SOLVING A CONSTRAINED ECONOMIC LOT SIZE PROBLEM
BY RANKING EFFICIENT PRODUCTION POLICIES

ANTONIO SEDENO-NODA
Departamento de Matemáticas, Estadística e Investigación Operativa
Universidad de La Laguna, CP 38271 - La Laguna, Tenerife (Spain)

JOSÉ M. GUTIÉRREZ
Departamento de Matemáticas, Estadística e Investigación Operativa
Universidad de La Laguna, CP 38271 - La Laguna, Tenerife (Spain)

(Communicated by Alexander Kononov)

ABSTRACT. We address the problem of finding the K best Zero-Inventory-Ordering (ZIO) policies of an Economic Lot-Sizing Problem (ELSP) with n time periods. To this end, we initially focus on devising an efficient algorithm to determine the Second Best ZIO policy. Based on both this latter algorithm and recent results from the state-of-the-art literature, we propose a solution method to compute the remaining K Best ZIO policies in O(n(K + n)) time and O(K + n^2) space. One claimed advantage of this approach is that it would efficiently solve a family of ELS problems, which includes a basic knapsack type constraint. A computational experiment is carried out to test the performance of the new algorithm K-ZIO and Constrained-ZIO for different values of K and n. For the particular case when the left-hand side coefficients in that constraint are equal, we provide an O(n^3) solution method.

1. Introduction. The Economic Lot Sizing (ELS) problem consists of satisfying at minimum cost the known non-negative demands for a specific commodity in n consecutive periods of a planning horizon. Accordingly, the demand in each period is met from either the replenishment in that period or units of the commodity stored in the warehouse in previous periods. Therefore, we assume that stock-outs are not allowed. This problem was first addressed by Wagner and Whitin [14], who proposed an O(n^2) dynamic programming algorithm based on the Zero-Inventory-Ordering property, or ZIO for short. This property states that in an optimal solution for the ELSP an order is placed only when the inventory is depleted. Later, Aggarwal and Park [1], Federgruen and Tzur [3] and Wagelmans et al. [15] independently developed more efficient methods that run in O(n log n) or O(n).

In many optimization problems, computing alternative solutions for a given optimal solution might be very useful for a decision maker. Knowledge of these alternative solutions could contribute to efficiently solve problems in which different criteria (e.g. Romeijn et al. [9]) are considered or to identify whether near-optimal solutions...

2020 Mathematics Subject Classification. Primary: 90C27, 90B05; Secondary: 65K05, 90C35.

Key words and phrases. The economic lot sizing problem, zero-inventory-ordering policies, K best solutions, production planning, graph algorithms.

The first author is supported by the grant MTM2016-74877-P from the Ministerio de Economía y Competitividad.

* Corresponding author: Antonio Sedeño-Noda.
can lead to a better overall solution, by allowing other qualitative attributes to be considered (e.g., see Guazzelli and Cunha [4]).

Another situation in which having a set of ranked ZIO plans is helpful arises when the ELSP formulation includes a knapsack type constraint. In this case, the knapsack constraint could be used to handle budgetary or environmental constraints (e.g., Helmrich et al. [5]). We prove that this variant of the ELSP admits an optimal solution satisfying the ZIO property.

To address the above situations, we propose an algorithm based on a reformulation of the ELS problem as a shortest path problem in an acyclic graph to find the $K$ best production plans for the ELSP. This set of solutions can be used by the decision maker either to identify near-optimal solutions with additional qualitative attributes or to optimally solve variants of the ELSP that include a knapsack type constraint. To this end, we consider first the problem to obtain the Second Best ZIO policy, for which we provide an $O(n \log n)$ time and $O(n)$ space algorithm. The solution of this problem is then used to determine the $K$ best ZIO solutions of the classical mathematical formulation of the ELSP in $O(n(K+n))$ time and $O(K+n^2)$ space. Therefore, the proposed algorithms become useful tools to handle situations described above.

The structure of the paper is as follows. We describe in Section 2 the mathematical formulations for the ELSP and the classical solution methods based on dynamic programming. Further, we formulate the ELSP as a Single-Source Shortest Path (SSP) problem in an acyclic graph, and we relate the solutions of both approaches, namely the dynamic programming and the SSP. The underlying network is used in Section 3 to devise an efficient algorithm that determines the Second Best ZIO policy. In Section 4, we use again the underlying network and recent results in the literature to develop the algorithm $K$-ZIO that enumerates the $K$ Best ZIO policies. The computational results show that the algorithm $K$-ZIO is surprisingly fast even when $K = 10^7$ policies, since one might need to enumerate efficiently many ZIO policies to solve ELSP problems with additional constraints. We introduce in Section 5 a variant of the ELSP including a knapsack type constraint, for which combining the $K$-ZIO algorithm with pruning strategies is helpful, as evidenced by the computational experiment in this section. Additionally, we propose a solution method for the particular case when the coefficients of the variables in the knapsack constraint are equal. Finally, we conclude with some remarks in Section 6.

2. Mathematical formulations for the ELSP. In order to introduce a classical mathematical formulation of the ELS problem, let us consider the following parameters. Let $n$ be the length of the planning horizon and let $d_i, p_i, f_i, h_i$ be, respectively, the demand, the unit production cost, the setup cost and the unit holding cost in period $i, i = 1, \ldots, n$. We assume without loss of generality that $f_i \geq 0$ (see [15],[13]) and $d_i > 0$, for all period $i$. Additionally, we denote by $d_{ij} = \sum_{k=i}^{j} d_k$, $1 \leq i \leq j \leq n$, the cumulative demand from period $i$ to period $j$. We also define the following variables: $x_i$ stands for the number of units produced/ordered in period $i$; $I_i$ represents the inventory level at the end of the period $i$ and $y_i$ is a 0-1 variable that takes the value 1 when a setup occurs in period $i$ and 0 otherwise. Further, we assume that the initial and final inventory levels of the planning horizon are zero, i.e. $I_0 = I_n = 0$, and hence a mathematical formulation of the ELS problem is given by:
SOLVING A C-ELSP BY RANKING ZIO POLICIES

\[
\min C(x, I) = \sum_{i=1}^{n} (f_i y_i + c_i x_i + h_i I_i)
\]  

subject to:

\[x_i + I_{i-1} - I_i = d_i \quad i = 1, \ldots, n\]  
\[d_{in} y_i - x_i \geq 0 \quad i = 1, \ldots, n\]  
\[I_0 = I_n = 0\]  
\[x_i, I_i \geq 0, \quad y_i \in \{0, 1\} \quad i = 1, \ldots, n\]

It is well-known that at least one optimal solution of the ELS problem satisfies the ZIO property (see [14], [16] and [17]). Recall that the ZIO property ensures that \(x_i + I_{i-1} - I_i = d_i\) for all \(i = 1, \ldots, n\). In other words, the production in period \(i\) equals 0 or \(d_{ik}\) for some period \(k \geq i\). From this property, the ELSP is solved by several dynamic programming approaches. Alternatively, we can remove the inventory level variables \(I_i\) from the formulation using \(I_i = \sum_{k=1}^{i} (x_k - d_k)\), \(i = 1, \ldots, n\), which leads to the following model:

\[
\min C(x) = \sum_{i=1}^{n} (f_i y_i + c_i x_i) - \sum_{i=1}^{n} (h_i d_{in})
\]  

subject to:

\[\sum_{k=1}^{n} x_k = d_{in}\]  
\[\sum_{k=1}^{i} x_k \geq d_{1i} \quad i = 1, \ldots, n - 1\]  
\[d_{in} y_i - x_i \geq 0 \quad i = 1, \ldots, n\]  
\[x_i \geq 0, \quad y_i \in \{0, 1\} \quad i = 1, \ldots, n\]

Where \(c_i = p_i + \sum_{k=1}^{n} h_k\), \(i = 1, \ldots, n\), are hereafter referred to as the marginal production costs, and they represent the cost of ordering one unit of commodity in period \(i\) and holding it up to period \(n\). From the constraint (7), we can assume without loss of generality that \(c_i \geq 0\) for all \(i = 1, \ldots, n\), since adding the same amount to all marginal production costs changes the objective function of any feasible solution by the same amount. Additionally, there is not loss of generality in assuming nonnegative setup costs (see [13]).

2.1. Algorithms to solve the ELSP. An \(O(n \log n)\) algorithm for solving the ELSP was devised by Wagelmans et al. [15]. The method is a backward dynamic programming algorithm based on the ZIO property. The authors define \(G(i)\) to be the cost of an optimal solution to the instance of ELS with a planning horizon consisting of periods \(i\) to \(n\), for all \(i = 1, \ldots, n\), with \(G(n+1) = 0\). From the ZIO property, the following recursion holds:

\[
G(i) = \begin{cases} 
  f_i + \min_{i < k \leq n+1} \{ c_i d_{ik-1} + G(k) \} & \text{if } d_i > 0 \\
  \min \{ G(i+1), f_i + \min_{i < k \leq n+1} \{ c_i d_{ik-1} + G(k) \} \} & \text{if } d_i = 0
\end{cases}
\]
A straightforward implementation of the above recursion leads to an $O(n^2)$ time algorithm. However, Wagelmans et al. [15] introduced a geometric technique to determine the optimal production period $k$ successor to $i$ in $O(\log n)$ time, and hence the overall computational effort to solve the above recurrence equations is $O(n \log n)$ time. Similarly, Van Hoesel [12] devised an $O(n \log n)$ algorithm to solve the ELSP introducing a forward dynamic programming algorithm. In this case, $F(i)$ is defined as the cost of an optimal solution to the instance of ELS with a planning horizon consisting of periods 1 to $i$, for all $i = 1, \ldots, n$. If we set $F(0) = 0$, the recursion equation can be formulated as follows:

$$F(i) = \min_{0 < k \leq i} \{ f_k + c_k d_{ki} + F(k - 1) \}$$

Once again, a straightforward implementation of the above recursion leads to an $O(n^2)$ time algorithm. Nevertheless, Van Hoesel [12] developed an algorithm that solves the above recurrence equations in $O(n \log n)$ time. The reader is referred to [13] and [15] for further details on these algorithms. To illustrate the above recurrence equations, we show in Table 1 the input data for an instance of the ELSP used in [14] and the corresponding values $G(i), F(i - 1), i = 1, \ldots, n + 1$.

Additionally, Table 1 also includes the values for $\text{Succ}(i)$ and $\text{Pred}(i)$, which represent, respectively, the period successor and predecessor to period $i$ in the corresponding recursion. It is well-known that the ELSP can be formulated as a Single-Source Shortest Path (SSP) problem in acyclic graph $H = (V, A)$ (see [13] for further details). Where $V = 1, 2, \ldots, n + 1$ denotes the set of nodes and $A = \{(i, j)|1 \leq i < j \leq n + 1\}$ is the set of arcs, so that for each arc $(i, j)$, we define its cost as $l_{ij} = f_i + c_i d_{ij}$, if $d_{ij} > 0$ and $l_{ij} = 0$, otherwise. We assume that for a given arc $(i, j)$, the value $l_{ij}$ is computed in constant time. Clearly, the length of a shortest path from $i$ to $n + 1$ in $H$ is $G(i)$ and the length of a shortest path from 1 to $i$ is $F(i - 1)$. In this case, $G(i) + F(i - 1) = G(0) + F(1)$ if and only if node $i$ belongs to the shortest path from 1 to $n + 1$ in $H$. The shortest path tree from any node in $V - \{n + 1\}$ to node $n + 1$ is depicted in Figure 1.a. The shortest path from node 1 to any node in $V - \{1\}$ is shown in Figure 1.b. In Figure 1.a, $\text{Succ}(i)$ is the node successor to node $i$ in the shortest path from $i$ to $n + 1$, whereas

|     | $d_{in}$ | $f_i$ | $c_i$ | $G(i)$ | $\text{Succ}(i)$ | $F(i - 1)$ | $\text{Pred}(i)$ |
|-----|---------|-------|------|--------|----------------|------------|----------------|
| 1   | 630     | 85    | 12   | 4724   | 3              | 0          | 1              |
| 2   | 561     | 102   | 11   | 3858   | 4              | 913        | 1              |
| 3   | 532     | 102   | 10   | 3463   | 5              | 1261       | 1              |
| 4   | 496     | 101   | 9    | 3041   | 5              | 1693       | 1              |
| 5   | 435     | 98    | 8    | 2391   | 8              | 2333       | 3              |
| 6   | 374     | 114   | 7    | 1859   | 8              | 2892       | 4              |
| 7   | 348     | 105   | 6    | 1634   | 8              | 3126       | 4              |
| 8   | 314     | 86    | 5    | 1325   | 10             | 3399       | 5              |
| 9   | 247     | 119   | 4    | 935    | 11             | 3820       | 8              |
| 10  | 202     | 110   | 3    | 679    | 11             | 4045       | 8              |
| 11  | 135     | 98    | 2    | 368    | 13             | 4356       | 10             |
| 12  | 56      | 114   | 1    | 170    | 13             | 4593       | 10             |
| 13  | 0       | 13    |      | 4724   | 11             |            |                |

Table 1. Input data and the values of $G(i), F(i - 1), \text{Succ}(i)$ and $\text{Pred}(i)$. 
Pred\( (i) \) is the predecessor node to node \( i \) in the shortest path from 1 to \( i \) in Figure 1.b. We set \( \text{Succ}(n + 1) = n + 1 \) and \( \text{Pred}(1) = 1 \) for convenience.

Note that any path from node \( i \) to node \( n + 1 \) in Figure 1.a represents the optimal schedule for the ELSP considering exclusively the planning horizon from \( i \) to \( n + 1 \). For example, the shortest path from 6 to 13 is 6 → 8 → 10 → 11 → 13. That is, an order is placed in period 6 to satisfy the accumulated demand for periods 6 and 7. In period 8, the order quantity meets the accumulated demand for periods 8 and 9, whereas the replenishment in period 10 meets its own demand only. Finally, an order is placed in period 11 to satisfy the demands of periods 11 and 12. The overall cost is \( G(6) = 1859 \). Likewise, any path from node 1 to node \( i \) in the shortest path tree in Figure 1.b represents the optimal planning production for the ELSP considering exclusively the periods 1 to \( i - 1 \). For instance, the shortest path from 1 to 6 is 1 → 4 → 6. That is, in period 1 the order satisfies the demand for the periods 1, 2 and 3. The replenishment in period 4 meets the demands for periods 4 and 5. The overall cost is \( F(6 - 1) = 2892 \). In the next section, we assume that the values \( G(i) \), \( F(i - 1) \), for all \( i = 1, \ldots, n \) and the values of \( \text{Succ}(i) \) and \( \text{Pred}(i) \) for all \( i = 1, \ldots, n + 1 \), are available.

3. The second best ZIO policy. We introduce and prove the basic results to solve efficiently the Second Best ZIO policy problem. Accordingly, given an optimal
ZIO policy $x^*$ for the ELSP, the Second Best ZIO Policy problem is to find a ZIO policy $x$ such that $C(x^*) \leq C(x)$ and any other ZIO policy $\hat{x} \neq x$ and $\hat{x} \neq x^*$ satisfies $C(\hat{x}) \geq C(x)$. Remark that the ELSP could have alternative optimal non-ZIO solutions. Moreover, the second, third and so on best solutions could be non-ZIO solutions. However, we are interested in this paper only in the computation of ZIO solutions, since later we will search for the optimal solution of an ELSP with additional constraints, which will keep the ZIO property. We show below that there exists a relationship between the previously introduced problem and that considering shortest path problems in a modified graph $H$.

**Remark 1.** The Second Best ZIO policy can be determined as the second best shortest path from 1 to $n+1$ in $H$.

Remark 1 holds since any path from node 1 to node $n+1$ in $H$ determines a ZIO policy for the ELSP. Thus, let $Path$ be the set of nodes in the shortest path from node 1 to node $n+1$ in $H$. There are several ways to compute the second best shortest path in a directed acyclic graph (DAG) (see, for instance, [2] and [8]). One of them consists of determining the shortest paths from node 1 to node 2 in the graphs $H$ there exists a relationship between the previously introduced problem and that with additional constraint, which will keeps the ZIO property. We show below that there exists a relationship between the previously introduced problem and that considering shortest path problems in a modified graph $H$.

**Remark 1.** The Second Best ZIO policy can be determined as the second best shortest path from 1 to $n+1$ in $H$.

Accordingly, let $sPred(i)$ be the period $j \leq i$ such that $SF(i) = F(j-1) + f_j + c_j d_j$. Finally, $R(i) = SF(i-1) + G(i)$ for each $i \in Path$ with $i > 2$ and the optimal value of the Second Best ZIO policy equals $\min \{R(i) : i \in Path\}$ with $i > 2$.

Clearly, if we use $i^* = \arg\min \{R(i) : i \in Path\}$ with $i > 2$, the production schedule for the Second Best ZIO policy is obtained by concatenating the sub-path from $i^*$ to $n+1$ identified by $\text{Succ}(i)$ labels and the sub-path from node 1 to $sPred(i^*)$ using $\text{Pred}(i)$ labels.

In Table 2, we show the values for $SF(i)$ and $sPred(i)$, for $i = 2, \ldots, n$, and the values for $R(i)$ for $i \in \{3, 5, 8, 10, 11, 13\}$. The minimum $R(i)$ is attained in
Table 2. Values of \( SF(i - 1) \), \( sPred(i) \) and \( R(i) \) for the example in Table 1.

| \( i \) | \( G(i) \) | \( Succ(i) \) | \( F(i - 1) \) | \( Pred(i) \) | \( SF(i - 1) \) | \( sPred(i) \) | \( R(i) \) |
|---|---|---|---|---|---|---|---|
| 1 | 4724 | 3 | 0 | 1 |
| 2 | 3858 | 4 | 913 | 1 |
| 3 | 3463 | 5 | 1261 | 1 |
| 4 | 3041 | 5 | 1261 | 1 |
| 5 | 2391 | 8 | 2333 | 3 |
| 6 | 1859 | 8 | 2892 | 4 |
| 7 | 1634 | 8 | 3126 | 4 |
| 8 | 1325 | 10 | 3399 | 5 |
| 9 | 935 | 11 | 3820 | 8 |
| 10 | 679 | 11 | 4045 | 8 |
| 11 | 368 | 13 | 4593 | 10 |
| 12 | 170 | 13 | 4724 | 11 |
| 13 | 0 | 13 | 4724 | 11 |

period \( i = i^* = 5 \), which implies that the Second Best ZIO policy is therefore: \( 1 \rightarrow 4 \rightarrow 5 \rightarrow 8 \rightarrow 10 \rightarrow 11 \rightarrow 13 \), and its optimal cost is 4734 (in bold in Table 2).

Lemma 3.1. The problem of determining the Second Best ZIO policy can be solved in \( O(n \log n) \) time using \( O(n) \) space.

Proof. The claim holds from the fact that the values \( F(i) \) and \( SF(i) \), for all \( i = 0, \ldots, n \), are computed simultaneously using the algorithm given in [12] and the values for \( G(i) \), for all \( i = 1, \ldots, n+1 \), are obtained using the algorithm in [15]. Both methods require \( O(n \log n) \) time and \( O(n) \) space. Further, \( \min \{ R(i) : i \in \text{Path} \} \) with \( i > 2 \) can be determined in \( O(n) \) additional time as well as the identification of the Second Best ZIO policy using \( Pred() \), \( Succ() \) and \( sPred(i^*) \) labels.

In the next section, we show that the methodology above for computing the Second Best ZIO policy can be exploited to identify the \( K \) Best ZIO policies.

4. The \( K \) best ZIO policies problem. The \( K \) Best ZIO policies (\( K\)-ZIO) problem consists of determining the \( K \) best ZIO solutions of the ELSP. In other words, the goal is to identify \( K \) ZIO policies \( x^k \) with \( k \in \{1, \ldots, K\} \) such that \( C(x^1) \leq C(x^2) \leq \ldots \leq C(x^K) \) and \( C(x^p) \geq C(x^k) \), for any other ZIO policy \( x^p \neq x^k \).

We can easily design an algorithm based on the results in section 3, where in each iteration we find the optimal and the second best ZIO policy for a particular instance of the ELSP. Clearly, this approach leads to an algorithm that consumes \( O(n \log n) \) time and \( O(Kn) \) space. Note that the ELSP with \( n \) periods has \( 2^n \) possible ZIO policies, so that the maximum value for \( K \) is \( 2^n - 1 \).

However, we prefer to compute the \( K \) Best ZIO policies identifying the \( K \) shortest paths (\( K\)-SP) from node 1 to \( n+1 \) in the previously described acyclic graph \( H \) by using Remark 1. To this end, we adapt the results in Pascoal and Sedeño-Noda [8] and Sedeño-Noda [10] for the ELSP to devise an algorithm that runs in \( O(n(K + n)) \) time and uses \( O(n^2 + K) \) space. Observe that this algorithm is preferable to that commented in the above paragraph when \( K > n/\log n - 1 \). Some ELS problems with additional constraints are NP-hard and their exact solution methods require...
enumerating more than $n$ ZIO policies. Therefore, in this context, the new approach will be most useful. Now, we give a sketch of the proposed algorithm.

Let $i, j \in V$ be two distinct nodes in $H = (V, A)$. We define a directed path $P_{ij}$ as a sequence $< i_1, (i_1, i_2), i_2, \ldots, i_{l-1}, (i_{l-1}, i_l), i_l >$ of nodes and arcs satisfying $i_1 = i, i_l = j$ with $(i_v, i_{v+1}) \in A$ for all $1 \leq v \leq l - 1$. The length of a directed path $P$ is the sum of the arc lengths in the path, that is, $c(P) = \sum_{(i, j) \in P} l_{ij}$. Let $\Gamma_i^- = \{ j : 1 \leq j < i \}$ be the set of predecessor nodes of node $i$ in $H$.

The reduced costs of the arcs with respect to the values $F(i)$, for $i = 0, \ldots, n$, are computed in $O(1)$ time for any arc $(i, j) \in A$ as $\bar{l}_{ij} = l_{ij} + F(i - 1) - F(j - 1)$. Note that the optimality conditions for the shortest path problem are $\bar{l}_{ij} \geq 0$ for all $(i, j) \in A$. Therefore, we can solve the $K$-SP problem as the problem that determines the paths $P_{st}^k$ for $k \in \{1, \ldots, K\}$ satisfying $c(P_{st}^1) \leq c(P_{st}^2) \leq \ldots \leq c(P_{st}^K)$ and such that $c(P_{st}^1) \geq c(P_{st}^k)$ for any other path $P_{st} \neq P_{st}^k$. Here, $s = 1$ and $t = n + 1$. The algorithm in [8] is based on the next result:

**Lemma 4.1.** Let $P_{st}^1$ be the shortest $s-t$ path in a DAG. The second best $s-t$ path is $P_{st}^2 = P_{st}^1 \cup (i, j) \cup P_{jt}^1$, where $(i, j) = \arg \min_{u \in \Gamma_i^-} \{ \bar{l}_{uv} : v \in P_{st}^1 \text{ and } (u, v) \notin P_{st}^1 \}$.

![Figure 2.a. Illustration of the Lemma 1.](image1)

![Figure 2.b. The second best path with a fixed sub-path (black nodes).](image2)

**Figure 2. Shortest paths in graph $H$.**

Figure 2.a shows schematically Lemma 4.1 where $(i, j) = (4, 5)$ is the arc defined in the lemma (in red color). The Second Best ZIO policy for the instance in Table 1 is determined by the arc $(4, 5)$ with $l_{45} = 10$, according to Lemma 4.1. Note that
this implies that the second best path has length \( c(P_{st}^1) + \bar{l}_{45} = 4724 + 10 = 4734 \) as already calculated in the previous section.

The algorithm uses a binary partition strategy to prevent the same path from being computed more than once. In this strategy, the algorithm associates a set of non-path arcs with each best path. The initial set of non-path arcs for \( P_{st}^1 \) is \( A(P_{st}^1) = \{(u,v)|u \in \Gamma^*_v, v \in P_{st}^1 \} \). That is, the set of non-path arcs arriving at any node in \( P_{st}^1 \) (the argument of the minimum in Lemma 4.1). Once the second best path \( P_{st}^2 = P_{st}^1 \cup (i,j) \cup P_{jt}^1 \) is obtained from \( P_{st}^1 \), the set \( A(P_{st}^1) \) is updated as follows: \( A(P_{st}^1) = A(P_{st}^1) - \{(i,j)\} \). Now, the sub-path \( P_{it}^2 = (i,j) \cup P_{jt}^1 \) in \( P_{st}^2 \) is fixed (sub-path \( P_{4,13} \) in 2.b). This means that any other path obtained from \( P_{st}^2 \) must contain the sub-path \( P_{st}^2 \) (in particular these paths will contain the arc \((i,j)\)). Therefore, the set of non-path arcs of \( P_{st}^2 \) is \( A(P_{st}^2) = \{(u,v)|u \in \Gamma^*_v, v \in P_{st}^1 \} \). That is, the set of non-path arcs arriving at some node in \( P_{st}^2 \) (the nodes in the non-fixed sub-path). Clearly, the second best path of \( P_{st}^2 \) (containing the sub-path \( P_{st}^2 \)) satisfies Lemma 4.1 considering the arcs in the set \( A(P_{st}^2) \). Next, the second best paths for the paths \( P_{st}^1 \) and \( P_{st}^2 \) are iteratively computed and so on until the algorithm computes the \( K \) best paths. A detailed scheme of the algorithm is given in [8]. Thus, we have the next result.

**Theorem 4.2.** The K-ZIO algorithm computes the \( K \) Best ZIO policies in \( O(n(K+n)) \) time and \( O(n^2+K) \) space for a planning horizon with \( n \) periods.

**Proof.** Firstly, the algorithm computes the values \( F(i) \) and \( Pred(i) \) for \( i = 0 \) to \( n \) in \( O(n \log n) \) time (see [12]). The data structure associated to the graph \( H \), where each arc \((i,j)\) has length/cost \( l_{ij} \), can be arranged in \( O(n^2) \) time. Determining \((i,j) = \arg \min_{u \in \Gamma^*_v} \left\{ l_{uv} : v \in P_u^k \text{ and } (u,v) \notin P_{st}^k \right\} \) consumes \( O(n) \) time (see Lemma 4 in [8]). Accordingly, in each iteration the computation of the Second Best ZIO policy consumes \( O(n) \) time. Clearly, the algorithm makes at most \( K \) iterations. In each iteration of the algorithm, two second best solutions are determined in \( O(n) \) overall time. Moreover, in each iteration, the algorithm carries out, respectively, one extraction and two insertion operations in a binary heap in \( O(\log k + \log k + \log(k+1)) \) time. Thus, the worst case complexity of the algorithm is \( O(n^2+Kn+K \log K) \) time, which leads to \( O(n(K+n)) \) time, since \( K < 2^n \). On the other hand, the space required by the algorithm is \( O(n^2+K) \), because there are at most \( K \) candidates in the heap and the graph \( H \) needs \( O(n^2) \) space.

Theorem 4.2 follows directly from the results in [8], which show that finding the \( K \) best shortest paths in a DAG with \( n \) nodes and \( m \) arcs can be done in \( O(m+Kn) \) time using \( O(K+m) \) space since the graph \( H \) has \( m = n(n+1)/2 \) arcs. The algorithm above outputs all ZIO polices in lexicographic min order and with \( O(n) \) delay time between two consecutive ZIO policies, once the first best ZIO policy is obtained in \( O(n \log n) \) time and the graph \( H \) is setting up in \( O(n^2) \) time. Johnson et al. [6] introduced the notion of polynomial delay and polynomial space algorithms. Following these ideas, the described algorithm is a polynomial delay algorithm, but it is not a polynomial space algorithm since it uses potentially exponential space, in the form \( O(n^2+K) \).

The algorithm explicitly does not store the paths, only the first shortest path tree is stored (Figure 2a). The remaining \( K-1 \) best paths can be rebuilt using a minimal information. To this end, the algorithm uses a father vector where \( father[k] = \{p, (u,v)\} \). Here \( p \) is the index of the best path \( P_{st}^p \) (father path) from
Table 3. Average running times (in seconds) for each pair \((n, K)\).

| \(n\)  | 10,000 | 100,000 | 1,000,000 | 10,000,000 |
|--------|--------|---------|-----------|------------|
| 25     | 0      | 0.033   | 0.419     | 5.474      |
| 50     | 0.004  | 0.053   | 0.532     | 5.621      |
| 100    | 0.006  | 0.061   | 0.714     | 7.391      |
| Mean   | 0.003  | 0.049   | 0.555     | 6.162      |

the \(k\)th path, which is determined as \(P^k_{st} = P^1_{st} \cup (u, v) \cup P^p_{vt}\), where the arc \((u, v)\) fulfills Lemma 4.1. Using this information, the fixed sub-path \(P^f_{st} = (u, v) \cup P^p_{vt}\) (the sub-path from 4 to 13 in Figure 2.b for example) can be derived from the predecessor labels \((\text{Pred})\) by the next recursive algorithm \(BP\).

Algorithm 1 \((BP)\) BuildingPath\((k, \text{father}, \text{Pred}, \text{var} u)\)

1: if \((s \neq i)\) then
2: \(BP(\text{father}[k].p, \text{father}, \text{Pred}, u)\);
3: \((i, j) = \text{father}[k].(u, v)\);
4: while \((u \neq j)\) do
5: \(\text{print } (u \leftarrow )\);
6: \(u = \text{Pred}(u)\);
7: \(\text{print } (j \leftarrow )\);
8: \(u = i;\)

Basically, the algorithm \(BP\) concatenates the fixed sub-paths of the ascendants paths of the \(k\)th path in the inverse order of the sequence identified by \(\text{father}[k].p, \text{father}[\text{father}[k].p], \ldots, 1\). Once the algorithm \(BP\) ends, the sub-path from node 1 to the modified node \(u\) still needs to be found using predecessor labels to identify the complete path \(P^k_{st}\). This scheme was introduced in [10].

4.1. Computational experiment with the \(K\)-ZIO algorithm. In order to test the practical performance of the proposed tool, the algorithm \(K\)-ZIO was programmed in C and compiled with option O3. The computational experiment were performed on a computer with an Intel Xeon 3.70GHz \times 8 with 64GB of RAM running Ubuntu 16.04. The instances considered in our experiment are similar to those in Helmrich et al. [5], and hence the number of periods \((n)\) takes the values 25, 50 and 100. The demand for each period was generated from a discrete uniform distribution with minimum 0 and maximum 200, \(\text{DU}(0,200)\). The production and inventory costs were generated from \(\text{DU}(0,20)\) and the set-up costs from \(\text{DU}(500,1500)\). The values of the number of ZIO policies \(K\) were 10,000, 100,000, 1,000,000 and 10,000,000. For each fixed value of \(n\) and \(K\), we generate ten instances using ten different seeds. The performance of the algorithm was measured considering two values for each instance: the CPU time (in seconds) and the gap between the costs of the optimal ZIO policy and the \(K\)th policy, i.e. \(c(P^K) - c(P^1)\).

In Table 3, we observe that the average running times of the best 10,000 or 100,000 ZIO policies are practically negligible. Moreover, computing one million of ZIO policies takes less than one second, whereas the \(K\)-ZIO algorithm requires at most 7 seconds to rank ten millions of ZIO policies. Observe that for \(n=25\) periods,
the number of possible ZIO policies is $2^{24}$, which implies that enumerating these ZIO policies sorted by their costs requires at most $1.6 \times 5.474 = 8.76$ seconds with the proposed algorithm.

Furthermore, we are interested in checking if the difference between the values of the first and the last ($K$th) ZIO policies has influence on the practical behavior of the algorithm. Thus, Figures 3 and 4 show the CPU times against the difference $c(P^K) - c(P^1)$ for $K$ being 1,000,000 and 10,000,000, respectively. Surprisingly, we observe that the $K$-ZIO algorithm takes more time for smaller differences of $c(P^K) - c(P^1)$. However, the behavior of the algorithm in practice is not completely explained by this measure, because it is linear in $K$.

In the next section, we exploit the power of the $K$-ZIO algorithm in a variant of the ELSP. Specifically, we include a knapsack type constraint to the formulation of the ELSP to model situations in which an extra limitation (e.g., budgetary or environmental) should be handled.

5. The ELSP with knapsack type constraint. Let us consider the ELSP with an additional constraint as that included in the basic knapsack problem. We call this new problem the Constrained-ELSP. The new formulation can be seen as a variant of the ELSP, which may be used to handle real-world situations where an extra restriction (e.g., budgetary or environmental) is imposed (see, for instance [5], for a more general model). The Constrained-ELSP can be formally expressed as follows:
\[
\min C(x) = \sum_{i=1}^{n} (f_i y_i + c_i x_i) - \sum_{i=1}^{n} (h_i d_{in}) \tag{11}
\]
subject to:
\[
\sum_{k=1}^{n} x_k = d_{1n} \tag{12}
\]
\[
\sum_{k=1}^{i} x_k \geq d_{1i} \quad i = 1, \ldots, n - 1 \tag{13}
\]
\[
d_{in} y_i - x_i \geq 0 \quad i = 1, \ldots, n \tag{14}
\]
\[
\sum_{k=1}^{n} w_k y_k \leq R \tag{15}
\]
\[
x_i \geq 0, \quad y_i \in \{0, 1\} \quad i = 1, \ldots, n \tag{16}
\]

Where \( w_k, \ k = 1, \ldots, n, \) is a non-negative constant representing the usability of a resource of economic, environmental or of any other nature, and \( R \) denotes the availability or limitation of such a resource. Observe that it can easily be shown that there exists an optimal solution for the Constrained-ELSP satisfying the ZIO property. To this end, let us consider an optimal non-ZIO solution \((x, y)\)

\[\text{Figure 4. CPU times vs. } c(P^K) - c(P^1) \text{ for } K=10,000,000.\]
of the Constrained-ELSP, and let $S = \{i : y_i = 1, i = 1, \ldots, n\}$ be its set of replenishment periods, where $\bar{n} = |S|$. Next, it can be easily derived an unconstrained instance of ELSP consisting only of the periods in $\bar{S}$ with demands $\bar{d}_{ji} = d_{ji,n+1}$, $\bar{c}_{ji} = c_{ji}$, and $\bar{f}_j = f_j$, for all $i = 1, \ldots, \bar{n}$. Clearly, $x$ is also an optimal solution of this unconstrained ELSP, since the periods where $y_i = 0$ in the constrained plan does not affect the optimal solution of this ELSP instance. Now, the same arguments as in [14] can be used to build an alternative optimal solution $(\bar{x}, \bar{y})$ satisfying both the ZIO property and implicitly the additional constraint $\sum_{k \in S} w_k \bar{y}_k \leq R$.

Hereinafter we assume only instances of that ELSP so that $\sum_{k=1}^{n} w_k y_k > R$ holds ($y_k$ denotes the optimal value for $y_k$ in the corresponding instance of the ELPS), the instance would be unconstrained, otherwise.

In Romeijn et al. [9] a family of ELS problems, denoted by $\Upsilon(x, b)$, is discussed, where $\Upsilon(x, b)$ stands for an ELS problem with $n/l$ expenditure constraints and $b$ is the maximum availability of the resource. According to this notation, the Constrained-ELSP introduced in this paper corresponds to the problem $\Upsilon(x, R)$, where only the setup expenditures are considered in the expenditure constraint. However, the authors only focus on solving the classes of $\Upsilon(x, b)$ with nonspeculative costs. Therefore, the sections below deal with the resolution of two more general cases than those addressed in [9].

5.1. An exact method to solve the general case of the constrained-ELSP. A classical way to solve the Constrained-ELSP is the Dynamic Programming (DP) scheme. The main feature of the algorithms based on DP is that they can be extremely fast for reasonably sized networks. However, they might fail to scale well for very large networks due to the curse of dimensionality of DP. The reason follows from the huge number of labels that need to be computed/stored. To show this behaviour of the DP scheme, we propose an algorithm based on this approach. Accordingly, let $F(i, r)$ be the optimal cost of the ELS problem consisting of periods 1 to $i$ with resource consumption $r$, for all $i = 1, \ldots, n$ and for all $r = 0, \ldots, R$. If we set $F(0, r) = 0$, for all $r = 0, \ldots, R$, the recursion equations can be formulated as follows

$$F(i, r) = \min_{1 \leq k \leq i} \left\{ \begin{array}{ll} F(k-1, r-w_k) + f_k + c_k d_{ki} & \text{if } r - w_k \geq 0 \\ +\infty & \text{otherwise} \end{array} \right.$$ 

The optimal value is determined by $F(n, R)$. A direct implementation of the above recurrent equation leads to an $O(n^2 R)$ time and $O(n R)$ space algorithm. Clearly, the DP algorithm becomes unsuccessful for medium/large values of $R$. Nevertheless, the DP algorithm would be useful for small values of $R$. We will show this behavior later in a computational experiment.

As an alternative approach to the DP scheme, we propose to use the $K$-ZIO algorithm introduced in the previous section to develop a method, which solves the Constrained-ELSP to optimality. Further, we use pruning strategies introduced in Sedeño-Noda and Alonso-Rodríguez [11] to avoid the enumeration of non-promising ZIO policies in the $K$-ZIO algorithm.

We need additional notation here. Let $W(P_k) = \sum_{i \in P_k} w_i$ be the left-hand side value of the knapsack constraint for the $k$th best path obtained by the $K$-ZIO algorithm. Similarly, let $W_f(P_k) = \sum_{i \in P_f} w_i$ be the sum restricted to the fixed nodes in the fixed sub-path $P_f^k$ (see Section 4). Clearly, the algorithm determines
the optimal solution for the first ZIO policy enumerated \( P^k \) satisfying \( W(P^k) \leq R \). Let \( w^*_i \) be the length of the optimal path from 1 to \( i \) in \( H \) where the cost of the arc \((u,v)\) is \( w_u \), for all \((u,v)\in A\). Obviously, \( w^*_i \) is the minimum consumption of the resource \( R \) among all the paths from 1 to \( i \) in \( H \). The calculation of the vector \( w^* \) is made simultaneously when is computed the optimal ZIO police without the knapsack constraint.

The proposed algorithm only explores those paths containing promising feasible fixed sub-paths. In order words, a path \( P^k \) with a fixed sub-path starting at node \( i \) is discarded when \( W_f(P^k) + w^*_i > R \). This pruning strategy is denominated pruning by infeasibility (see [11]).

The new approach also uses another pruning strategy, which is called pruning by dominance. Accordingly, the algorithm stores for any node \( i \) in \( H \) the best value of \( W_f(P^k) \) for any enumerated fixed sub-path starting at node \( i \), which is denoted by \( \text{best}_w_i \). Clearly, any enumerated path \( P \) must contain a fixed non-dominated sub-path. If the fixed sub-path starts at node \( i \), then the path \( P \) must satisfy \( W_f(P) < \text{best}_w_i \) since the paths are enumerated in non-decreasing order of costs \( C(P) \). Conversely, when a fixed sub-path starting at node \( i \) of \( P^k \) satisfies \( W_f(P^k) \geq \text{best}_w_i \), the path \( P^k \) is discarded.

Thus, the \( K \)-ZIO algorithm including the above pruning strategies becomes an efficient exact algorithm to solve the Constrained-ELSP as shown in the next computational experiment.

In this experiment, we consider only one instance with 25, 50 and 100 periods, respectively, from the experiment in Section 4.1. The values of \( w_i \) were generated from \( DU(0,100000) \). For each instance, we define \( \text{Max}R \) as the maximum consumption of the resource in the optimal ZIO solution (we make a lexicographic optimization here). That is, \( \text{Max}R \) is the minimal resource consumption among optimal ZIO policies. Similarly, \( \text{Min}R \) is the minimum consumption of the resource constraint and it equals to \( w^*_n+1 \). Therefore, the values of \( R \) for each instance were obtained from \( R = p(\text{Max}R - \text{Min}R) + \text{Min}R \) for percentages \( p \in \{0.1, 0.2, \ldots, 0.9\} \). Low (high) values of \( p \) mean that the resource consumption constraint is tight (loose).

The results from the experiment are shown in Table 5.1. The first column refers to the number of periods; the second column shows the cost of the optimal ZIO solution without constraint and the values of \( \text{Max}R \) and \( \text{Min}R \), whereas the third column shows the value of \( R \) in the Constrained-ELSP. In the next two columns are the value of the optimal solution and the resource consumption of the Constrained-ELSP, respectively. The last two columns indicate the CPU time in seconds of the proposed algorithm (PA) and the DP algorithm, respectively.

The proposed algorithm needs less than 0.75 seconds to solve any instance. The results reveal that the new algorithm to solve the Constrained-ELSP is very fast and practical. It is observed that the CPU times and the number of enumerated paths used by the algorithm increase as the value of \( R \) decreases, with the exception of the smallest value of \( R \) for each value of \( n \). The reason is that the strategy of pruning by infeasibility discards most of the candidate paths in the most restrictive cases, since these paths often contain infeasible fixed sub-paths. Instead, the DP algorithm needs at most 57 seconds to solve any case. The proposed algorithm is always faster than the DP algorithm. In this case, the running time consumed by the DP algorithm decreases as \( R \) decreases, which is consistent with its theoretical complexity.
Table 4. Experiment results for each pair \((n, R)\).

| \(n\) | \(R\) | Optimal Cost | Resource Consump. | \(PA\) time(s) | \(DP\) time(s) |
|---|---|---|---|---|---|
| 25 | \(C(P1) = 351014\) | 720672 | 351344 | 707712 | 0.00 | 0.64 |
| | \(MaxR = 790991\) | 650354 | 351822 | 614810 | 0.00 | 0.50 |
| | \(MinR = 87809\) | 580036 | 353534 | 555182 | 0.00 | 0.43 |
| | | 509718 | 354195 | 434312 | 0.00 | 0.34 |
| | | 369081 | 356091 | 363770 | 0.00 | 0.30 |
| | | 257863 | 359051 | 296180 | 0.00 | 0.26 |
| | | 228445 | 365764 | 225688 | 0.00 | 0.20 |
| | | 158127 | 376905 | 149418 | 0.00 | 0.15 |
| 50 | \(C(P1) = 1505508\) | 1366251 | 1506134 | 1330674 | 0.00 | 4.96 |
| | \(MaxR = 1508301\) | 1224202 | 1506831 | 1210488 | 0.00 | 4.70 |
| | \(MinR = 87809\) | 1082153 | 1507974 | 1081825 | 0.00 | 3.87 |
| | | 940104 | 1509988 | 939444 | 0.01 | 3.40 |
| | | 798055 | 1512701 | 797902 | 0.02 | 2.73 |
| | | 656005 | 1517148 | 653103 | 0.02 | 2.20 |
| | | 513956 | 1525302 | 511455 | 0.01 | 1.61 |
| | | 371907 | 1544098 | 370770 | 0.01 | 1.05 |
| | | 229858 | 1571338 | 225693 | 0.01 | 0.69 |
| 100 | \(C(P1) = 5180639\) | 2468624 | 5181363 | 2457749 | 0.00 | 57.08 |
| | \(MaxR = 2733159\) | 2204089 | 5182719 | 2203037 | 0.02 | 48.74 |
| | \(MinR = 87809\) | 1939554 | 5184763 | 1937453 | 0.08 | 43.48 |
| | | 1675019 | 5188282 | 1662572 | 0.16 | 37.67 |
| | | 1410484 | 5193502 | 1408381 | 0.24 | 30.42 |
| | | 1145949 | 5206379 | 1145394 | 0.41 | 23.21 |
| | | 881414 | 5226838 | 880005 | 0.61 | 16.78 |
| | | 616879 | 5261660 | 616134 | 0.71 | 10.34 |
| | | 352344 | 5344062 | 347168 | 0.50 | 4.18 |

5.2. A polynomial algorithm for the constrained-ELSP with equal usability for all periods. In the particular case when \(w_k = w\) for all periods, the knapsack constraint (15) can be replaced by \(\sum_{k=1}^{n} y_k \leq \bar{R}\), with \(\bar{R} = \lceil R/w \rceil\). Note that \(\bar{R} < n\) since \(wn \geq \sum_{k=1}^{n} y_k > R\) by hypothesis. It is easy to prove again that an optimal solution satisfying the ZIO property exists by checking that such a solution corresponds to the shortest path from 1 to \(n + 1\) in \(H\) using at most \(\bar{R}\) arcs. Note that any path \(P\) from 1 to \(n + 1\) in \(H\) determines a production schedule such that the number of active periods is the number of nodes in \(P\) minus 1 (excluding period \(n + 1\)). This value equals the number of arcs in \(P\). Therefore, this particular case of the Constrained-ELSP can be solved using recurrence equations similar to those given by Karp [7] and Romeijn et al. [9]. Accordingly, we denote by \(dist^k(i)\) the length of the shortest path from node 1 to node \(i\) in \(H\) using at most \(k\) arcs. In particular, setting \(dist^0(1) = 0\) and \(dist^0(i) = +\infty\), for all \(i > 1\), leads to the following recurrence equation:
\[ \text{dist}^k(i) = \min \left\{ \text{dist}^{k-1}(i), \min_{j<i} \{ \text{dist}^{k-1}(j) + l_{ji} \} \right\}, \text{ for all } i \in V, 1 \leq k \leq \bar{R} \]

Clearly, \( \text{dist}^{\bar{R}}(n+1) \) is the optimal value for this Constrained-ELS problem. To implement the recurrence above, we can use labels \( \text{Pred}(\cdot) \) as in Section 3 to identify the optimal path. Moreover, note that \( \text{dist}^k(\cdot) \) is computed from \( \text{dist}^{k-1}(\cdot) \), thus only two vectors are needed in the implementation of the recurrence.

**Lemma 5.1.** The Constrained-ELSP with \( w_k = w \) for all \( k \in \{1, \ldots, n\} \) can be solved in \( O(n^2 \bar{R}) \) time using \( O(n) \) space.

**Proof.** The above equations can be implemented using three nested loops. The index \( k \) of the external loop ranges from 1 to \( \bar{R} \), the index \( i \) of the intermediate loop iterates from 1 to \( n+1 \). The inner loop calculates \( \text{dist}(i) \) making \( i \) comparisons. Therefore, the algorithm runs in \( O(n^2 \bar{R}) \). The algorithm only uses \( O(n) \) space, since the values of \( l_{ji} \) are obtained in constant time as \( l_{ji} = f_j + c_j d_{ji-1} \) when \( d_{ji-1} > 0 \), and \( l_{ji} = 0 \), otherwise.

Clearly, this particular Constrained-ELS problem is solved in \( O(n^3) \) time because \( \bar{R} = O(n) \).

6. **Conclusions.** We address the ELSP with a planning horizon of \( n \) periods. We develop an algorithm to determine the Second Best ZIO policy in \( O(n \log n) \) by dynamic programming. From this result, we devise an algorithm (\( K \)-ZIO) to find the \( K \) Best ZIO policies in \( O(n(K + n)) \) time and \( O(K + n^2) \) space. The algorithm exhibits a very fast running times in practice so that it can be used to efficiently solve ELS problems with knapsack type constrains (Constrained-ELSP). We include some pruning strategies in the \( K \)-ZIO algorithm to avoid the computation of non-promising feasible solutions. These strategies speed up the algorithm for solving the Constrained-ELSP as suggested in the computational experiment. Further, this experiment confirms the superiority of the proposed algorithm over the classical Dynamic Programming scheme.

Specifically, for when the left-side hand coefficients in that constraint are equal, we propose an algorithm that runs in \( O(n^2 \bar{R}) \) time and uses \( O(n) \) space, where \( \bar{R} \) denotes the right-hand side in the basic knapsack constraint.

**Acknowledgments.** We would like to thank the anonymous referees for their valuable comments.

**REFERENCES**

[1] A. Aggarwal and J. K. Park, Improved algorithms for economic lot size problems, *Oper. Res.*, **41** (1993), 549–571.
[2] D. Eppstein, Finding the \( k \) shortest paths, *SIAM Journal on Computing*, **28** (1999), 652–673.
[3] A. Federgruen and M. Tzur, 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.
[4] C. S. Guazzelli and C. B. Cunha, Exploring \( k \)-best solutions to enrich network design decision-making, *Omega*, **78** (2018), 139–164.
[5] M. J. R. Helmarich, R. Jans, W. Van Den Heuvel and A. P. M. Wagelmans, The economic lot-sizing problem with an emission capacity constraint, *European Journal of Operational Research*, **241** (2015), 50–62.
[6] D. S. Johnson, M. Yannakakis and C. H. Papadimitriou, On generating all maximal independent sets, *Information Processing Letters*, **27** (1988), 119–123.
[7] R. M. Karp, A characterization of the minimum cycle mean in a digraph, *Discrete Mathematics*, 23 (1978), 309–311.

[8] M. M. B. Pascoal and A. Sedeño-Noda, Enumerating $K$ best paths in length order in DAGs, *European Journal of Operational Research*, 221 (2012), 308–316.

[9] H. E. Romeijn, D. R. Morales and W. Van Den Heuvel, Computational complexity of finding Pareto efficient outcomes for biobjective lot-sizing models, *Naval Research Logistics*, 61 (2014), 386–402.

[10] A. Sedeño-Noda, Ranking one million simple paths in road networks, *Asia-Pacific Journal of Operational Research*, 33 (2016), 5, 1650042-1.

[11] A. Sedeño-Noda and S. Alonso-Rodríguez, An enhanced K-SP algorithm with pruning strategies to solve the constrained shortest path problem, *Applied Mathematics and Computation*, 265 (2015), 602-618.

[12] S. Van Hoesel, *Model and algorithms for single item lot sizing problems*, Ph.D thesis, Erasmus University in Rotterdam, 1991.

[13] S. Van Hoesel and A. Wagelmans, Sensitivity analysis of the economic lot-sizings problem, *Discrete Applied Mathematics*, 45 (1993), 291–312.

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

[15] A. 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, *Oper. Res.*, 40 (1992), S145–S156.

[16] H. Wagner, A postscript to dynamic problems in the theory of the firm, *Naval Research Logistics Quarterly*, 7 (1960), 7–12.

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

Received October 2020; revised March 2021.

E-mail address: asedeno@ull.edu.es
E-mail address: jmgrrez@ull.edu.es