T$, the remaining values l_t, $t = 1, \ldots, T-1$ can be determined in $\mathcal{O}(T)$ as shown below.

Determining l_t for t = 1,. . . ,T+1 in $\mathcal{O}(T)$

$l_{T+1} = l_T = T$
$j = 1$
For $t = 1, \ldots, T-1$ do
 While $W_t \geq D_{t,j}$ and $j \leq T$ do
 $j = j + 1$
  endwhile
 $l_t = j - 1$
endfor
return $l_t,\ t \in H$

To define a dynamic programming recurrence for Formulation II, a key result to characterize optimal plans for the ELSPBI introduced by Love in [11] is used. Assuming that stockouts are not permitted, he claimed that *between two adjacent inventory points* (periods in which the inventory level is either 0 or at its maximum feasible capacity that can be stored) *there is at most one production point* (period with nonzero replenishment). Accordingly, let $k \in \{0,1\}$ be a superscript that indicates the value of the inventory level at the end of the inventory point i so that $I_i^k = 0$ if $k = 0$ and $I_i^k = W_i - d_i$ if $k = 1$. Moreover, let i and j be two consecutive inventory points, and let $g(i,j,k,l)$ denote the minimum cost of a subproblem involving periods from i to $j-1$ with inventories on hand I_{i-1}^k at the end of period $i-1$ and I_{j-1}^l at the end of period $j-1$, respectively, and where, at most, only one replenishment point $t:\ i \leq t \leq l_i + 1$ and $t < j \leq l_t + 1$ is allowed. Therefore, $g(i,j,k,l)$ can be expressed as follows:

$$g(i,j,k,l) = \begin{cases} \min\limits_{\substack{i \leq t \leq l_i+1 \\ t < j \leq l_t+1}} \left(f_t \delta\left(D_{i,j-1} - I_{i-1}^k + I_{j-1}^l\right) + c_t\left(D_{i,j-1} - I_{i-1}^k + I_{j-1}^l\right)\right) & \text{if } I_{i-1}^k \geq D_{i,t-1} \\ +\infty & \text{otherwise} \end{cases} \quad (1)$$

where $\delta(x) = 0$ if $x \leq 0$ and $\delta(x) = 1$ if $x > 0$.

It is clear that a naive implementation of (1) to compute all the values $g(i,j,k,l)$ for $i \in H$, $j \in H \cup \{T+1\}$ and $k,l \in \{0,1\}$ takes $\mathcal{O}(T^3)$ time. This complexity actually represents a bottleneck for determining the minimum cost $G(i,k)$ of a subproblem starting in period i with incoming inventory I_{i-1}^k and ending in period $T+1$ with final inventory $I_T^0 = 0$. Note that $G(i,k)$ can be recursively expressed as

$$G(i,k) = \min_{\substack{i < j \leq l_{l_i+1}+1 \\ l \in \{0,1\}}} (g(i,j,k,l) + G(j,l)) \tag{2}$$

where $G(T+1,0) = 0$, $G(T+1,1) = +\infty$ and where $G(1,0)$ stands for the optimal cost for a corresponding instance of the ELSPBI.

At this point, the goal is to reduce the time complexity for computing (1). To this end, the values that $g(i,j,k,l)$ can take, depending on the relationship between the positions of the inventory points i and j, are analyzed, as well as the inventory levels at these points. Accordingly, let us denote by $B = \{t \in H : W_t - d_t < W_{t+1}\}$ the sorted list of breaking periods (i.e., the set of periods $t \in H$ for which $W_t = S_t$). The term breaking periods is used to refer to the periods in B because they break the monotonicity when determining the values of W_t, $t = 1, .., T$. It is noted that, given two consecutive breaking periods $t_n < t_{n+1}$ in B, the sequence of values $W_{t_n+1} \geq W_{t_n+2} \geq \cdots \geq W_{t_{n+1}}$ is monotonically nonincreasing. A simple procedure to generate the sorted list B and the array Pr is sketched in Procedure 1, where $Pr[t]$ gives the number of the subproblem containing period $t \in H$.

Procedure 1: Construct the sorted list B and the array Pr in $\mathcal{O}(T)$

$B = \{\varnothing\}$
$n = 1$
For $t = 1, \ldots, T-1$**do**
  $Pr[t] = n$
    If $W_t < W_{t+1} + d_t$ **then**
        Insert to B
        $n = n+1$
    endif
endfor
If $W_{T-1} < W_T + d_{T-1}$ **then** $Pr[T] = Pr[T-1] + 1$ **else** $Pr[T] = Pr[T-1]$
If $B \neq \{\varnothing\}$ **then**
  Insert 0 at the beginning of B
  Insert T at the end of B
endif
Return B and Pr

The breaking periods in B allow us to decompose the original problem into a set of subproblems. Thus, given two consecutive breaking periods $t_n, t_{n+1} \in B$, the subproblem $P(t_n+1, t_{n+1})$ consists of consecutive periods $t = t_n+1, \ldots, t_{n+1}$, satisfying $W_t - D_{t,t_{n+1}-1} = W_{t_{n+1}}$. Therefore, the size of subproblem $P(t_n+1, t_{n+1})$ is given by $T_n = t_{n+1} - (t_n+1) + 1 = t_{n+1} - t_n$. It should be noted that if $B = \varnothing$ for a given instance of the ELSPBI, then the instance is uncapacitated, and hence it can be solved by any of the classical methods running in $\mathcal{O}(T \log T)$ (see [18,26,27]). Henceforth, we focus our attention on the case where list B is not empty. Let $|B|$ denote the number of breaking periods and, consequently, $|B|-1$ denotes the number of subproblems. Accordingly, when $B \neq \{\varnothing\}$, then $t_1 = 0$ and $t_{|B|} = T$.

3. Restricted Recurrence Equations

Now, some properties to reduce from $\mathcal{O}(T^3)$ to $\mathcal{O}(T^2)$ the time complexity of the solution method to compute (1) are shown. In Properties 1 and 2 below, the case to determine $g(i,j,k,l)$ when i and j are within the same subproblem (i.e., when $t_n+1 < i < j \leq t_{n+1}$) is first addressed. Under this assumption, a result similar to the planning horizon theorem, first introduced by Wagner and Whitin [1] , is provided. Afterward, the case in which $i = t_n+1$, $n = 2, \ldots, |B|-1$ and $k = 1$ is separately analyzed in Property 3, because computing $g(t_n+1, j, 1, l)$ differs from the former technique for intermediate periods. Specifically, the lower envelope of the set of line segments $F_n = \{s_t(x) = f_t + c_t x \mid t = t_n+1, \ldots, l_{t_n}+1 \text{ and } x \in (0, W_t - (I^1_{t_n} - D_{t_n+1,t-1})]\}$, related to the subproblem starting in period t_n+1, is constructed in $\mathcal{O}(T_n \log T_n)$. Once the lower envelope $L_{F_n}(x) = \min_{t_n+1 \le t \le l_{t_n}+1} s_t(x)$ has been constructed using, for instance, an adaptation of the classical approach described in [15] , a list $\mathcal{L}_n$ of tuples of the form (x, t), where the first component stands for the x-coordinate of a breakpoint β in L_{F_n} and the second term indicates the line segment s_t such that $\beta \in L_{F_n} \cap s_t$, can be defined. Note that the list $\mathcal{L}_n$ is sorted in increasing order of the x-coordinates, and we denote by $\mathcal{L}_n[m].x$ and $\mathcal{L}_n[m].t$ the x-coordinate and the corresponding replenishment point t related to the breakpoint at position m in $\mathcal{L}_n$, respectively. Besides that, the overall time complexity to construct all the lower envelopes $L_{F_n}(x)$ for $n = 2, \ldots, |B|$-1 is

$$\sum_{n=2}^{|B|-1} \mathcal{O}(T_n \log T_n) \le \sum_{n=2}^{|B|-1} \mathcal{O}(T_n \log T) \le \mathcal{O}(\log T) \sum_{n=2}^{|B|-1} \mathcal{O}(T_n) \le \mathcal{O}(T \log T).$$

To determine $g(t_n+1, j, k, l)$, it suffices to seek for, by binary search, the consecutive values of x-coordinates in $\mathcal{L}_n$, $\mathcal{L}_n[m].x$ and $\mathcal{L}_n[m+1].x$ such that $\mathcal{L}_n[m].x \le D_{t_n+1,j-1} - I^k_{t_n} + I^l_{j-1} < \mathcal{L}_n[m+1].x$ for $j = t_n+2, \ldots, l_{t_n+1}+1$ and the corresponding line segment $s_{\mathcal{L}_n[m].t}$. It is noted that the case where $n = 1$ does not need to be considered because $i = t_1 + 1 = 1$, and hence $g(1, j, 1, l) = \infty$. Finally, the case in which the inventory points i and j are not in the same subproblem is discussed in Property 4.

Property 1. *Given two consecutive breaking periods $t_n < t_{n+1}$ in B, the replenishment point t that minimizes $g(i, j, k, l)$ is not smaller than the replenishment point t' minimizing $g(i-1, j, k, l)$ when either $t_n + 1 < i - 1 < j \le t_{n+1}$ and $k, l \in \{0, 1\}$ or if $k = 0$, $l \in \{0, 1\}$ and $t_n + 1 = i - 1$.*

Proof. Suppose that $t' > t$. Then, the following cases can arise:

1. If $k = 0$ and $d_{i-1} \ne 0$, then $t' = i - 1$, and hence $t' < t$;
2. If $k = 0$ and $d_{i-1} = 0$, then $g(i-1, j, k, l) = \min\Big(f_{i-1} + c_{i-1}\Big(D_{i,j-1} + I^l_{j-1}\Big), g(i, j, k, l)\Big)$ because $D_{i-1,t-1} = D_{i,t-1}$ and $I^k_{i-2} = I^k_{i-1} = 0$. Thus, either $t' = i - 1$ or $t' = t$, and hence $t' \le t$;
3. If $k = 1$, then $I^k_{i-2} = I^k_{i-1} + d_{i-1} = W_{i-1}$ and $I^k_{i-1} = W_{i-1} - d_{i-1} = W_i$, and hence

$$g(i-1, j, k, l) = \min\Big(f_{i-1} + c_{i-1}\Big(D_{i-1,j-1} - I^k_{i-2} + I^l_{j-1}\Big), g(i, j, k, l)\Big) \tag{3}$$

Clearly, the quantity to be replenished in period $i - 1$ must be greater than 0 (i.e., $D_{i-1,j-1} - I^k_{i-2} + I^l_{j-1} > 0$) so that period $i - 1$ is a replenishment period. However, $D_{i-1,j-1} + I^l_{j-1} \le I^k_{i-2} = W_{i-1}$ to ensure the feasibility of the decision, which implies that period $i - 1$ is not actually a replenishment period and $g(i-1, j, k, l) = g(i, j, k, l)$, and hence $t' = t$. It is noted that the same argument can be applied to the rest of the periods in the subproblem (i.e., for $\hat{t} = i - 1, \ldots, j$, $W_{i-1} - D_{i-1,\hat{t}-1} = W_{\hat{t}}$), and hence no replenishment is possible if $k = l = 1$. Therefore, $g(i, j, k, l) = 0$ and $t' = t = T + 1$ (*dummy period*). Conversely, if $k = 1$ and $l = 0$, then $g(i, j, 1, 0) = \infty$ since if $k = 1$, then necessarily, $l = 1$ as well.

Moreover, as Equation (3) holds for all $t_n + 1 < i < j \le t_{n+1}$ and $k, l \in \{0, 1\}$, $g(i, j, k, l)$ can be determined applying the following result. □

Property 2. *Given $l \in \{0, 1\}$ and two consecutive breaking periods $t_n < t_{n+1}$ in B, the cost $g(i, j, k, l)$ for either $k \in \{0, 1\}$ and $t_n + 1 < i < j \le t_{n+1}$ or $k = 0$ and $t_n + 1 = i$ can be recursively determined as follows:*

$$g(i,j,k,l) = \begin{cases} +\infty & if\ k > l, \\ 0 & if\ k = l = 1, \\ f_i + c_i\left(D_{i,j-1} + I^l_{j-1}\right) & if\ k = 0\ and\ d_i \neq 0, \\ \min\left(f_i\delta\left(D_{i,j-1} + I^l_{j-1}\right) + c_i\left(D_{i,j-1} + I^l_{j-1}\right), g(i+1,j,k,l)\right) & if\ k = 0\ and\ d_i = 0. \end{cases} \tag{4}$$

Proof. By virtue of the characterization described in [11], it is well known that there must be at most one replenishment point between two inventory points i and j. Moreover, as k and l are given for i and j, the inventories on hand are also fixed for all the intermediate periods $i \leq t < j$, and hence the following cases can be distinguished:

1. If $k = 1$, $l = 0$ and $i > t_n + 1$, then the inventory on hand at the end of period $i-1$ is enough to satisfy the demands for periods i through $j-1$ and to hold $I^k_{i-1} - D_{i,j-1} \neq 0$ units at the end of period $j-1$. Therefore, this case is infeasible for all $t_n < i < j \leq t_{n+1}$ since it is assumed that $l = 0$ and hence, $I^l_{j-1} = 0$. Consequently, $g(i,j,k,l) = +\infty$;
2. If $k = l = 1$ and $i > t_n + 1$, then the inventory on hand I^k_{i-1} at the end of period $i-1$ is enough to satisfy the demands for periods i through $j-1$ and to hold I^l_{j-1} units at the end of period $j-1$. Therefore, there cannot be a replenishment point between i and j, and $g(i,j,k,l) = 0$ for all $t_n + 1 < i < j \leq t_{n+1}$;
3. If $k = 0$ and $l = 0$, then there must be at most one replenishment point between i and j whenever $D_{i,j-1} \neq 0$. Clearly, the replenishment point must be either the period i or the period that minimizes $g(i+1,j,k,l)$, according to (3) in Property 1;
4. If $k = 0$ and $l = 1$, then there must be at most one replenishment point between i and j, even when $D_{i,j-1} = 0$. Again, the replenishment point should be either the period i or the period that minimizes $g(i+1,j,k,l)$, according to (3) in Property 1.

Therefore, the values $g(i,j,k,l)$ for $t_n < i < j \leq t_{n+1}$ can be computed in constant time, which leads to a time complexity of $\mathcal{O}\left(T_n^2\right)$. Given that $T = \sum_{n=1}^{|B|-1} T_n$, then the overall time complexity to compute all the values $g(i,j,k,l)$, $t_n < i < j \leq t_{n+1}$ and $n = 1, \ldots, |B|-1$ is $\mathcal{O}\left(T^2\right)$.

The next property addresses the case in which the inventory point i is a period successor to a breaking period (i.e., $i = t_n + 1$) for $n > 1$, and hence, either $I^k_{i-1} = 0$ or $I^k_{i-1} = S_{t_n} - d_{t_n}$ if $t_n < T$. Under this assumption, Property 1 and Property 2 are no longer applicable, because it cannot be ensured that $g(i,j,k,l) = \min\left(f_i + c_i\left(D_{i,j-1} + I^l_{j-1} - I^k_{i-1}\right), g(i+1,j,k,l)\right)$ when $k = 1$ $\left(I^k_{i-1} = I^1_{t_n} = S_{t_n} - d_{t_n}\right)$. This situation is related to the notions of an elementary nonsingular plan in [14] or a block in [17] with a positive initial inventory level, or a subproblem starting with a warehouse point as in [23], which were solved in $\mathcal{O}\left(T^2\right)$.

Note that the inventory on hand I^k_{i-1} at the end of period $i-1$ is not enough to fully meet the demands for periods i through $l_{t_n} + 1$. Thus, an order should be placed in that period $t = \mathrm{argmin}\, g(i,j,k,l)$; in other words, $f_t + c_t\left(D_{i,j-1} + I^l_{j-1} - I^k_{i-1}\right) = \min_{i \leq t' \leq l_{t_n}+1 < j}\left(f_{t'} + c_{t'}\left(D_{i,j-1} + I^l_{j-1} - I^k_{i-1}\right)\right)$. Instead of using an $\mathcal{O}(T_n)$ sequential search to determine such a period, period t can be obtained in $\mathcal{O}(\log T_n)$ by a binary search if the lower envelope $L_{F_n}(x)$ of the set of line segments F_n on the interval $\left(0, D_{t_n+1,t_{n+1}} + I^1_{t_{n+1}}\right]$ was previously computed in $\mathcal{O}(T_n \log T_n)$ time. Clearly, given a period $j : i < j \leq t_{n+1}$ and $l \in \{0,1\}$, if $D_{t_n+1,j-1} - I^k_{t_n} + I^l_{j-1} \notin \left(0, D_{t_n+1,t_{n+1}} + I^1_{t_{n+1}}\right]$, then $g(i,j,k,l) = \infty$. Conversely, either $t = \mathcal{L}_n[1].t$ if there is only one entry in $\mathcal{L}_n$, or there must be two adjacent entries, say m and $m+1$, in list $\mathcal{L}_n$ such that $\mathcal{L}_n[m].x \leq D_{t_n+1,j-1} - I^1_{t_n} + I^l_{j-1} < \mathcal{L}_n[m+1].x$, and hence $t = \mathcal{L}_n[m].t$. $\square$

Property 3. *Given $k = 1, l \in \{0,1\}$ and $n > 1$, the cost $g(t_n + 1, j, 1, l)$ for $t_n + 1 < j \leq l_{t_n+1} + 1$ can be determined as follows:*

$$g(t_n+1,j,1,l) = \begin{cases} 0 & if\ D_{t_n+1,j-1} + I^l_{j-1} = I^1_{t_n}, \\ f_t + c_t\left(D_{t_n+1,j-1} - I^1_{t_n} + I^l_{j-1}\right) & if\ t < j\ and\ D_{t_n+1,j-1} - I^1_{t_n} + I^l_{j-1} \in \left(0, W_t - \left(I^1_{t_n} - D_{t_n+1,t-1}\right)\right] \\ +\infty & otherwise \end{cases}, \tag{5}$$

where the replenishment point $t = \underset{t_n+1 \leq t' \leq l_{t_n}+1}{argmin} \left(f_{t'} + c_{t'}\left(D_{t_n+1,j-1} - I^1_{t_n} + I^l_{j-1}\right)\right)$ is initially determined in $\mathcal{O}(\log T_n)$ by a binary search of the first component of the tuples in list $\mathcal{L}_n$. If $t \geq j$, then a sequential search is implemented to finally obtain in $\mathcal{O}(T_n)$ the actual period $t < j$ with the minimum cost.

Proof. Clearly, if $D_{t_n+1,j-1} + I^l_{j-1} = I^k_{t_n}$, there is no need to place an order, and $g(t_n + 1, j, k, l)$ = 0. Let us assume now that $D_{t_n+1,j-1} - I^k_{t_n} + I^l_{j-1} \in \left(0, W_t - \left(I^1_{t_n} - D_{t_n+1,t-1}\right)\right]$. If $\mathcal{L}_n$ has only one entry, then $t = \mathcal{L}_n[1].t$. Otherwise, there must be two consecutive entries, say m and $m+1$, in $\mathcal{L}_n$ such that $\mathcal{L}_n[m].x \leq D_{t_n+1,j-1} - I^1_{t_n} + I^l_{j-1} < \mathcal{L}_n[m+1].x$. Thus, the replenishment point $t = \mathcal{L}_n[m].t$ is optimal for $g(t_n + 1, j, 1, l)$ because it is feasible (i.e., $D_{t_n+1,t-1} \leq I^1_{t_n}$), and the cost returned by (5) is minimal because the line segment s_t is either partially or completely contained in L_{F_n}, or s_t is the feasible segment $(t < j)$ above the lower envelope with a minimum cost. On the contrary, when $D_{t_n+1,j-1} + I^l_{j-1} < I^k_{t_n}$ for some period, $j \leq t_{n+1}$ is due to $k = 1$ and, thereby, the inventory on hand $I^1_{t_n} = S_{t_n} - d_{t_n}$ is above the requirements to satisfy $D_{t_n+1,j-1} + I^l_{j-1}$. Consequently, an infeasibility occurs, since the actual inventory on hand at the end of period $j-1$ is greater than I^l_{j-1}, which leads to $g(t_n + 1, j, k, l) = +\infty$. □

The case in which periods i and j are not in the same subproblem $P(t_n + 1, t_{n+1})$ (i.e., $t_n < i \leq t_{n+1} < j$) is addressed below.

Property 4. *Given $k, l \in \{0,1\}$, the cost $g(i, j, k, l)$ for $t_n < i \leq t_{n+1} < j$ can be recursively determined as follows:*

$$g(i,j,k,l) = \begin{cases} f_i + c_i\left(D_{i,j-1} + I^l_{j-1}\right) & if\ k=0,\ D_{i,j-1} + I^l_{j-1} \leq W_i,\ d_i \neq 0, \\ \min\left(f_i\delta\left(D_{i,j-1} + I^l_{j-1}\right) + c_i\left(D_{i,j-1} + I^l_{j-1}\right), g(i+1,j,k,l)\right) & if\ k=0,\ D_{i,j-1} + I^l_{j-1} \leq W_i,\ d_i = 0, \\ g(i+1,j,k,l) & if\ k=0,\ D_{i,j-1} + I^l_{j-1} > W_i,\ d_i = 0, \\ g(t_{n+1}+1,j,k,l) & if\ k=1\ and\ i-1 > t_n, \\ +\infty & otherwise. \end{cases} \tag{6}$$

Proof. Let us analyze the following cases:

1. When $k = 0$ and $D_{i,j-1} + I^l_{j-1} \leq W_i$, then it is feasible to place a single order to satisfy the required quantity $D_{i,j-1} + I^l_{j-1}$ as in Property 2, depending on whether $d_i \neq 0$ or $d_i = 0$. When $D_{i,j-1} + I^l_{j-1} > W_i$, no replenishment can be placed in period i. However, if $d_i = 0$, then $g(i, j, k, l) = g(i + 1, j, k, l)$;
2. When $k = 1$, two situations may occur. On the one hand, when $i - 1 > t_n$, the inventory on hand I^k_{i-1} at the end of period $i - 1$ is enough to satisfy the demands for periods i through t_{n+1} and to hold $I^k_{t_{n+1}} = W_{t_{n+1}} - d_{t_{n+1}}$ units at the end of period t_{n+1}. Therefore, no replenishment can be placed between i and t_{n+1}, since the inventory capacity in that period is at its maximum. The only admissible decision is to order $D_{t_{n+1}+1,j-1} + I^l_{j-1} - I^k_{t_{n+1}}$ in that period $t_{n+1} + 1 \leq t \leq l_{t_{n+1}} + 1$ that minimizes the

corresponding cost, and hence $g(i,j,k,l)$ reduces to calculate $g(t_{n+1}+1,j,k,l)$ using Property 3. On the other hand, if $i-1=t_n$, two replenishments must be placed to meet the quantities $\left(D_{t_n+1,t_{n+1}}+I^{k'}_{t_{n+1}}-I^1_{t_n}\right)$ and $\left(D_{t_{n+1}+1,j-1}-I^{k'}_{t_{n+1}}+I^l_{j-1}\right)$ with $k'\in\{0,1\}$, and hence $g(i,j,k,l)=+\infty$. The exception in this case is when $j=t_{n+1}+1$. If this occurs, k' and l must be necessarily equal, and the cost $g(t_n+1,t_{n+1}+1,k,l)$ is computed according to Equation (5) in Property 3;

3. For the remaining cases, either two replenishments must be placed to satisfy the quantities $\left(D_{i,t_{n+1}}+I^k_{t_{n+1}}\right)$ and $\left(D_{t_{n+1}+1,j-1}-I^k_{t_{n+1}}+I^l_{j-1}\right)$, or the inventory on hand $I^l_{j-1}=W_{t_{n+1}}-D_{t_{n+1},j-1}$ at the end of period $j-1$ does not correspond to the actual value for that variable (i.e., $I^l_{j-1}=W_{j-1}-d_{j-1}$) when $l=1$. Both cases lead to infeasibilities, and hence $g(i,j,k,l)=+\infty$.

An additional array to store the replenishment point related to $g(i,j,k,l)$ is necessary. Thus, let $\mathcal{T}=\{\tau[i,j,k,l]\}$ be a $T\times(T+1)\times 2\times 2$ matrix in which each element $\tau[i,j,k,l]$ represents the replenishment point that minimizes $g(i,j,k,l)$ (i.e., $\tau[i,j,k,l]=\arg\min g(i,j,k,l)$). It is clear that when $g(i,j,k,l)=+\infty$, then $\tau[i,j,k,l]=T+1$ to indicate that a replenishment cannot be placed between the inventory points i and j when the inventories on hand are I^k_{i-1} and I^l_{j-1}, respectively. Moreover, if $k=0$ and $\left(D_{i,j-1}+I^l_{j-1}\right)=0$, then $g(i,j,k,l)=0$ in Property 2 and Property 4, and therefore the procedure sets $\tau[i,j,k,l]=T+1$ to also specify that an order cannot be placed between periods i and j. □

The method to determine the values $g(i,j,k,l)$ and $\tau[i,j,k,l]$ for all combinations $(i,j,k,l)\in H\times H\cup(T+1)\times\{0,1\}\times\{0,1\}$ is sketched in Procedure 2. Clearly, $g(i,j,k,l)=+\infty$ for those combinations with $i\geq j$, and hence $\tau[i,j,k,l]=T+1$. Moreover, setting $l_{T+1}=T$ ensures that the index j does not exceed the correct range when $l_i=T$.

Procedure 2: Determine $g(i,j,k,l)$ and $\tau[i,j,k,l]$ for $1\leq i<j\leq T+1$ and $k,l\in\{0,1\}$ in $\mathcal{O}(T^2)$

apply Procedure 1 to obtain the sorted list B and vector Pr
If $B=\{\varnothing\}$ **then** use the geometric technique developed by Wagelmans et al. in [18]
else
  initialize $g(i,j,k,l)=+\infty$ and $\tau[i,j,k,l]=T+1, 1\leq i\leq T+1, 1\leq j\leq T+1$ and $k,l\in\{0,1\}$
  initialize $g(T,T+1,k,0)=f_T\delta\left(d_T-I^k_{T-1}\right)+c_T\left(d_T-I^k_{T-1}\right)$ and $\tau[T,T+1,k,0]=T$, for $k\in\{0,1\}$
  initialize $g(T,T+1,k,1)=g(T,T+1,k,0)$ and $\tau[T,T+1,k,1]=T$, for $k\in\{0,1\}$
compute the lists $\mathcal{L}_n$, $n=2,\ldots,|B|-1$ in $\mathcal{O}(T\log T)$ using an adaptation of the algorithm in [15]
  For $i=T-1,\ldots,1$ **do**
    $n=Pr[i]$
    For $j=i+1,\ldots,l_{l_i+1}+1$ **do**
      For $k=0,1$**do**
        For $l=0,1$ **do**
          If $i=t_n+1$ **then**
            If $k=0$ **then**
              If $j\leq t_{n+1}+1$ **then** compute $g(t_n+1,j,0,l)$ using (4) in Property 2
                If $g(t_n+1,j,0,l)\neq+\infty$ **then**
$\tau[t_n+1,j,0,l]=\arg\min g(t_n+1,j,0,l)$
endif
else
compute $g(t_n+1,j,0,l)$ using (6) in Property 4
If $g(t_n+1,j,0,l)\neq+\infty$ **then**
$\tau[t_n+1,j,0,l]=\arg\min g(t_n+1,j,0,l)$
endif
endif

elseif $i > 1$ **then**
compute $g(t_n+1, j, 1, l)$ using (5) in Property 3
If $g(t_n+1, j, 1, l) \neq +\infty$ **then**
$\tau[t_n+1, j, 1, l] = \text{argmin}g(t_n+1, j, 1, l)$
endif
endif
else
If $i > t_n+1$ and $j \leq t_{n+1}$ **then**
compute $g(i, j, k, l)$ using (4) in Property 2
If $g(i, j, k, l) \neq +\infty$ **then**
$\tau[i, j, k, l] = \text{argmin}g(i, j, k, l)$
endif
else
compute $g(i, j, k, l)$ using (6) in Property 4
If $g(i, j, k, l) \neq +\infty$ **then**
$\tau[i, j, k, l] = \text{argmin}g(i, j, k, l)$
endif
endif
endif
endfor
endfor
endfor
endfor
endif
return $g(i, j, k, l)$ and $\tau[i, j, k, l]$ for $1 \leq i < j \leq T$ and $k, l \in \{0, 1\}$

4. Solution Method and Numerical Example

Once all the values $g(i, j, k, l)$ and $\tau[i, j, k, l]$ are computed, $G(1, 0)$ is solved in $\mathcal{O}(T^2)$ as shown in Algorithm 1. This algorithm assumes that $B \neq \{\varnothing\}$; otherwise, more efficient $\mathcal{O}(T \log T)$ algorithms could be used (see, for instance, [18,26,27]). A new array, *succ*, is introduced in which each element $succ[i, k]$, $1 \leq i \leq T$, and $k \in \{0, 1\}$ is a two-tuple (j, l). Precisely, the first component $succ[i, k].j$ represents the inventory point j adjacent to the inventory point i, and the second term $succ[i, k].l$ stands for the inventory level I^l_{j-1} at the end of period $j-1$. This array is used in Procedure 3 to construct the optimal plan for an instance of the ELSPBI.

Algorithm 1: Determine $G(1, 0)$ in $\mathcal{O}(T^2)$ when $B \neq \{\varnothing\}$

apply Procedure 2 to compute $g(i, j, k, l)$ and $\tau(i, j, k, l)$ for $1 \leq i < j \leq T+1$ and $k, l \in \{0, 1\}$
initialize $G(i, k) = +\infty$ and $succ[i, k] = (0, 0)$ for all $1 \leq i \leq T+1$ and $k = 0, 1$
initialize $G(T+1, 0) = 0$
For $i = T, \ldots, 1$ **do**
For $j = i+1, \ldots, l_{l_i+1}+1$ **do**
For $k = 0, 1$ **do**
For $l = 0, 1$ **do**
If $G(i, k) > g(i, j, k, l) + G(j, l)$ **then**
$G(i, k) = g(i, j, k, l) + G(j, l)$
$succ[i, k] = (j, l)$
endif
endfor
endfor
endfor
endfor
return $G(1, 0)$

As outlined in Procedure 3, an optimal schedule $(x_1, x_2, \ldots, x_T)$ and $(y_1, y_2, \ldots, y_T)$ can be obtained in $\mathcal{O}(T)$ for a given instance of the ELSPBI.

Procedure 3: Determine an optimal plan $(x_1,x_2,\ldots,x_T)$ and $(y_1,y_2,\ldots,y_T)$ in $\mathcal{O}(T)$

initialize $(x_1,x_2,\ldots,x_T)$ and $(y_1,y_2,\ldots,y_T)$ to $(0,0,\ldots,0)$
$k=0$
$i=1$
While $i<T+1$ **do**
  $j=succ[i,k].j$
  $l=succ[i,k].l$
  $x[\tau[i,j,k,l]]=D_{i,j-1}+I^l_{j-1}-I^k_{i-1}$
  $y[\tau[i,j,k,l]]=1$
  $i=j$
  $k=l$
endwhile
return $(x_1,x_2,\ldots,x_T)$ and $(y_1,y_2,\ldots,y_T)$

A simple example consisting of $T=5$ periods is solved to illustrate Algorithm 1. The parameters and preliminary values are shown in Table 1, where it is assumed that $c_t=p_t$ and hence $h_t=0$ for $t=1,\ldots,5$. Otherwise, if $h_t\neq 0$ for some period t, then the quantity $\sum_{t=1}^{T}h_tD_{1,t}$ must be subtracted from the value $G(1,0)$ to obtain the actual minimum cost.

Table 1. Input data.

	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$
d_t	2	0	8	3	10
f_t	2	4	8	6	7
c_t	3	28	5	10	9
S_t	12	4	9	15	11
W_t	6	4	9	13	10
l_t	2	2	3	5	5
$Pr[t]$	1	1	2	3	3

Since $B=\{0,2,3,5\}\neq\{\varnothing\}$, three subproblems, namely $P(1,2)$, $P(3,3)$ and $P(4,5)$, are defined, and hence the values $g(i,j,k,l)$ for $1\le i<j\le 6$ and $k,l\in\{0,1\}$ can be determined according to Procedure 2. Precisely, Table 2 shows four $(T+1)\times(T+1)$ arrays $G'_{(k,l)}=\left\{g'_{i,j}=g(i,j,k,l)\right\}$, one for each combination of $(k,l)\in\{(0,0),(0,1),(1,0),(1,1)\}$, in which the element occupying row i and column j corresponds to the cost $g(i,j,k,l)$.

Table 2. Values for $g(i,j,k,l)$, $i,j\in H\cup(T+1)$ and $k,l\in\{0,1\}$.

$$G'_{(0,0)}=\begin{pmatrix}\infty&8&8&\infty&\infty&\infty\\\infty&\infty&0&48&\infty&\infty\\\infty&\infty&\infty&48&\infty&\infty\\\infty&\infty&\infty&\infty&36&136\\\infty&\infty&\infty&\infty&\infty&97\\\infty&\infty&\infty&\infty&\infty&\infty\end{pmatrix}$$

Case a. $(k,l)=(0,0)$

$$G'_{(0,1)}=\begin{pmatrix}\infty&20&20&\infty&\infty&\infty\\\infty&\infty&116&53&\infty&\infty\\\infty&\infty&\infty&53&\infty&\infty\\\infty&\infty&\infty&\infty&136&136\\\infty&\infty&\infty&\infty&\infty&97\\\infty&\infty&\infty&\infty&\infty&\infty\end{pmatrix}$$

Case b. $(k,l)=(0,1)$

$$G'_{(1,0)}=\begin{pmatrix}\infty&\infty&\infty&\infty&\infty&\infty\\\infty&\infty&\infty&28&\infty&\infty\\\infty&\infty&\infty&28&\infty&\infty\\\infty&\infty&\infty&\infty&26&126\\\infty&\infty&\infty&\infty&\infty&0\\\infty&\infty&\infty&\infty&\infty&\infty\end{pmatrix}$$

Case c. $(k,l)=(1,0)$

$$G'_{(1,1)}=\begin{pmatrix}\infty&\infty&\infty&\infty&\infty&\infty\\\infty&\infty&\infty&33&\infty&\infty\\\infty&\infty&\infty&33&\infty&\infty\\\infty&\infty&\infty&\infty&126&126\\\infty&\infty&\infty&\infty&\infty&0\\\infty&\infty&\infty&\infty&\infty&\infty\end{pmatrix}$$

Case d. $(k,l)=(1,1)$

Additionally, the $T\times(T+1)$ arrays $\mathcal{T}_{(k,l)}=\left\{\tau_{ij}=\tau(i,j,k,l)\right\}$ corresponding to each combination (k,l) are shown in Table 3.

Table 3. Values for $\tau(i, j, k, l)$, $i \in H$, $j \in H \cup (T + 1)$ and $k, l \in \{0,1\}$.

$\mathcal{T}_{(0,0)} = \begin{pmatrix} 6 & 1 & 1 & 6 & 6 & 6 \\ 6 & 6 & 6 & 3 & 6 & 6 \\ 6 & 6 & 6 & 3 & 6 & 6 \\ 6 & 6 & 6 & 6 & 4 & 4 \\ 6 & 6 & 6 & 6 & 6 & 5 \end{pmatrix}$ Case a. $(k, l) = (0, 0)$	$\mathcal{T}_{(0,1)} = \begin{pmatrix} 6 & 1 & 1 & 6 & 6 & 6 \\ 6 & 6 & 2 & 3 & 6 & 6 \\ 6 & 6 & 6 & 3 & 6 & 6 \\ 6 & 6 & 6 & 6 & 4 & 4 \\ 6 & 6 & 6 & 6 & 6 & 5 \end{pmatrix}$ Case b. $(k, l) = (0, 1)$
$\mathcal{T}_{(1,0)} = \begin{pmatrix} 6 & 6 & 6 & 6 & 6 & 6 \\ 6 & 6 & 6 & 3 & 6 & 6 \\ 6 & 6 & 6 & 3 & 6 & 6 \\ 6 & 6 & 6 & 6 & 4 & 4 \\ 6 & 6 & 6 & 6 & 6 & 6 \end{pmatrix}$ Case c. $(k, l) = (1, 0)$	$\mathcal{T}_{(1,1)} = \begin{pmatrix} 6 & 6 & 6 & 6 & 6 & 6 \\ 6 & 6 & 6 & 3 & 6 & 6 \\ 6 & 6 & 6 & 3 & 6 & 6 \\ 6 & 6 & 6 & 6 & 4 & 4 \\ 6 & 6 & 6 & 6 & 6 & 6 \end{pmatrix}$ Case d. $(k, l) = (1, 1)$

A $(T + 1) \times 2$ array $G = \{G(i, k)\}$ is used now to store the minimum cost of the problems starting in period i, where $1 \le i \le T$, and ending in period $T + 1$, with the inventory on hand being I^k_{i-1}, $k \in \{0, 1\}$. The arrays G and $succ[i, k]$ are shown in Table 4.

Table 4. Arrays G and $succ$ for the instance in Table 1.

$G = \begin{pmatrix} 176 & \infty \\ 176 & 156 \\ 176 & 156 \\ 133 & 123 \\ 97 & 0 \\ 0 & \infty \end{pmatrix}$	$succ = \begin{pmatrix} (2,1) & (0,0) \\ (3,0) & (4,1) \\ (4,1) & (4,1) \\ (5,0) & (5,0) \\ (6,0) & (6,0) \\ (0,0) & (0,0) \end{pmatrix}$

Procedure 3 leads to the optimal plan $(x_1 = 6, x_2 = 0, x_3 = 5, x_4 = 2, x_5 = 10)$ and $(y_1 = 1, y_2 = 0, y_3 = 1, y_4 = 1, y_5 = 1)$ with a minimum cost of 176.

Observe that the optimal plan ordered six units in period 1, which did not correspond to the sum of demands or to the physical capacity of the warehouse, but to the maximum feasible storage usage (W_1). The same occurred in period 3, where the sum of the inventory on hand ($I_2 = 4$ units) plus the order quantity ($x_3 = 5$ units) equaled $W_3 = 9$. However, the quantity ordered in period 4 ($x_4 = 2$ units) was such that when it was added to the inventory on hand ($I_3 = 1$ units), it just satisfied the demand in that period ($d_4 = 3$ units). Finally, the order quantity in period 5 met the demand for that period ($d_5 = W_5 = 10$ units). It is worth noting that the maximum feasible storage usage (W_t) and not the physical storage capacity (S_t) was the parameter that was involved in computing an optimal solution.

5. The Wagner and Whitin Cost Structure

The cost structure proposed by Wagner and Whitin (W-W) in [1] assumes that the replenishment costs are constant (i.e., $p_t = p \ge 0$) and the holding costs are nonnegative (i.e., $h_t \ge 0$, $t = 1, \ldots, T$). Accordingly, the marginal costs $c_t = p_t + \sum_{t'=t}^{T} h_{t'}$ satisfy the inequality $c_t \ge c_{t+1}$ for $t = 1, \ldots, T-1$. This cost structure is also referred to as nonspeculative costs, because advancing partial orders to meet future demands does not lead to less costly solutions.

Although this case has been previously studied in [23], where the authors devised an $\mathcal{O}(T)$ algorithm inspired by the geometric technique introduced in [18], an alternative approach based on the on-line array-searching $\mathcal{O}(T)$ method in Monge arrays described in [28], is proposed. To this end, the following properties are introduced.

Property 5. *If a Wagner and Whitin cost structure is assumed, a ZIO optimal plan for the ELSPBI can be determined.*

Proof. For a contradiction, let us assume an optimal plan for the ELSPBI with Wagner and Whitin costs, where there are at least two consecutive replenishment points $t < t'$ such that $I_{t'-1} > 0$. Clearly, either a better plan or an equally optimal plan can be derived by simply ordering $I_{t'-1}$ units in period t' rather than in period t, which leads to savings in $(c_t - c_{t'})I_{t'-1}$. By applying the same argument to the rest of the adjacent non-ZIO replenishment periods, a ZIO optimal schedule is obtained. □

According to Property 5, the search of the optimal plan must be narrowed to the case where the inventories on hand for any pair of consecutive inventory points i and j are zero (i.e., $k = l = 0$, and so $I^k_{i-1} = I^l_{j-1} = 0$). Moreover, as Property 1 holds, it follows that $\tau(i, j, 0, 0) \leq \tau(i+1, j, 0, 0)$ and $succ[i, 0].j \leq succ[i+1, 0].j$. Given that $k = 0$ in the optimal solution, Property 3 is not applicable. For notational convenience, let $G(i)$ be the minimum cost of the problem starting in period i and ending in period T when the inventory on hand at the end of period $i-1$ is zero, and let $succ[i]$ be the replenishment period successor to period i. It is noted that $G(i) = G(i, 0)$ and $succ[i] = succ[i, 0].j$, where $G(i, 0)$ and $succ[i, 0].j$ were defined in the previous section. Similar to Wagelmans et al. [24], the cost $G(i)$ can be formulated using the following recurrence equation:

$$G(i) = \begin{cases} \min\limits_{i<j\leq succ[i+1]} \left(f_i + c_i D_{i,j-1} + G(j)\right) & if\ d_i > 0 \\ \min\left(\min\limits_{i+1<j\leq succ[i+1]} \left(f_i + c_i D_{i,j-1} + G(j)\right), G(i+1)\right) & if\ d_i = 0 \end{cases},$$

where $G(T+1) = 0$ and $succ[T+1] = T+1$.

To compute the current $G(T), \ldots, G(1)$ in linear time, let us consider the $T \times (T+1)$ array $A = \{a[i, j]\}$, where

$$a[i, j] = \begin{cases} f_i + c_i D_{i,j-1} + G(j) & if\ i < j \leq l_i + 1, \\ +\infty & if\ i \geq j\ or\ j > l_i + 1. \end{cases}$$

Following similar observations to those introduced by Aggarwal and Park in [26], $G(i)$ can be expressed according to the following recurrence equation for $1 \leq i \leq T$:

$$G(i) = \begin{cases} \min\limits_{i<j\leq l_i+1} a[i, j] & if\ d_i > 0, \\ \min\left(\min\limits_{i+1<j\leq l_i+1} a[i, j], G(i+1)\right) & if\ d_i = 0. \end{cases}$$

At this point, let us demonstrate that A is a Monge array. Note that A is a Monge array if and only if for $1 \leq i_1 < i_2 \leq T$ and $1 \leq j_1 < j_2 \leq T+1$, the following inequality holds:

$$a[i_1, j_1] + a[i_2, j_2] \leq a[i_1, j_2] + a[i_2, j_1] \tag{7}$$

Property 6. *A is a Monge array.*

Proof. Let $A' = \{a'[k, l]\}$ denote a 2×2 subarray of A, and for $k, l \in \{1, 2\}$, let $r_{k,l} = f_{i_k} + c_{i_k} D_{i_k, j_l - 1} + G(j_l) < +\infty$ be a real number. Accordingly, sixteen setting types of A' may be given:

$$\begin{pmatrix} r_{1,1} & r_{1,2} \\ r_{2,1} & r_{2,2} \end{pmatrix}, \begin{pmatrix} r_{1,1} & r_{1,2} \\ r_{2,1} & +\infty \end{pmatrix}, \ \ldots \ \begin{pmatrix} +\infty & +\infty \\ +\infty & r_{2,2} \end{pmatrix}, \begin{pmatrix} +\infty & +\infty \\ +\infty & +\infty \end{pmatrix}$$

Note that only those subarrays A' with real numbers in the secondary diagonal might not satisfy (7). Therefore, the next four cases could be critical and thereby need to be evaluated:

1. If $A' = \begin{pmatrix} r_{1,1} & r_{1,2} \\ r_{2,1} & r_{2,2} \end{pmatrix}$.

Given that $c_{i_2} \leq c_{i_1}$, then $c_{i_2} D_{j_1,j_2-1} \leq c_{i_1} D_{j_1,j_2-1}$, and when adding $f_{i_1} + G(j_1) + f_{i_2} + G(j_2)$ to both sides of the inequality, the following holds:

$$f_{i_1} + G(j_1) + c_{i_2} D_{j_1,j_2-1} + f_{i_2} + G(j_2) \leq f_{i_1} + G(j_1) + c_{i_1} D_{j_1,j_2-1} + f_{i_2} + G(j_2).$$

Moreover, as $D_{j_1,j_2-1} = D_{i_2,j_2-1} - D_{i_2,j_1-1}$ and also $D_{j_1,j_2-1} = D_{i_1,j_2-1} - D_{i_1,j_1-1}$, then

$$f_{i_1} + c_{i_1} D_{i_1,j_1-1} + G(j_1) + f_{i_2} + c_{i_2} D_{i_2,j_2-1} + G(j_2) \leq f_{i_1} + c_{i_1} D_{i_1,j_2-1} + G(j_2) + f_{i_2} + c_{i_2} D_{i_2,j_1-1} + G(j_1),$$

and hence $r_{1,1} + r_{2,2} \leq r_{1,2} + r_{2,1}$ so that A' holds (7);

2. When $A' = \begin{pmatrix} r_{1,1} & r_{1,2} \\ r_{2,1} & +\infty \end{pmatrix}$.

This combination of values for $a'[2,2] = +\infty$ and $a'[2,1] = r_{2,1} < +\infty$ can only occur when $j_2 > l_{i_2} + 1$. However, if $j_2 > l_{i_2} + 1$, then $j_2 > l_{i_2} + 1 \geq l_{i_1} + 1$ since $i_1 < i_2$, which leads to $a'[1,2] = +\infty$. Therefore, the resulting subarray is a Monge array;

3. If $A' = \begin{pmatrix} +\infty & r_{1,2} \\ r_{2,1} & r_{2,2} \end{pmatrix}$.

Once again, the combination of values for $a'[1,1] = +\infty$ and $a'[2,1] = r_{2,1} < +\infty$ implies that $j_1 > l_{i_1} + 1$, which leads to $j_2 > j_1 > l_{i_1} + 1$, and hence $a'[1,2] = +\infty$. Thus, the resulting subarray satisfies (7);

4. When $A' = \begin{pmatrix} +\infty & r_{1,2} \\ r_{2,1} & +\infty \end{pmatrix}$.

Following identical arguments to those in the preceding points, it holds that the resulting subarray is a Monge array.

Given that all the critical cases for A' are either Monge or reduced to Monge subarrays, then A is also a Monge array.

Analogous to [26], the Eppstein's on-line array-searching algorithm in [28] can be applied to determine an optimal plan for the ELSPBI with a W-W cost structure in $\mathcal{O}(T)$ time, since $G(i)$ can be computed in constant time from $G(i+1)$ and the ith column minimum of A, and $a[i,j]$ can be also obtained from $G(i)$ in constant time for all $1 \leq i \leq T$ and $1 \leq j \leq T+1$. □

6. Computational Experiment

In order to validate both the time complexity and the performance of Algorithm 1, a computational experiment was carried out using C++ (g++ 5.1) and CPLEX on a MacBook Pro with an Intel Core i5 at 2.3 GHz and 8 GB of RAM running OS X Yosemite.

For each period T from 10 to 100, with a step of 10, ten instances were randomly generated for each problem, with demands, setup costs and production costs uniformly distributed in the interval $[0,\ 10]$. Moreover, the holding costs were set to 0 in all cases, and $A \in \{5, 10, 20, 25, 50, 100\}$ denotes the percentage of capacity availability. Thus, if $A = 20$ for one instance, this means that the warehouse allowed an extra capacity of 20% of the accumulated demand up to period T. Therefore, the physical storage capacity in period t, S_t, was uniformly distributed in the interval $\left[d_t, d_t + \frac{A}{100} D_{t+1}\right]$ to ensure the feasibility of the instances.

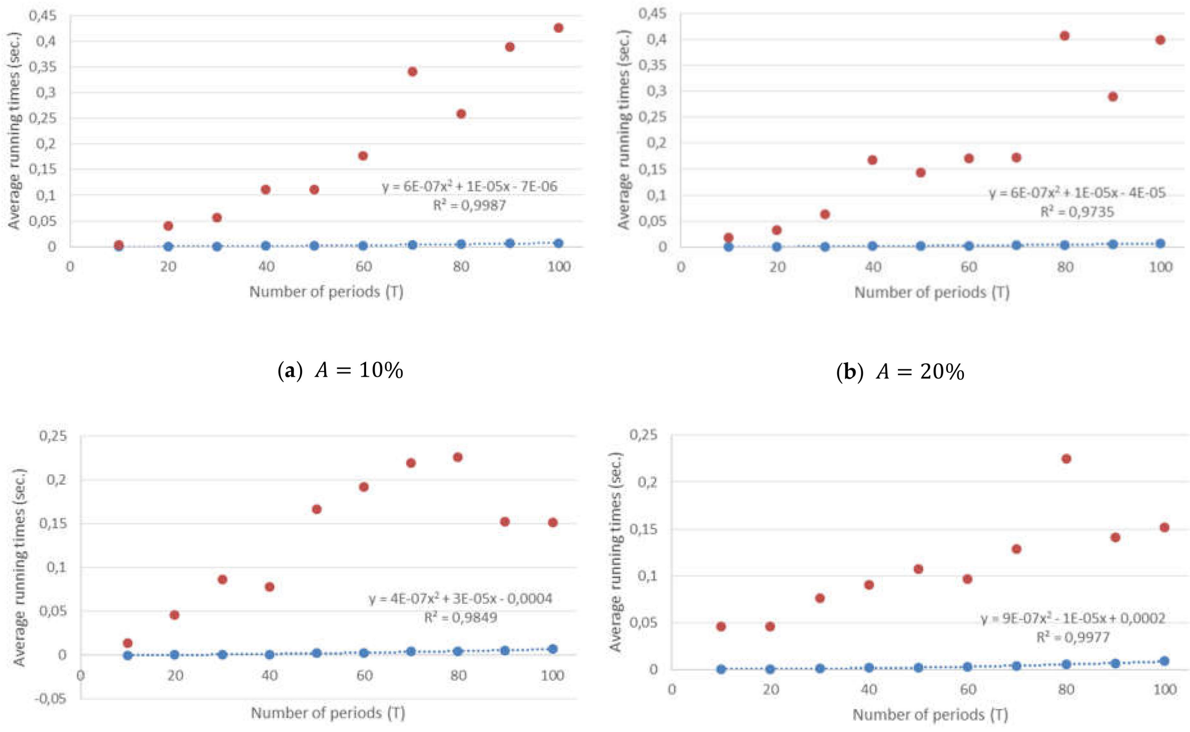
The output data reported by CPLEX and Algorithm 1 are shown in Table 5. Specifically, columns one and seven indicate the number of periods (T), whereas columns two and eight show the percentage of capacity availability (A). The third and fourth columns, as well as columns nine and ten, provide the average running times (in seconds) of the ten instances for a fixed T reported by CPLEX and Algorithm 1, respectively. Finally, the fifth and sixth columns, and similarly the eleventh and twelfth columns, list the average number of breaking periods (β) and the average usage rate of Property 3 (*Prop*. 3), respectively. Specifically, the *Prop*. 3 columns show the average rate of the number of calls to Property 3 (i.e., the usage of the geometric technique), which is defined as the quotient between the number of calls to Property 3 and the total number of calls to all the properties (i.e., $Prop.\ 3/\sum_{i=2}^{4} Prop.\ i$) in Procedure 2.

Table 5. Output data of the computational experiment.

T	*A*	CPLEX	ALG 1	β	*Prop.* 3	*T*	*A*	CPLEX	ALG 1	β	*Prop.* 3
10	5	0.00299	0.00015	11.3	0.43158	60	5	0.11656	0.00247	51.6	0.40626
	10	0.00320	0.00016	11.0	0.43085		10	0.17744	0.00278	44.5	0.34104
	20	0.01891	0.00016	10.5	0.37831		20	0.17122	0.00234	35.3	0.25744
	25	0.02099	0.00016	10.3	0.38940		25	0.27469	0.00283	33.7	0.24487
	50	0.01416	0.00016	9.1	0.27863		50	0.19270	0.00308	25.4	0.17087
	100	0.04529	0.00014	7.3	0.21151		100	0.09676	0.00274	19.5	0.11570
20	5	0.00762	0.00044	19.6	0.42551	70	5	0.21274	0.00436	58.2	0.39074
	10	0.04129	0.00047	19.4	0.41511		10	0.34147	0.00407	48.7	0.32003
	20	0.03216	0.00038	18.2	0.37849		20	0.17300	0.00436	37.9	0.23139
	25	0.05471	0.00038	17.2	0.34397		25	0.29032	0.00432	37.1	0.22554
	50	0.04583	0.00038	14.2	0.27908		50	0.21935	0.00447	26.0	0.14902
	100	0.04606	0.00035	10.5	0.17480		100	0.12861	0.00408	20.0	0.10927
30	5	0.02520	0.00064	29.3	0.44306	80	5	0.24428	0.00525	63.0	0.37283
	10	0.05722	0.00085	26.0	0.37833		10	0.25894	0.00488	53.1	0.30406
	20	0.06357	0.00076	23.0	0.32821		20	0.40658	0.00454	43.0	0.23567
	25	0.12000	0.00062	22.0	0.31140		25	0.33478	0.00461	39.5	0.21222
	50	0.08627	0.00087	16.2	0.20420		50	0.22652	0.00491	30.4	0.15197
	100	0.07567	0.00071	13.3	0.16098		100	0.22512	0.00526	22.3	0.11115
40	5	0.05712	0.00152	36.4	0.41755	90	5	0.26251	0.00673	69.0	0.36266
	10	0.11096	0.00151	33.4	0.37898		10	0.38952	0.00618	56.7	0.28845
	20	0.16717	0.00142	27.9	0.29700		20	0.28925	0.00511	44.8	0.21447
	25	0.07575	0.00136	24.6	0.27455		25	0.24529	0.00614	42.3	0.20008
	50	0.07852	0.00121	20.1	0.20402		50	0.15298	0.00535	30.5	0.13162
	100	0.09048	0.00151	13.9	0.12192		100	0.14102	0.00661	22.9	0.09540
50	5	0.09562	0.00213	42.9	0.39505	100	5	0.49571	0.00766	73.4	0.35169
	10	0.11130	0.00217	37.0	0.33252		10	0.42649	0.00751	60.1	0.26814
	20	0.14278	0.00223	31.5	0.27841		20	0.39862	0.00743	47.8	0.21000
	25	0.22647	0.00221	30.1	0.25846		25	0.28704	0.00729	44.0	0.19311
	50	0.16692	0.00227	21.7	0.17446		50	0.15190	0.00706	32.3	0.13127
	100	0.10691	0.00206	17.5	0.12979		100	0.15214	0.00865	23.0	0.08664

As expected, the running times in Table 5 reported by Algorithm 1 were practically negligible when compared with those obtained by CPLEX. Furthermore, for a fixed T, the average running times provided by Algorithm 1 were almost constant, and the average number of breaking periods in B diminished as parameter A increased. Note that the values in the column labeled β might have been greater than T due to periods 0 and T being included in B. Moreover, the column *Prop*.3 in Table 5 suggests that the average rate of the number of calls to Property 3 decreased as the percentage of capacity availability increased. Conversely, this rate increased to vary within the interval $(35\%, 45\%)$ in the most extreme conditions; that is when the percentage of storage capacity availability (A) decreased to 5%. The latter shows that the $\mathcal{O}(T \log T)$ geometric technique used in this paper played a decisive role in the resolution of the ELSPBI when the storage capacity availability was limiting, since it outperformed the $\mathcal{O}(T^2)$ approaches previously proposed in the literature for the same family of subproblems.

A comparison of the average running times reported by both CPLEX (in red) and Algorithm 1 (in blue) for different values of A is shown in Figure 1, as well as the expression of the best fit curve for the computing times of Algorithm 1 and the coefficient of determination (R square), which was above 97% in all the cases. It is worth noting that when A was below 50%, the average running times were on average 100 times faster than those reported when A was above 50% as a result of applying Property 3 (the geometric technique).

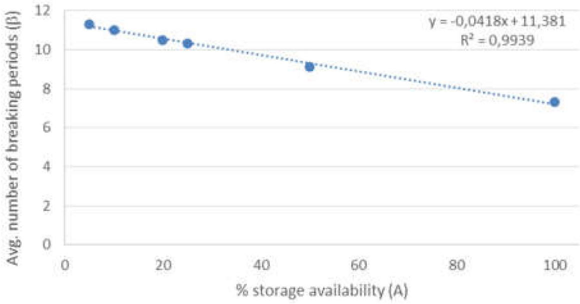


(**a**) $A = 10\%$ (**b**) $A = 20\%$

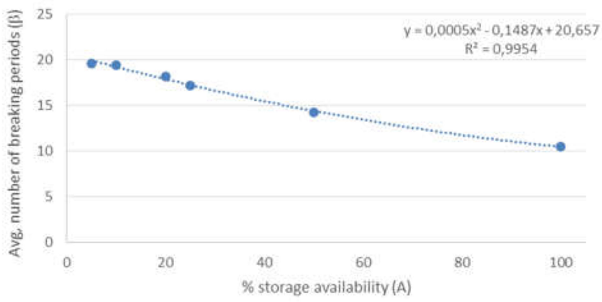
(**c**) $A = 50\%$ (**d**) $A = 100\%$

Figure 1. Comparison of average running times for different values of A reported by CPLEX (in red) and Algorithm 1 (in blue).

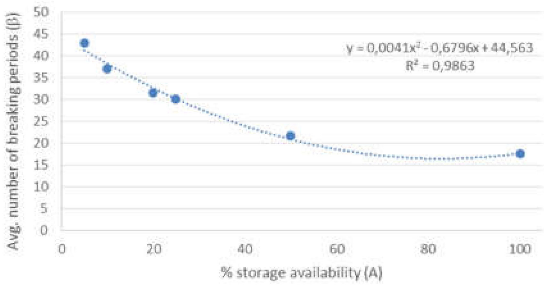
Furthermore, the average number of breaking periods in B versus the parameter A for different values of T is depicted in Figure 2. As one would expect, the average number of breaking periods (β) tended to diminish with the increasing percentage of capacity availability, as the empirical results in both Table 5 and Figure 2 reveal. According to the same results, the linear relationship between both parameters for relatively small values of T became a quadratic curve as T increased.



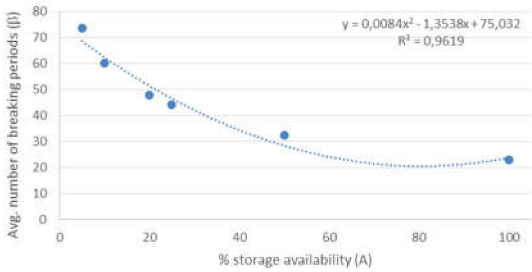
(**a**) $T = 10$



(**b**) $T = 20$



(**c**) $T = 50$



(**d**) $T = 100$

Figure 2. Average number of breaking periods (β) vs. the percentage of storage availability (A) for different values of T.

7. Concluding Remarks

Two dynamic programming algorithms for the ELSPBI were presented. For the general linear cost structure, the new $\mathcal{O}(T^2)$ approach focused on the evaluation of the feasible solution set by conveniently decomposing the original problem into subproblems. As the computational experiment revealed, the average number of such subproblems diminished as the storage capacity per period increased. Moreover, an $\mathcal{O}(T \log T)$ geometric technique was used to speed up the algorithm for those subproblems, fulfilling the same assumptions of either an elementary nonsingular plan as in [14] or a block with a positive initial inventory level as in [17], or the subpartition started with a warehouse point as in [23]. The computational results also showed that the usage of this technique was predominant under restrictive storage capacities, and hence when the percentage of capacity availability was below 50%, the average running times were on average 100 times faster than those reported when this percentage was above 50%.

On the other hand, for the case with non-speculative costs, we showed that an $\mathcal{O}(T)$ on-line array-searching method in Monge arrays could be used.

Clearly, the proposed algorithm cannot be used for a concave cost structure, and the method in [23] should be applied instead, even when an efficient algorithm for optimal piecewise linear function approximation is available, because its time complexity would likely be worse than $\mathcal{O}(T^2)$. However, the case where the stockouts are assumed to be backlogged (when the demand in a period cannot be met on time and it is met with an order placed in a later period) can be analyzed in future research to deal with issues (e.g., vacations or strikes) which can imply a disruption in the production process. Therefore, it should be demonstrated that the geometric technique used in this paper is also valid when the demands can be backlogged.

Another even more interesting topic is to determine whether an $\mathcal{O}(T \log T)$ algorithm can be devised for the ELSPBI with either a general concave or linear and fixed-charge cost structure. Although several researchers have focused their interest on this topic in the last few decades, this is currently a challenging open problem which likely will be overcome using some technique derived from the computational geometry.

Author Contributions: Investigation, J.M.G., B.A.-J. and J.S.; Methodology, J.M.G.; Validation, B.A.-J. and J.S.; Visualization, I.R.-M.; Writing—review & editing, I.R.-M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Ministerio de Ciencia e Innovación (Spain) grant number PID2019-104928RB-I00.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: We thank Marcos Colebrook for his assistance with the programming of the algorithms.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wagner, H.M.; Whitin, T.M. Dynamic version of the economic lot size model. *Manag. Sci.* **1958**, *5*, 89–96. [CrossRef]
2. Manne, A.S. Programming of economic lot sizes. *Manag. Sci.* **1958**, *4*, 115–135. [CrossRef]
3. Veinott, A.F. *Unpublished Class Notes for the Program in Operational Research*; Standford University: Standford, CA, USA, 1963.
4. Brahimi, N.; Dauzère–Pérès, S.; Najid, N.M.; Nordli, A. Single item lot sizing problems. *Eur. J. Oper. Res.* **2006**, *168*, 1–16. [CrossRef]
5. Wolsey, L.A. Progress with single-item lot-sizing. *Eur. J. Operational Res.* **1995**, *86*, 395–401. [CrossRef]
6. Jing, F.; Mu, Y. Dynamic lot-sizing model under perishability, substitution, and limited storage capacity. *Comput. Oper. Res.* **2020**, *122*, 104978. [CrossRef]
7. Phouratsamay, S.L.; Kedad-Sidhoum, S.; Pascual, F. Two-level lot-sizing with inventory bounds. *Discret. Optim.* **2018**, *30*, 1–19. [CrossRef]
8. Suzanne, E.; Absi, N.; van den Heuvel, W. A single-item lot-sizing problem with a by-product and inventory capacities. *Eur. J. Oper. Res.* **2020**, *287*, 844–855. [CrossRef]
9. Fan, J.; Wan, G. Joint optimization of dynamic lot and warehouse sizing problems. *Eur. J. Oper. Res.* **2018**, *267*, 849–854.
10. Zhong, J.; Chu, F.; Chu, C.; Yang, S. Polynomial dynamic programming algorithms for lot sizing models with bounded inventory and stockout and/or backlogging. *J. Syst. Sci. Syst. Eng.* **2016**, *25*, 370–397. [CrossRef]
11. Love, F. Bounded production and inventory models with piecewise concave costs. *Manag. Sci.* **1973**, *20*, 313–318.
12. Gutiérrez, J.M.; Colebrook, M.; Abdul-Jalbar, B.; Sicilia, J. A new characterization for the dynamic lot size problem with bounded inventory. *Comput. Oper. Res.* **2003**, *30*, 383–395. [CrossRef]
13. Gutiérrez, J.M.; Sedeño-Noda, A.; Colebrook, M.; Sicilia, J. A polynomial algorithm for the production/ordering planning problem with limited storage. *Comput. Oper. Res.* **2007**, *34*, 934–937. [CrossRef]
14. Toczylowski, E. An $\mathcal{O}(T^2)$ algorithm for the lot-sizing problem with limited inventory levels. In Proceedings of the INRIA/IEEE Symposium on Emerging Technologies and Factory Automation, Paris, France, 10–13 October 1995; Volume 3, pp. 77–85.
15. Hershberger, J. Finding the upper envelope of n line segments in $\mathcal{O}(n \log n)$ time. *Inf. Process. Lett.* **1989**, *33*, 169–174. [CrossRef]
16. Sedeño-Noda, A.; Gutiérrez, J.M.; Abdul-Jalbar, B.; Sicilia, J. An $\mathcal{O}(T \log T)$ algorithm for the dynamic lot size problem with limited storage and linear costs. *Comput. Optim. Appl.* **2004**, *28*, 311–323. [CrossRef]
17. Atamtürk, A.; Kücükyavuz, S. An $\mathcal{O}(n^2)$ algorithm for lot sizing inventory bounds and fixed costs. *Oper. Res. Lett.* **2008**, *36*, 297–299. [CrossRef]
18. Wagelmans, A.P.M.; van Hoesel, C.P.M.; Kolen, A. Economic lot sizing: An $\mathcal{O}(T \log T)$ algorithm that runs in linear time in the Wagner-Whitin case. *Oper. Res.* **1992**, *40*, S145–S156. [CrossRef]
19. Liu, T. Economic lot sizing with inventory bounds. *Eur. J. Oper. Res.* **2008**, *185*, 204–215. [CrossRef]
20. Gutiérrez, J.M.; Sedeño-Noda, A.; Colebrook, M.; Sicilia, J. An efficient approach for solving the lot-sizing problem with time-varying storage capacities. *Eur. J. Oper. Res.* **2008**, *189*, 682–693. [CrossRef]
21. Önal, M.; van den Heuvel, W.; Liu, T. A note on "The economic lot sizing problem with inventory bounds". *Eur. J. Oper. Res.* **2012**, *223*, 290–294. [CrossRef]
22. Van den Heuvel, W.; Gutiérrez, J.M.; Hwang, H. Note on "An efficient approach for solving the lot-sizing problem with time-varying storage capacities". *Eur. J. Oper. Res.* **2011**, *213*, 455–457. [CrossRef]
23. Hwang, H.C.; van den Heuvel, W. Improved algorithms for a lot-sizing problem with inventory bounds and backlogging. *Nav. Res. Logist.* **2012**, *59*, 244–253. [CrossRef]

24. Wolsey, L.A. Erratum: A tight formulation for uncapacitated lot-sizing with stock upper bounds. *Math. Program. A* **2017**, *161*, 603–609. [CrossRef]
25. Akbalik, A.; Penz, B.; Rapine, C. Capacitated lot sizing problems with inventory bounds. *Ann. Oper. Res.* **2015**, *229*, 1–18. [CrossRef]
26. Aggarwal, A.; Park, J.K. Improved algorithms for economic lot-size problems. *Oper. Res.* **1993**, *41*, 49–571.
27. Federgruen, A.; Tzur, M. A simple forward algorithm to solve general dynamic lot-sizing models with n periods in $\mathcal{O}(n \log n)$ or $\mathcal{O}(n)$ time. *Manag. Sci.* **1991**, *37*, 909–925. [CrossRef]
28. Eppstein, D. Sequence comparison with mixed convex and concave costs. *J. Algorithms* **1990**, *11*, 85–101.