University of Groningen

Economic lot sizing with remanufacturing

Kilic, Onur A.; van den Heuvel, Wilco

Published in:
IIE Transactions

DOI:
[10.1080/24725854.2019.1593555](https://doi.org/10.1080/24725854.2019.1593555)

IMPORTANT NOTE: You are advised to consult the publisher's version (publisher's PDF) if you wish to cite from it. Please check the document version below.

Document Version
Publisher's PDF, also known as Version of record

Publication date:
2019

[Link to publication in University of Groningen/UMCG research database](#)

Citation for published version (APA):

Kilic, O. A., & van den Heuvel, W. (2019). Economic lot sizing with remanufacturing: structural properties and polynomial-time heuristics. *IIE Transactions*, 51(12), 1318-1331.
<https://doi.org/10.1080/24725854.2019.1593555>

Copyright

Other than for strictly personal use, it is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license (like Creative Commons).

The publication may also be distributed here under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license. More information can be found on the University of Groningen website: <https://www.rug.nl/library/open-access/self-archiving-pure/taverne-amendment>.

Take-down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from the University of Groningen/UMCG research database (Pure): <http://www.rug.nl/research/portal>. For technical reasons the number of authors shown on this cover page is limited to 10 maximum.

Economic lot sizing with remanufacturing: Structural properties and polynomial-time heuristics

Onur A. Kilic & Wilco van den Heuvel

To cite this article: Onur A. Kilic & Wilco van den Heuvel (2019) Economic lot sizing with remanufacturing: Structural properties and polynomial-time heuristics, IIE Transactions, 51:12, 1318-1331, DOI: [10.1080/24725854.2019.1593555](https://doi.org/10.1080/24725854.2019.1593555)

To link to this article: <https://doi.org/10.1080/24725854.2019.1593555>

Accepted author version posted online: 03
Apr 2019.
Published online: 07 Jun 2019.

Submit your article to this journal [↗](#)

Article views: 121

View related articles [↗](#)

View Crossmark data [↗](#)

Economic lot sizing with remanufacturing: Structural properties and polynomial-time heuristics

Onur A. Kilic^a and Wilco van den Heuvel^b

^aDepartment of Operations, University of Groningen, Groningen, The Netherlands; ^bEconometric Institute, Erasmus University Rotterdam, Rotterdam, The Netherlands

ABSTRACT

We consider the economic lot sizing problem with remanufacturing, an NP-hard problem which appears in integrated manufacturing and remanufacturing systems. We identify the network flow structure of the problem and derive important properties of its optimal solution. These properties are used to decompose the problem and to show that the resulting subproblems can be solved in polynomial-time. However, the number of subproblems to be solved is exponential, which we overcome by evaluating a limited set of subproblems such that the overall complexity is kept polynomial. This approach leads to a class of polynomial-time heuristics, where the time complexity depends on how the set of subproblems are chosen and how individual subproblems are solved. As a result, a trade-off can be made between computation time and solution quality. A numerical study, where we compare several heuristics within our class of heuristics, shows that our heuristics provide almost optimal solutions and significantly outperform earlier heuristics.

ARTICLE HISTORY

Received 22 May 2018
Accepted 4 March 2019

KEYWORDS

Lot sizing;
remanufacturing; heuristic

1. Introduction

As a response to rapidly increasing environmental awareness, reverse logistics has emerged as an important field in the last few decades. The concept of reverse logistics encompasses all recovery processes where products and materials are moved from the point of consumption to the point of production. Among such processes, the recovery operations concerning remanufacturing are prominent in a variety of industries. As a recovery process, remanufacturing entails the recovery of used products returned from customers by re-processing them into reusable products that are as-good-as-new. Typical examples of remanufactured products involve service parts of automobiles and industrial machinery, engines, and technological products such as computer parts (De Brito *et al.*, 2005). Today, the largest international manufacturers of these products are expanding their production facilities with large-scale remanufacturing lines. As such, traditional manufacturing systems are being replaced by integrated manufacturing and remanufacturing systems (Guide and van Wassenhove, 2009).

The main challenge in integrated manufacturing and remanufacturing systems is to match supply to demand while coordinating manufacturing and remanufacturing operations. The applicability of traditional production and inventory control methods in such integrated systems is rather limited. This has motivated research efforts towards extending traditional inventory control problems with remanufacturing options.

This article considers an important and pervasive problem that appears in integrated manufacturing and

remanufacturing systems: The Economic Lot Sizing Problem with Remanufacturing (ELSR). The problem can be described as follows. A firm faces deterministic and time-varying demands and returns over a planning horizon. The planning horizon is comprised of a finite number of time periods. The demand can be met via two alternative sources: manufacturing new products and remanufacturing returned products. It is assumed that new and remanufactured products are identical, and hence, both are regarded as serviceables, i.e., entities that can be used to satisfy demands. The firm holds both serviceable and return inventories. The problem is to satisfy all demand on time while minimizing the total cost over the planning horizon which is comprised of (fixed) production costs and inventory holding costs.

1.1. Relevant literature

A large variety of ELSR versions—with slightly different definitions have been studied in the literature. Richter and Sombrutzki (2000) and Richter and Weber (2001) addressed the special case where it is assumed that an abundant amount of returned products is available at the beginning of the planning horizon. The main consequence of this assumption is that the demand over the planning horizon can be satisfied solely by the initial return inventory. Hence, it is not necessary to place manufacturing orders—yet doing so may result in lower costs. These authors showed that the optimal solution of this particular version of the problem satisfies the zero-inventory property, and provided a polynomial-time dynamic programming algorithm to compute the

optimal replenishment plan. Richter and Weber (2001) further showed that there exists an optimal switching point from remanufacturing to manufacturing. Golany *et al.* (2001) and Yang *et al.* (2005) investigated a variant of *ELSR* where it is possible to dispose returned products. Their analysis was based on the restrictive assumption that total manufacturing and remanufacturing quantities over the planning horizon are given. They showed that this variant of the problem is *NP-hard*. Yang *et al.* (2005) also developed a polynomial-time heuristic for this problem.

Teunter *et al.* (2006) addressed *ELSR* under joint and separate setup assumptions for manufacturing and remanufacturing. For the joint setup case, they showed that the optimal solution satisfies the zero-inventory property for manufacturing, and developed an exact dynamic programming algorithm which runs in polynomial-time. For the separate setup case, they proved that the optimal solution does not necessarily satisfy the zero-inventory property, and conjectured that the problem is *NP-hard*. In addition, they provided heuristic approaches that build on the well-known lot sizing heuristics Silver-Meal, Least Unit Cost, and Part Period Balancing (see Silver *et al.* 1998, Chapter 6). Teunter *et al.*'s separate setup variant of the problem is now considered as the standard *ELSR*, and it has recently been subject to detailed scrutiny. Schulz (2011) proposed a procedure that improves the performance of Teunter *et al.*'s modified Silver-Meal heuristic. Retel Helmrich *et al.* (2014) proved Teunter *et al.*'s conjecture on *NP-hardness*, and developed a mixed-integer linear programming formulation which outperforms the conventional formulation in terms of both computation time and linear relaxation bound. Fazle Baki *et al.* (2014) reported a similar complexity result and presented an alternative mixed-integer programming formulation. They also developed a constructive heuristic that runs in polynomial-time. Cunha and Melo (2016) proposed several efficient formulations for the problem and compared them in a computational study. Ali *et al.* (2018) identified some strong flow cover inequalities and showed under which conditions they are effective. Other recent work concentrated on approaching the problem by using meta-heuristics; Li *et al.* (2014) proposed a tabu search algorithm, Parsopoulos *et al.* (2015) developed a differential evolution algorithm, and Sifaleras *et al.* (2015) presented a variable neighborhood search algorithm.

1.2. Our contribution

This study contributes to the relevant literature by providing structural results and developing polynomial-time heuristics for *ELSR*. The conceptual idea behind our work is to decompose the overall problem into subproblems, each of which can be handled in polynomial-time. To this end, we first identify the network flow structure of the problem and derive important properties of an optimal solution. Then, we make use of these properties to decompose the problem and to show that the resulting subproblems can be solved in polynomial-time. This decomposition scheme does not immediately warrant a polynomial-time algorithm, as the

number of subproblems that should be evaluated is not polynomially bounded. We overcome this issue by evaluating a limited set of subproblems such that the overall complexity is kept polynomial. This approach leads to a class of polynomial-time heuristics. The time complexity of the heuristics that emanate from our approach depends on how the set of subproblems are chosen and how individual subproblems are solved. As such, our approach provides practitioners with the flexibility of adopting a heuristic that belongs to a complexity class of choice.

We demonstrate the effectiveness of our approach by means of a numerical study where we compare several heuristics within our class of heuristics against the optimal solution as well as earlier polynomial-time heuristics from the literature on two well-known sets of problem instances. The results of the numerical study show that our heuristics provide almost optimal solutions and significantly outperform earlier heuristics.

1.3. Structure of this article

The remainder of this article is organized as follows. In Section 2, we provide a formal mathematical formulation of *ELSR*. In Section 3, we discuss the network flow structure of *ELSR* and derive properties of its optimal solution. In Section 4, we introduce our decomposition scheme. In Section 5 and Section 6, we present our heuristic approach and provide simple improvements. In Section 7, we conduct a numerical study to illustrate the effectiveness of our heuristic approach. In Section 8, we conclude the article.

2. Mathematical formulation

In this section, we first provide a formal problem definition and set the notation. Then, we present a mathematical formulation of *ELSR*.

We consider a finite planning horizon of T periods indexed by $t = 1, \dots, T$. For each period t , demands and returns are denoted by D_t and R_t . For the sake of argument and flow of the article, we assume that demands and returns are positive. The structural properties derived later still hold, but proofs turn out to be more cumbersome in the case of zero returns. It is possible to meet demands by manufacturing new products or remanufacturing returned products. The products that are manufactured and remanufactured are identical and both referred to as serviceables. In each production period, a separate fixed setup cost is incurred for manufacturing and/or remanufacturing. The system carries serviceable and return inventories over the planning horizon, and a holding cost is charged on each unit of serviceable and return inventory in each time period. The cost parameters can be time-variant. For each period t , the setup costs for manufacturing and remanufacturing are denoted by K_t^m and K_t^r , and the holding costs for serviceables and returns by h_t^s and h_t^r (with the common assumption $h_t^s \geq h_t^r$). The objective is to find a production schedule that specifies the manufacturing and remanufacturing lot sizes in

Figure 1. Alternative network flow representations of the problem.

each period, such that all demand is met on time and the total cost over the planning horizon is minimized.

We note that the problem definition provided here follows the general convention in the literature and assumes that unit production (manufacturing and remanufacturing) costs are negligible (Teunter *et al.*, 2006; Schulz, 2011; Fazle Baki *et al.*, 2014; Li *et al.*, 2014; Parsopoulos *et al.*, 2015; Sifaleras *et al.*, 2015). Nevertheless, the presented work can easily be extended to cases where this assumption is relaxed.

Throughout this article, we shall use the following additional notation. For any two integers w_0, w_1 , we let $[w_0, w_1] = \{w_0, w_0 + 1, \dots, w_1\}$ whenever $w_0 \leq w_1$ and $[w_0, w_1] = \emptyset$, otherwise. For any sequence $a_1, a_2, \dots, a_n$ and set of integers $W \subseteq \{1, \dots, n\}$, we let $a_W = \sum_{w \in W} a_w$, where the summation over an empty set is zero by convention. We denote the cardinality and the power set of any set Q by $|Q|$ and 2^Q , respectively. We also let $\min Q$ and $\max Q$ be the minimum and maximum over the set Q .

We now provide a mixed-integer programming formulation of *ELSSR*. To that end, we use the following decision variables. We let y_t^m and y_t^r be binary variables indicating whether or not a manufacturing and remanufacturing setup is done in period t , and x_t^m and x_t^r stand for the associated manufacturing and remanufacturing lot sizes, respectively. We also let I_t^s and I_t^r be the serviceable and return inventory levels at the end of period t . Then the following is a valid formulation of *ELSSR*:

$$\min \sum_{t=1}^T \{K_t^m y_t^m + K_t^r y_t^r + h_t^s I_t^s + h_t^r I_t^r\}, \quad (1a)$$

$$\text{s.t. } I_t^s = I_{t-1}^s + x_t^m + x_t^r - D_t \quad t \in [1, T], \quad (1b)$$

$$I_t^r = I_{t-1}^r - x_t^r + R_t \quad t \in [1, T], \quad (1c)$$

$$x_t^s \leq D_{[t,T]} y_t^m \quad t \in [1, T], \quad (1d)$$

$$x_t^r \leq D_{[t,T]} y_t^r \quad t \in [1, T], \quad (1e)$$

$$I_0^s = 0, I_0^r = 0, \quad (1f)$$

$$x_t^m, x_t^r, I_t^s, I_t^r \geq 0; y_t^m, y_t^r \in \{0, 1\} \quad t \in [1, T]. \quad (1g)$$

The model can be described as follows: Equation (1a) is the objective function expressing the total cost, Equations (1b) and (1c) are inventory balance equations for serviceables and returns, respectively, Equations (1d) and (1e) enforce setups for manufacturing and remanufacturing periods, Equation (1f) assumes zero initial serviceable and return inventories, and Equation (1g) sets variable domains.

3. Structural properties of an optimal solution

In this section, we present structural properties of an optimal solution to *ELSSR* by interpreting it as a minimum cost network flow problem. We shall use these properties when decomposing the problem and developing our heuristic approach.

3.1. Network flow representation and extreme point solutions

As is the case for many other lot sizing problems, *ELSSR* can be viewed as a network flow problem with concave costs. In Figure 1, we provide two different network representations of a four-period *ELSSR* instance. These network representations illustrate serviceable and return nodes (upper and lower rows respectively), demand and return flows (dashed lines), and arcs that correspond to variable manufacturing and remanufacturing quantities and serviceable and return inventories (solid lines). In Figure 1(a), arcs corresponding to variable flows that serve as source or sink are connected by a “super node” with an inflow of $D_{[1,T]} - R_{[1,T]}$ units (or an outflow of $R_{[1,T]} - D_{[1,T]}$ units, whichever is positive). In Figure 1(b), these arcs are disconnected. It turns out that Figure 1(a) is more convenient when developing optimality properties, whereas Figure 1(b) is more useful when decomposing the problem.

As we minimize a concave cost function over a bounded convex area, an optimal solution to this network flow problem can be found among the extreme point solutions, i.e., feasible solutions that cannot be written as a convex combination of other feasible solutions. It is easy to relate extreme point solutions to the network in Figure 1(a) where each feasible solution is associated with a network consisting

Figure 2. An illustration of extreme point solutions.

of free arcs, i.e., arcs with flows that are strictly between their upper and lower bounds. In *ELSR*, free arcs are equivalent to arcs with positive flows, because there are no upper bounds on continuous decision variables. It has been shown that a network associated with an extreme point solution is cycle free (see e.g., Zangwill (1968) and Ahuja *et al.* (1993)). Such a network is called a *cycle-free solution*. Hence, to find an optimal solution we may restrict ourselves to cycle-free solutions. For further details, we refer the reader to Ahuja *et al.* (1993, Chapter 11).

3.2. Regeneration networks

We now focus our attention on an extreme point solution and its corresponding cycle-free solution. To decompose the problem, we can translate the cycle-free property to the network in Figure 1(b). Then, we observe that any extreme point solution corresponds to a forest, i.e., a collection of trees. We refer to a single tree in a forest as a *regeneration network*.

Definition 1. A regeneration network (U, V) is a tree that contains a (possibly empty) set of serviceable nodes $U \subseteq [1, T]$ and a (possibly empty) set of return nodes $V \subseteq [1, T]$.

It is important to remark that regeneration networks are defined on the nodes they contain, rather than arcs. Therefore, they are not unique. There may exist different regeneration networks where the same sets of nodes are connected as a tree, but via different sets of arcs. Also, a regeneration network may not exist for some sets of serviceable and return nodes as it could not be possible to connect them as a tree.

Figure 2 illustrates four typical examples of extreme point solutions and regeneration networks therein (demand and

return flows are omitted for the sake of visual brevity). This figure will be utilized in explaining further concepts in the remainder of this article, yet for now we are only interested in regeneration networks. For instance, in Figure 2(a) we see regeneration networks $(\{1, 2\}, \emptyset)$, $(\{4, 5\}, \emptyset)$, $(\{7\}, \emptyset)$, $(\{3, 6, 8, 9, 10\}, \{1, 2, 3, 4, 5, 6, 7, 8\})$, $(\emptyset, \{9, 10\})$.

The following proposition provides insights into the sets of serviceable and return nodes contained in a regeneration network. For notational convenience, we shall find it useful to define intervals $[u_0, u_1] = [\min U, \max U]$ and $[v_0, v_1] = [\min V, \max V]$. That is $[u_0, u_1]$ and $[v_0, v_1]$ are the tightest intervals covering U and V , respectively.

Proposition 1. Any regeneration network (U, V) satisfies $U = [u_0, u_1] \setminus Z$ and $V = [v_0, v_1]$ such that $Z \subseteq [w_0 + 1, w_1 - 1]$ where $[w_0, w_1] = [u_0, u_1] \cap [v_0, v_1]$.

Proof. We first show that $V = [v_0, v_1]$. That is, V is an interval. Let us suppose there exists a period $w \in [v_0 + 1, v_1 - 1]$ such that return node w is not in the network. The return inflow in this node should be directed to either serviceable node w or return node $w + 1$. The former implies that serviceable node w and return node w do not belong to the network. However, then v_0 and v_1 cannot be connected. The latter, on the other hand, suggests that return node $w + 1$ does not belong to the network. If we iterate the same argument over $w + 1, w + 2, \dots, v_1 - 1$, we obtain a contradiction with the fact that return node v_1 is indeed in the network.

Next we show that $Z \subseteq [w_0 + 1, w_1 - 1]$. We have by definition that $Z \subseteq [u_0 + 1, u_1 - 1]$. If there exists a period w that lies in Z such that return node w is not in the network, then serviceable nodes u_0 and u_1 cannot be connected. This immediately implies that $Z \subseteq [v_0, v_1]$. Therefore, what needs to be shown is v_0 and v_1 cannot be in Z . Let us suppose v_0 is in Z . This implies that $v_0 = \min Z$ as $Z \subseteq [v_0, v_1]$, and

hence, serviceable node $v_0 - 1$ must be in the network. Then the return inflow in return node $v_0 - 1$, which is not in the network, should be directed to either serviceable node $v_0 - 1$ or return node v_0 , while both of these nodes are in the network. Let us suppose v_1 is in Z . Then the return inflow in return node v_1 , which is in the network, should be directed to either serviceable node v_1 or return node $v_1 + 1$, but neither of these nodes are in the network. $\square$

Based on [Proposition 1](#), in what follows we shall express regeneration networks as $([u_0, u_1] \setminus Z, [v_0, v_1])$. The following propositions provide properties regarding the flows in regeneration networks that will be utilized in the analysis to follow.

Proposition 2. *Any regeneration network that is a part of an optimal extreme point solution has no serviceable inventory arc at the end of the planning horizon.*

Proof. The serviceables at the end of the planning horizon cannot be originated from manufacturing, as otherwise it results in a cycle (see [Figure 1\(a\)](#)). This suggests that they originate from returns and carried over to the end of the planning horizon after being remanufactured in some period. However, then there must be another solution where they are carried to the end of the planning horizon without being remanufactured. This solution has a lower cost as we have $h^s \geq h^r$. $\square$

Proposition 3. *Any regeneration network that is a part of an optimal extreme point solution has one of the following: (i) a manufacturing arc in a single period; (ii) a return inventory arc at the end of the planning horizon; or (iii) no manufacturing arc and no return inventory arc at the end of the planning horizon.*

Proof. First, we have from [Proposition 2](#) that no optimal solution has a serviceable arc at the end of the planning horizon. Second, a regeneration network cannot have a manufacturing arc at more than one period, as it results in a cycle (see also [Figure 1\(a\)](#)). Finally, using the same cycle argument, a regeneration network cannot have both a manufacturing arc and a return inventory arc, which completes the proof. $\square$

The flow conservation dictates that the cumulative inflow should match the cumulative outflow in a regeneration network. It follows from [Proposition 3](#) that the inflow is comprised of returns in return nodes and the flow over a single manufacturing arc, whereas the outflow is comprised of demands in serviceable nodes and the flow over the return arc at the end of the planning horizon. As demands and returns are fixed, this suggests that whether a regeneration network has a manufacturing arc in some period or a return inventory arc at the end of the planning horizon is known in advance and the associated flows are readily available. Let $\bar{x}^m = D_{[u_0, u_1] \setminus Z} - R_{[v_0, v_1]}$. Then it follows that: (i) if $\bar{x}^m > 0$ then there is a manufacturing arc in some period with flow $\bar{x}^m$; (ii) if $\bar{x}^m < 0$ then there is a return inventory arc at the end of the planning horizon with flow $-\bar{x}^m$; and (iii) if $\bar{x}^m = 0$ then there is neither a manufacturing arc in some period nor a

return inventory arc at the end of the planning horizon. This also shows that a regeneration network with $\bar{x}^m < 0$ can only exist if it contains the last return node, as otherwise leftover returns cannot be carried to the end of the planning horizon.

3.3. Classes of regeneration networks

We proceed by defining three classes of regeneration networks based on whether they contain serviceable nodes and/or return nodes, namely *serviceable networks*, *return networks*, and *joint networks*. In what follows, we investigate these classes of regeneration networks and eventually conclude that the optimal cost of any regeneration network can be obtained in polynomial-time.

3.3.1. Serviceable networks

A regeneration network is called a serviceable network if it contains only serviceable nodes. [Proposition 1](#) shows that serviceable networks are regeneration networks of type $([u_0, u_1], \emptyset)$. Therefore, the number of serviceable networks is $\mathcal{O}(T^2)$. [Figure 2](#) provides several examples of serviceable networks. These are indicated by gray arcs. For instance, in [Figure 2\(a\)](#) we have serviceable networks $([1, 2], \emptyset)$, $([4, 5], \emptyset)$, and $([7, 7], \emptyset)$. The following proposition shows that the inventory flows over a serviceable network are fixed and hence are the associated costs.

Proposition 4. *The cost of any serviceable network is readily available.*

Proof. We have from [Proposition 3](#) that any serviceable network $([u_0, u_1], \emptyset)$ has a single manufacturing arc with flow $D_{[u_0, u_1]}$. This arc must be positioned in the earliest period of the network, as it is the only source of serviceables to meet all demands. The cost of a serviceable network is thus comprised of the fixed manufacturing setup cost in period u_0 and the holding costs of remaining serviceables carried in periods $[u_0, u_1]$, and it can be written as $K_{u_0}^m + \sum_{w=u_0}^{u_1-1} h_w^s D_{[w+1, u_1]}$. $\square$

3.3.2. Return networks

A regeneration network is called a return network if it contains only return nodes. [Proposition 1](#) and [Proposition 3](#) show that return networks are regeneration networks of type $(\emptyset, [v_0, T])$. Hence, the number of return networks is $\mathcal{O}(T)$.

[Figure 2](#) provides examples of return networks. These are indicated by white arcs. In [Figure 2\(a\)](#) and [Figure 2\(c\)](#), we see return networks $(\emptyset, [9, 10])$ and $(\emptyset, [8, 10])$, respectively. It is clear that inventory flows over return networks are fixed as such networks do not involve any production. Thus the next proposition follows.

Proposition 5. *The cost of any return network is readily available.*

Proof. The cost of a return network $(\emptyset, [v_0, T])$ is comprised of the holding costs of returns that accumulate in periods $[v_0, T]$. It can thus be written as $\sum_{w=v_0}^T h_w^r R_{[v_0, w]}$. $\square$

3.3.3. Joint networks

A regeneration network is called a joint network if it contains both return nodes and serviceable nodes. **Proposition 1** suggests joint networks are regeneration networks of type $([u_0, u_1] \setminus Z, [v_0, v_1])$ where $[u_0, u_1] \setminus Z$ and $[v_0, v_1]$ are non-empty. The number of joint networks is not polynomially bounded, as the number of possible Z s is exponential in the number of periods.

The following proposition suggests that we can differentiate between two types of joint networks based on the relative positions of serviceable and return nodes.

Proposition 6. *Any joint network $([u_0, u_1] \setminus Z, [v_0, v_1])$ satisfies $1 \leq v_0 \leq u_0 \leq v_1 \leq u_1 \leq T$ or $1 \leq v_0 \leq u_0 \leq u_1 < v_1 = T$.*

Proof. We first show that $u_0 \leq v_1$ and $v_0 \leq u_0$ hold. The former is straightforward; if $v_1 < u_0$ then serviceable nodes $[u_0, u_1] \setminus Z$ and return nodes $[v_0, v_1]$ cannot be connected. The latter proceeds by contradiction. Let us suppose $u_0 < v_0$. This implies that the return inflow in return node $v_0 - 1$ should be directed to either serviceable node $v_0 - 1$ or return node v_0 . However, return node $v_0 - 1$ is not in the network and both serviceable node $v_0 - 1$ and return node v_0 are in the network. This is a contradiction. We now have $u_0 \leq u_1$ and $v_0 \leq v_1$ by definition, and $u_0 \leq v_1$ and $v_0 \leq u_0$ from above. To arrive at the desired result, it is thus sufficient to show that $u_1 < v_1 < T$ cannot hold. Let us suppose this is the case. Then, the network should contain all return nodes but no serviceable node in $[v_1 + 1, u_1 - 1]$. This suggests that the return inflow in these return nodes cannot be directed to serviceable nodes. But they cannot be directed to successive return nodes either as return node T does not belong to the network. Hence, the result follows. $\square$

In light of **Proposition 3** and **Proposition 6**, we also observe that the joint networks of the first type may or may not have a manufacturing arc, whereas those of the second type cannot have a manufacturing arc as they must have a return inventory arc at the end of the planning horizon.

Figure 2 provides several examples of joint networks. These are indicated by black arcs. In **Figure 2(a)**, **2(b)**, and **2(c)**, we have joint networks $([3, 10] \setminus \{4, 5, 7\}, [1, 8])$, $([4, 6], [1, 4])$ and $([7, 10] \setminus \{8, 9\}, [5, 10])$, $([1, 10], [1, 7])$. These are all of the first type of joint networks specified in **Proposition 6**. In **Figure 2(d)**, we have joint network $([1, 7] \setminus \{3, 4, 5, 6\}, [1, 10])$ which is of the second type.

The following proposition shows that optimal flows on a joint network can be obtained by solving an equivalent remanufacturing-only lot sizing problem (see e.g., Richter and Sombrutzki (2000) once the position of its manufacturing arc (if it exists) is known.

Proposition 7. *The problem of finding the minimal cost of a joint network is equivalent to solving a remanufacturing-only lot sizing problem if the network has no manufacturing arc or has a manufacturing arc in a given period.*

Proof. We proceed with the proof as follows. We begin with the problem of finding the minimum cost of an arbitrary joint network. Then we reduce this problem to those of a

series of joint networks with no manufacturing arc where the set of serviceable nodes and return nodes both lie in the same closed interval, which indeed is a manufacturing-only lot sizing problem.

We first consider a joint network $([u_0, u_1] \setminus Z, [v_0, v_1])$ with a manufacturing arc. Let $[u_0, u_1] \setminus Z$ have N sub-intervals separated from each other by periods in Z and denote them by $[u_0^1, u_1^1], \dots, [u_0^N, u_1^N]$. As serviceable nodes in consecutive sub-intervals are not connected to each other over serviceable inventory arcs, serviceables cannot flow from one sub-interval to the next. Thus, the total production in any sub-interval $[u_0^n, u_1^n]$ should match the total demand $D_{[u_0^n, u_1^n]}$. Now suppose that the network has a manufacturing arc in some period α which lies in some sub-interval $[u_0^n, u_1^n]$. **Proposition 3** suggests that the flow on this arc is $\bar{x}^m = D_{[u_0, u_1] \setminus Z} - R_{[v_0, v_1]}$. Note that $\bar{x}^m$ should be strictly lower than $D_{[u_0^n, u_1^n]}$ as otherwise serviceable nodes $[\alpha, u_1^n]$ would be disconnected from the network and such a network thus cannot exist. Let us assume, without loss of generality, that demands from period α onwards are primarily met by the serviceables originated from the manufacturing arc. Let β be the first period where demand cannot be fully covered by manufactured serviceables. That is $\beta = \min\{w : w \geq \alpha \text{ and } \bar{x}^m < D_{[\alpha, w]}\}$. Then, the production and holding costs associated with the serviceables originated from the manufacturing arc can be written as $K_\alpha^m + \sum_{w=\alpha}^{\beta-1} h_w^s (\bar{x}^m - D_{[\alpha, w]})$. We can now disregard the demands met by these serviceables and reduce the network to a joint network with no manufacturing arc. This network is the same as the original one except $D_\alpha, \dots, D_{\beta-1}$ are offset and D_β is replaced by $D_{[\alpha, \beta]} - \bar{x}^m$.

We now consider a joint network $([u_0, u_1] \setminus Z, [v_0, v_1])$ with no manufacturing arc. Let $[w_0, w_1] = [u_0, u_1] \cap [v_0, v_1]$. All demand in this network should be met by remanufacturing in periods $[w_0, w_1] \setminus Z$, as these are the only periods where both serviceable and return nodes are in the network. As the network has no manufacturing arc, total remanufacturing in those periods should match total demand $D_{[u_0, u_1]}$. **Proposition 6** suggests that nodes that are not in $[w_0, w_1]$ are return nodes $[v_0, w_0 - 1]$ and, depending on the type of the network, either serviceable nodes $[w_1 + 1, u_1]$ or return nodes $[w_1 + 1, v_1]$. Hence, the inventories carried in these nodes are readily available. That is, the return inventories in periods $w \in [v_0, w_0 - 1]$ and, if applicable, in $w \in [w_1 + 1, v_1]$ are $R_{[v_0, w]}$ and $R_{[v_0, w]} - D_{[u_0, u_1]}$, respectively, and, if applicable, the serviceable inventory in periods $w \in [w_1 + 1, u_1]$ is $D_{[w_1 + 1, u_1]}$. Thus, we can account for all inventory holding costs in periods that do not lie in $[w_0, w_1]$ and disregard the associated inventories. This can be done by removing return nodes $[v_0, w_0 - 1] \cup [w_1 + 1, v_1]$ and serviceable nodes $[w_1 + 1, u_1]$ from the network and replacing R_{w_0} by $R_{[v_0, w_0]}$ and D_{w_1} by $D_{[w_1, u_1]}$.

We have so far shown that any joint network can be reduced to an equivalent joint network of the form $([u_0, u_1] \setminus Z, [u_0, u_1])$ that has no manufacturing arc. We now consider such a joint network. Let sub-intervals of $[u_0, u_1] \setminus Z$ are defined as above. The total remanufacturing in any sub-interval $[u_0^n, u_1^n]$ of this network should match the

total demand $D_{[u_0^n, u_1^n]}$, as all production has to be done by remanufacturing. This implies that for any period w with $u_1^{n-1} < w < u_0^n$, the return flow is $R_{[u_0, w]} - D_{[u_0, u_1^{n-1}] \setminus Z}$. Thus, we can explicitly account for the holding costs of returns in return nodes in Z and disregard the associated inventories. We can do this by removing all return nodes in Z and replacing $R_{u_0^n}$ with $R_{[u_0, u_0^n]} - D_{[u_0, u_1^{n-1}] \setminus Z}$. Then, what is left is to find the minimal total cost over all sub-intervals. As the initial serviceable inventory is zero, and the initial return inventory is fixed in each sub-interval, the costs over different sub-intervals are independent. Therefore, what we have for each sub-interval n is a joint network of the form $([u_0^n, u_1^n], [u_0^n, u_1^n])$. This is by definition a remanufacturing-only lot sizing problem. We can thus conclude that the minimal total cost of a joint network can be obtained by solving a series of remanufacturing-only lot sizing problems with a total number of periods equal to the number of periods in the network with both serviceable and return nodes. $\square$

The remanufacturing-only lot sizing problem is mathematically identical to the lot sizing problem with inventory bounds. This problem (with non-speculative costs) is known to have a time complexity of $\mathcal{O}(T)$ (Hwang and van den Heuvel, 2012). Thus, it follows that the cost of a joint network for a given manufacturing period can also be computed in $\mathcal{O}(T)$ time. As we have $\mathcal{O}(T)$ possible manufacturing periods, we can conclude that the cost of a joint network can be computed in $\mathcal{O}(T^2)$ time.

4. Decomposition of the problem

We now decompose *ELSR* into a polynomially bounded number of subproblems and devise a dynamic program which can solve the overall problem in polynomial-time, once we have the optimal costs of all subproblems.

4.1. Regeneration cycles

Thus far, we have established the decomposition of extreme point solutions into regeneration networks; however, we have an exponential number of regeneration networks, in the class of joint networks. We now bundle regeneration networks into a polynomially bounded number of collections that we refer to as *regeneration cycles*.

Definition 2. A regeneration cycle is a minimum cost collection of regeneration networks that contains a (possibly empty) set of serviceable nodes $[u_0, u_1] \subseteq [1, T]$ and a (possibly empty) set of return nodes $[v_0, v_1] \subseteq [1, T]$ such that serviceable nodes u_0, u_1 and return nodes v_0, v_1 are all in the same regeneration network.

We can better understand the concept of regeneration cycles by observing the grid-network structure of the underlying network flow problem. It is then obvious that regeneration cycles correspond to independent minimal cost subgrids that can be used to construct an extreme point solution. The number of these sub-grids is indeed polynomially bounded in the number of periods.

Figure 2 illustrates various regeneration cycles. It indicates a regeneration cycle by dotted lines surrounding the collection of regeneration networks that the regeneration cycle accommodates. For instance, in Figure 2(a) we have regeneration cycles $([1, 2], \emptyset)$, $([3, 10], [1, 8])$, and $(\emptyset, [9, 10])$.

It is important to observe that any regeneration cycle is coupled with a *covering* network that confines all other regeneration networks in the regeneration cycle. The covering network is a serviceable network if the regeneration cycle contains only serviceable nodes, and a return network if the regeneration cycle contains only return nodes. We have from Proposition 1 that the nodes in serviceable and return networks lie in closed intervals. Hence, regeneration cycles that contain only serviceable nodes and only return nodes are comprised of a single serviceable network and a single return network, respectively. Then, we have from Proposition 4 and Proposition 5 that the costs of such regeneration cycles are readily available.

For regeneration cycles with both serviceable and return nodes, the covering network is a joint network. We have from Proposition 1 that this network contains all return nodes in the regeneration cycle. Hence, the rest of the regeneration networks in the regeneration cycle should be serviceable networks. This suggests that the problem of finding the regeneration cycle is equivalent to finding a subset of serviceable nodes such that the sum of the optimal cost of the joint network that does not contain this subset of serviceable nodes and the total costs of the serviceable networks that do contain them is at minimum.

Proposition 7 provides insights into the problem of finding the optimal cost of a joint network to a remanufacturing-only lot sizing problem. The following proposition, on the other hand, reflects on the problem of finding a set of serviceable networks that contain a given set of serviceable nodes and shows that this problem is equivalent to a manufacturing-only lot sizing problem (see e.g. Wagner and Whitin (1958)).

Proposition 8. The problem of finding a minimum cost collection of serviceable networks that contains all serviceable nodes in Z is equivalent to solving a manufacturing-only lot sizing problem.

Proof. Let Z have N sub-intervals separated from each other by periods that are not in Z and denote them by $[z_0^1, z_1^1], \dots, [z_0^N, z_1^N]$. As serviceable nodes in consecutive subsets are disconnected, serviceable networks in different sub-intervals should also be disconnected, and there should be at least one serviceable network in each subset. This implies that once we find the minimum total cost of serviceable networks that contain all serviceable nodes $[z_0^n, z_1^n]$ for each sub-interval n , we also have the minimum total cost of serviceable networks that contain all serviceable nodes in Z . The problem on any sub-interval $[z_0^n, z_1^n]$ is indeed a manufacturing-only lot sizing problem. Hence, the minimum total cost of serviceable networks that contain all serviceable nodes Z can be obtained by solving a series of manufacturing-only lot sizing problems with a total number of periods smaller than the number of periods in Z . $\square$

The manufacturing-only lot sizing problem (with non-speculative costs) is known to have a time complexity of $\mathcal{O}(T)$ (Wagelmans *et al.*, 1992). Thus, it follows that the optimal cost of a joint network can also be obtained in $\mathcal{O}(T)$ time.

We now formalize the problem of finding a regeneration cycle that contains both serviceable and return nodes. Let us consider the regeneration cycle $([u_0, u_1], [v_0, v_1])$ and its covering network $([u_0, u_1] \setminus Z, [v_0, v_1])$. The problem of finding this regeneration cycle is equivalent to finding a subset $Z \subseteq [w_0 + 1, w_1 - 1]$ where $[w_0, w_1] = [u_0, u_1] \cap [v_0, v_1]$ such that the sum of the optimal cost of the joint network $([u_0, u_1] \setminus Z, [v_0, v_1])$ and the optimal total cost of serviceable networks that contain all serviceable nodes in Z is minimized.

Let us, with some abuse of notation, denote the cost of a regeneration cycle by $w(c)$ and the cost of its covering joint network by $w(c, Z)$ where $c = (u_0, u_1, v_0, v_1)$, and the minimum total cost of serviceable networks that contain all serviceable nodes Z by $w(Z)$. Then, we can write the problem of finding a regeneration cycle as

$$w(c) = \min_{Z \subseteq [w_0+1, w_1-1]} \{w(c, Z) + w(Z)\}. \quad (2)$$

Proposition 7 and **Proposition 8** show that $w(c, Z)$ and $w(Z)$ can be obtained in polynomial-time. Nevertheless, computing $w(c)$ is still quite involved, as the number of possible Z s is exponential in the number of periods. The concept of regeneration cycles is a building block of our heuristic approach, wherein (approximately) solving **Equation (2)** is of utmost importance. To avoid interrupting the present argument, we will address this in the next section.

4.2. A dynamic program

We now provide a polynomial-time dynamic program that could solve *ELSR*, once the costs of all regeneration cycles are available.

Let $f(u_0, v_0)$ be the minimum cost regeneration cycles which cover all serviceable nodes $[u_0, T]$ and all return nodes $[v_0, T]$. Then $f(\cdot, \cdot)$ satisfies the functional equation:

$$f(u_0, v_0) = \min_{(u_0, u_1, v_0, v_1) \in \Pi} \{w(u_0, u_1, v_0, v_1) + f(u_1 + 1, v_1 + 1)\}, \quad (3)$$

with the terminal cost $f(T + 1, T + 1) = 0$. Here, Π stands for the index set of all regeneration cycles. In addition, the cost of a regeneration cycle with both serviceable and return nodes is denoted by $w(u_0, u_1, v_0, v_1)$, the cost of a regeneration cycle with only return nodes is denoted by $w(u_0, u_0 - 1, v_0, v_1)$ for any u_0 so that $[u_0, u_0 - 1]$ is empty, and the cost of a regeneration cycle with only serviceable nodes is denoted by $w(u_0, u_1, v_0, v_0 - 1)$ for any v_0 so that $[v_0, v_0 - 1]$ is empty.

It should be easy to see that the dynamic program above is a shortest path problem where regeneration cycles correspond to arcs. If we disregard the time to compute the values

of $w(\cdot)$, the time complexity of solving *ELSR* with the presented dynamic program is $\mathcal{O}(T^4)$.

5. Heuristic approach

It follows from the analysis in the previous section that *ELSR* can be solved in polynomial-time once the costs of all regeneration cycles are known. We cannot yet hope (unless $P = NP$) for an optimal polynomial-time algorithm, as the problem is known to be *NP*-hard (Retel Helmrich *et al.*, 2014). This suggests that computing the costs of regeneration cycles is the bottleneck in solving *ELSR*. Based on this, we design our heuristic approach by *approximately* computing the costs of regeneration cycles in polynomial-time.

5.1. Approximating costs

It is evident from **Equation (2)** that the difficulty in finding the cost $w(c)$ of a regeneration cycle $c = (u_0, u_1, v_0, v_1)$ stems from the need to evaluate an exponential number of Z s. Furthermore, for each regeneration network (c, Z) we need to compute $w(Z)$ and $w(c, Z)$. We make use of two fundamental methods of (partly) by-passing these and approximately computing $w(c)$ in polynomial-time: (i) restricting the search space for Z ; and (ii) computing $w(Z) + w(c, Z)$ heuristically.

In the first approximation, we restrict ourselves to polynomially bounded search space for Z . This can be done by replacing **Equation (2)** by

$$w(c) = \min_{Z \in \mathcal{Z}(c)} \{w(c, Z) + w(Z)\}, \quad (4)$$

where $\mathcal{Z}(c) \subseteq 2^{[w_0+1, w_1-1]}$ such that $|\mathcal{Z}(c)|$ is polynomially bounded. As $w(Z)$ and $w(c, Z)$ can be computed in polynomial-time, this results in a polynomial-time approximation for $w(c)$.

The main idea behind this approximation is to choose a sufficiently small and polynomially bounded subset $\mathcal{Z}(c)$ that has the potential to provide a close-to-optimal value for $w(c)$. For instance, if $\mathcal{Z}(c)$ is empty, i.e., $\mathcal{Z}(c) = \emptyset$, then $w(c)$ can be computed by evaluating only $w(\emptyset)$ and $w(c, \emptyset)$, where the former is zero by definition. Alternatively, if $\mathcal{Z}(c)$ is the set of all sub-intervals of $[w_0 + 1, w_1 - 1]$, then the number of $w(Z)$ and $w(c, Z)$ that should be evaluated is quadratic in the number of periods. The choice of $\mathcal{Z}(c)$ thus appears to be an important design decision, due to number of ways to construct $\mathcal{Z}(c)$ being exponentially sized, and $|\mathcal{Z}(c)|$ having a direct effect on the complexity of the approximation. However, as we shall illustrate in our numerical study, even a primitive choice of $\mathcal{Z}(c)$ such as $\mathcal{Z}(c) = \emptyset$ leads to heuristics with an outstanding cost performance.

In the second approximation, we heuristically compute $w(Z) + w(c, Z)$. We have demonstrated that the problems associated with the computation of $w(Z)$ and $w(c, Z)$ can be transformed into well-established lot sizing problems. In particular, we can find $w(Z)$ by solving a single manufacturing-only lot sizing problem and $w(c, Z)$ by solving several remanufacturing-only lot sizing problems. This

transformation has the advantage of eliminating the need to develop tailor-made algorithms for these well-established problems, as we can make use of exact or approximate methods which are readily available in the literature. The exact computation of $w(Z) + w(c, Z)$ requires a running time of $\mathcal{O}(T^2)$, as $w(Z)$ and $w(c, Z)$ can be computed in $\mathcal{O}(T)$ and $\mathcal{O}(T^2)$, respectively. The bottleneck is thus the computation of $w(c, Z)$ and an immediate reduction method here is to limit the search for the manufacturing period of joint network (c, Z) , for example, by fixing it at the first possible period.

The approximation methods we proposed can be used simultaneously. This leads to a large class of heuristics with different levels of complexity, wherein the combination of the choice of $\mathcal{Z}(c)$ and the specific methods used to compute $w(Z) + w(c, Z)$ characterizes a heuristic.

5.2. Running times

The running time of any heuristic that emanates from our framework depends on the following components: the time complexity of the dynamic program (3), the cardinality of $\mathcal{Z}(c)$, and the time complexity of (exactly or approximately) computing $w(Z) + w(c, Z)$. Let us consider a heuristic where $|\mathcal{Z}(c)|$ is bounded by $\mathcal{O}(T^\Sigma)$ for all joint networks, and the time complexity of the method of choice for (exactly or heuristically) computing $w(Z) + w(c, Z)$ is $\mathcal{O}(T^\Delta)$. Then, computing $w(c)$ for a single joint network takes $\mathcal{O}(T^{\Sigma+\Delta})$, and because the number of joint networks is bounded by $\mathcal{O}(T^4)$, the input of the dynamic program can be obtained in $\mathcal{O}(T^{4+\Sigma+\Delta})$. As the solution of the dynamic program can be obtained in $\mathcal{O}(T^4)$, this step of the procedure is not the bottleneck. Therefore, we can conclude that the time complexity of the heuristic is $\mathcal{O}(T^{4+\Sigma+\Delta})$.

5.3. Two illustrative heuristics

We now introduce two specific heuristics that follow our framework for illustrative purposes, namely *HA* and *HB*. We shall assess the cost performance of these heuristics in our numerical study.

Let us recall that the combination of the choice of $\mathcal{Z}(c)$ and the methods used to compute $w(Z) + w(c, Z)$ characterizes a heuristic. The specifications of the illustrative heuristics regarding these are rather straightforward. In both *HA* and *HB*, we let $\mathcal{Z}(c) = \{\emptyset\}$. That is, both heuristics consider regeneration cycles that are comprised solely of a single regeneration network. Thus, what needs to be computed for each c is only $w(c, \emptyset)$ as we have $w(c) = w(c, \emptyset)$. *HA* and *HB* differ with respect to how they approach $w(c, \emptyset)$. *HA* approximates $w(c, \emptyset)$. The approximation is based on fixing the manufacturing period $\alpha = u_0$, implying that the manufacturing period is the first period where demand has to be satisfied, or, put in other words, as early as possible. The associated remanufacturing-only problem is then solved to optimality. *HB*, on the other hand, computes $w(c, \emptyset)$ exactly. This requires solving a remanufacturing-only lot sizing problem to optimality for each (feasible) manufacturing

period $\alpha \in [u_0, u_1]$. Based on these characteristics, it is clear that the time complexity of computing $w(c)$ is $\mathcal{O}(T)$ for *HA* and $\mathcal{O}(T^2)$ for *HB*. Hence, the running times of *HA* and *HB* are respectively $\mathcal{O}(T^5)$ and $\mathcal{O}(T^6)$.

6. Improvements

In this section, we provide two simple improvement schemes for the proposed class of heuristics. The first of these is based on extending the definition of joint networks and thereby considering more regeneration networks in the dynamic program. The second is based on employing an improvement procedure on the production schedule obtained by solving the dynamic program.

6.1. Extending the definition of regeneration networks

Let us recall that a joint network $([u_0, u_1] \setminus Z, [v_0, v_1])$ with $\bar{x}^m = D_{[u_0, u_1] \setminus Z} - R_{[v_0, v_1]} < 0$ can only exist if it contains the last return node, as otherwise leftover returns cannot be carried to the end of the planning horizon. It is possible to transform such a joint network into a feasible one by excluding the part of the returns that exceeds the demands and discarding the leftover returns. Let γ be the first period where accumulated returns exceed total demands in this joint network. That is $\gamma = \min\{w : w \geq v_0 \text{ and } R_{[v_0, w]} > D_{[u_0, u_1]}\}$. Then, $R_{[v_0, \gamma]} - D_{[u_0, u_1]}$ of returns in period γ together with all returns in periods $[\gamma + 1, v_1]$ should be discarded, thereby summing up to the total leftover returns $\bar{x}^m$. The cost of this should be equal to the cost of carrying these items to the end of the planning horizon through return inventory arcs. This can be written as $\sum_{w=\gamma}^T h_w^r (R_{[v_0, w]} - D_{[u_0, u_1]})$.

It should be clear that such a joint network is dominated by one or more regeneration network(s) in terms of costs. Nevertheless, as we do not consider all joint networks that can be a part of the optimal solution, taking into account such joint networks has the potential to improve the heuristic solution.

6.2. Ex-post improvement of the heuristic solution

The second improvement scheme is an ex-post procedure. It can be used to improve any feasible solution. Let us consider an initial feasible solution to *ELSR* as introduced in Equations (1a)–(1g). This solution specifies manufacturing and remanufacturing periods by means of binary variables y^m and y^r , manufacturing and remanufacturing quantities over continuous variables x^m and x^r , and, serviceable and return inventory levels over continuous variables I^s and I^r . Note that we drop period indices for simplicity. It is easy to see that we can find an alternative feasible solution that is at least as good as the initial solution if we fix binary variables y^m and y^r to their values in the initial feasible solution and optimize the model over continuous variables x^m , x^r , I^s , and I^r . As this problem includes only continuous variables, it can be efficiently solved to optimality as a linear program.

Table 1. Results on *TbVH*-instances.

Parameters	Average						Standard Deviation						Maximum					
	HA	HA ¹	HA ²	HB	HB ¹	HB ²	HA	HA ¹	HA ²	HB	HB ¹	HB ²	HA	HA ¹	HA ²	HB	HB ¹	HB ²
Demands																		
Stationary	0.64	0.26	0.15	0.04	0.01	0.00	1.27	0.63	0.50	0.22	0.07	0.06	11.69	6.83	5.83	4.62	2.03	1.92
Positive trend	1.23	0.41	0.22	0.03	0.02	0.01	3.00	0.77	0.56	0.16	0.12	0.09	37.03	8.34	8.05	3.29	3.29	3.29
Negative trend	0.56	0.25	0.15	0.02	0.01	0.00	1.29	0.58	0.46	0.10	0.07	0.05	21.19	6.71	5.16	1.65	1.40	1.38
Seasonal	0.69	0.26	0.14	0.03	0.00	0.00	1.44	0.64	0.48	0.19	0.05	0.04	21.40	6.60	6.07	4.04	1.99	1.99
Returns																		
Stationary	0.65	0.28	0.16	0.02	0.01	0.00	1.22	0.63	0.49	0.13	0.08	0.05	11.03	5.67	5.67	2.34	1.87	1.87
Positive trend	0.44	0.19	0.11	0.02	0.00	0.00	1.02	0.52	0.41	0.14	0.05	0.04	9.08	6.07	6.07	3.95	1.35	1.22
Negative trend	1.46	0.40	0.20	0.06	0.01	0.00	3.36	0.86	0.61	0.29	0.08	0.06	37.03	8.34	8.05	4.62	3.29	3.29
Seasonal	0.65	0.29	0.17	0.03	0.01	0.00	1.25	0.62	0.49	0.14	0.08	0.07	17.89	5.71	5.71	3.44	2.40	1.99
K^m																		
200	1.11	0.45	0.26	0.06	0.01	0.01	1.70	0.82	0.65	0.26	0.09	0.07	21.71	8.34	8.05	4.62	2.73	2.73
500	0.79	0.31	0.18	0.03	0.01	0.01	1.59	0.65	0.50	0.14	0.09	0.07	26.51	8.03	7.92	3.44	3.29	3.29
2000	0.39	0.11	0.05	0.00	0.00	0.00	2.08	0.38	0.23	0.06	0.01	0.01	37.03	7.62	5.37	2.46	0.58	0.52
K^r																		
200	1.17	0.49	0.28	0.03	0.02	0.01	2.49	0.85	0.66	0.15	0.12	0.09	37.03	8.34	8.05	3.29	3.29	3.29
500	0.90	0.32	0.18	0.06	0.00	0.00	1.65	0.65	0.51	0.26	0.04	0.03	23.46	6.60	5.50	4.62	1.37	1.37
2000	0.21	0.06	0.03	0.01	0.00	0.00	0.77	0.25	0.16	0.08	0.00	0.00	7.49	4.13	3.27	2.46	0.00	0.00
h^r/h^s																		
0.2	0.60	0.15	0.09	0.08	0.02	0.01	1.51	0.43	0.34	0.29	0.13	0.10	37.03	6.87	5.37	4.62	3.29	3.29
0.5	0.79	0.30	0.16	0.01	0.00	0.00	1.87	0.65	0.48	0.07	0.03	0.01	36.85	7.38	5.16	1.80	1.12	1.12
0.8	0.90	0.43	0.24	0.00	0.00	0.00	2.04	0.81	0.63	0.01	0.00	0.00	35.38	8.34	8.05	0.36	0.00	0.00
All Instances	0.76	0.29	0.16	0.03	0.01	0.00	1.83	0.66	0.50	0.18	0.08	0.06	37.03	8.34	8.05	4.62	3.29	3.29

7. Numerical study

The purpose of our numerical study is to investigate the cost performance of the proposed class of heuristics. To that end, we will assess how *HA* and *HB* (and their improved variants) perform as compared to the optimal solution and other polynomial-time heuristics currently available in the literature.

7.1. Test instances

We make use of three widely employed sets of test instances for *ELSR*. These are *TbVH*-instances (Teunter *et al.*, 2006), *S*-instances (Schulz, 2011), and *SKM*-instances (Sifaleras *et al.*, 2015). These instance sets are all based on a full factorial design. They use the same collection of time-invariant cost parameters. The considered values of manufacturing and remanufacturing setup costs are $K^m = 200, 500, 2000$ and $K^r = 200, 500, 2000$. The serviceable holding cost is fixed at $h^s = 1$. The return holding cost is set as a fraction of the serviceable holding cost such that $h^r/h^s = 0.2, 0.5, 0.8$. For each period, demand and return values are drawn from normal distributions with a pre-specified mean and a standard deviation, respectively characterizing the pattern over time and the deviation from the pattern. *TbVH*- and *S*-instances involve a variety of 12-period instances. *TbVH*-instances are designed to reflect on a variety of demand and return progressions. These involve variants of stationary, positive trend, negative trend, and seasonal patterns characterized by different degrees of variation. There are 10 demand patterns and 22 return patterns. For each demand pattern and return pattern, four series of random demands and returns are generated. This sums up to 95 040 instances in total. *S*-instances are based on stationary demands and returns, where different return-to-demand ratios are explicitly accounted for by means of average

demands and returns. The variation from the pattern is modeled by a fixed coefficient of variation over the planning horizon. There are 12 combinations of demand and return patterns. For each of these, 20 series of random demand and return realizations are generated. This setting leads to 6480 test instances in total. *SKM*-instances are designed with the same approach as *TbVH*- and *S*-instances. However, they are computationally more challenging as they are comprised of 52-period instances. They involve four series of random demands and returns for each combination of cost parameters, in total leading to 108 instances. For further details of *TbVH*-, *S*-, and *SKM*-instances, we refer the reader to the original publications (Teunter *et al.*, 2006; Schulz, 2011; Sifaleras *et al.*, 2015).

We have mentioned above that all available sets of instances assume time-invariant cost parameters. To assess the performance of our heuristics on problems with time-variant cost parameters, we introduce a fourth set of problem instances. We shall refer to this new instance set as *Svar*-instances, as it is a time-variant version of *S*-instances. *Svar*-instances are the same as *S*-instances, except manufacturing and remanufacturing setup costs and serviceable holding cost are all randomly drawn from uniform distributions with limits corresponding to $\pm 50\%$ of their nominal values. The return holding cost is once again set as a fraction of the serviceable holding cost, but this time on a period-by-period basis.

7.2. Comparison with the optimal solution

For each test instance, we first solve the mixed-integer program in (1a)–(1g) and compute the optimal solution *OPT*. We then implement heuristics *HA* and *HB* along with their improved variants with extended regeneration networks *HA*¹ and *HB*¹, and further improve their solutions by means of

Table 2. Results on S-instances.

Parameters	Average						Standard Deviation						Maximum					
	HA	HA ¹	HA ²	HB	HB ¹	HB ²	HA	HA ¹	HA ²	HB	HB ¹	HB ²	HA	HA ¹	HA ²	HB	HB ¹	HB ²
Demands																		
Small variance	0.67	0.28	0.15	0.02	0.00	0.00	1.24	0.64	0.48	0.13	0.03	0.01	7.30	6.33	6.33	2.57	0.91	0.61
Large variance	0.78	0.32	0.17	0.03	0.00	0.00	1.45	0.71	0.52	0.17	0.04	0.02	8.49	5.72	4.13	2.98	1.54	0.41
Returns																		
Small variance	0.70	0.30	0.15	0.02	0.00	0.00	1.32	0.67	0.47	0.15	0.03	0.01	8.49	6.33	6.33	2.98	0.91	0.41
Large variance	0.75	0.30	0.17	0.03	0.00	0.00	1.38	0.68	0.53	0.16	0.04	0.02	7.97	5.72	4.52	2.65	1.54	0.61
Return ratio																		
0.3	0.53	0.22	0.11	0.02	0.00	0.00	1.10	0.52	0.36	0.12	0.04	0.02	7.59	4.37	3.65	1.76	0.91	0.61
0.5	0.79	0.34	0.17	0.03	0.00	0.00	1.41	0.72	0.51	0.15	0.05	0.02	8.49	4.98	4.13	1.81	1.54	0.41
0.7	0.85	0.34	0.20	0.03	0.00	0.00	1.48	0.76	0.59	0.19	0.01	0.01	7.97	6.33	6.33	2.98	0.27	0.27
K^m																		
200	1.25	0.53	0.30	0.05	0.00	0.00	1.64	0.88	0.68	0.23	0.05	0.02	8.49	6.33	6.33	2.98	1.54	0.61
500	0.84	0.35	0.17	0.02	0.00	0.00	1.33	0.67	0.49	0.13	0.04	0.01	7.58	5.04	5.04	1.81	0.91	0.30
2000	0.07	0.02	0.01	0.00	0.00	0.00	0.52	0.17	0.07	0.00	0.00	0.00	7.77	3.05	1.64	0.00	0.00	0.00
K^r																		
200	0.72	0.30	0.16	0.03	0.00	0.00	1.63	0.89	0.69	0.11	0.06	0.02	8.49	6.33	6.33	1.76	1.54	0.61
500	0.87	0.31	0.16	0.06	0.00	0.00	1.39	0.64	0.47	0.24	0.02	0.01	7.97	4.25	3.53	2.98	0.30	0.30
2000	0.11	0.04	0.02	0.00	0.00	0.00	0.52	0.17	0.12	0.00	0.00	0.00	4.97	2.50	1.77	0.00	0.00	0.00
h^r/h^s																		
0.2	0.52	0.11	0.07	0.07	0.01	0.00	1.13	0.40	0.31	0.26	0.06	0.03	8.49	4.13	4.13	2.98	1.54	0.61
0.5	0.74	0.33	0.17	0.00	0.00	0.00	1.36	0.68	0.52	0.02	0.01	0.01	8.22	6.33	6.33	0.45	0.24	0.24
0.8	0.90	0.46	0.24	0.00	0.00	0.00	1.50	0.83	0.61	0.00	0.00	0.00	7.97	5.72	4.11	0.00	0.00	0.00
All Instances	0.72	0.30	0.16	0.03	0.00	0.00	1.35	0.68	0.50	0.15	0.03	0.02	8.49	6.33	6.33	2.98	1.54	0.61

the ex-post optimization approach HA^2 and HB^2 . We thus experiment on six different heuristics. For each of these heuristics, we record the (percentage) optimality gap which is defined as

$$\left(\frac{\mathcal{C}(H)}{\mathcal{C}(OPT)} - 1 \right) \times 100,$$

where $\mathcal{C}(H)$ and $\mathcal{C}(OPT)$ are the heuristic cost and optimal cost, respectively. We shall not discuss the computational times as it is possible to solve 12-period instances within a fraction of a second and 52-period instances within a minute for each heuristic under consideration.

Tables 1 to 4 present our results. As we consider a very large number of instances, it is not viable to present the optimality gap for each test instance. Hence, we opt for reporting summary statistics, namely, mean, standard deviation, and maximum, on optimality gaps over instances characterized by the same pivot parameter. The pivot parameters are specified in line with the organization of test sets. For *TBvH*-instances, these are demand and return patterns, and cost parameters. For *S*- and *Svar*-instances, they are demand and return variation, return ratio, and cost parameters. For *SKM*-instances, they involve cost parameters.

In the following, we discuss our findings. We first concentrate on *TBvH*- and *S*-instances. These involve instances with a relatively short planning horizon and time-invariant cost parameters. They do, however, reflect on how different demand and return progressions, return to demand ratios, and cost parameters. The results on these instances clearly demonstrate that proposed heuristics are very competitive (see Table 1 and Table 2). HA 's average optimality gap is less than 1%. The improved variants HA^1 and HA^2 further reduce this gap below 0.5%. HB 's performance is even more remarkable, it averages an optimality gap less than 0.05%.

The improved variants HB^1 and HB^2 almost close this gap, respectively averaging below 0.01% and 0.005%. The standard deviations of optimality gaps are in line with their averages. This suggests that the performance of the heuristics are relatively stable. These hold for all parameter settings. We observe that the maximum optimality gaps are higher for *TBvH*-instances as compared with *S*-instances. This suggests that the demand and return progressions have an impact on the worst-case performance.

Next, we comment on *Svar* and *SKM*-instances in comparison with the *S*-instances, thereby reflecting on the effects of time-variant cost parameters and instance size. The results show that time-variant costs has a detrimental effect on the heuristic performance (see Table 2 and Table 3). The average as well as maximum optimality gaps of HA and HB , and their improved variants are all higher when costs are time-variant. This is consistent for all parameter settings. The heuristics show an outstanding performance even when costs are time-variant. HA 's average optimality gap is less than 1.5%, and that of HA^1 and HA^2 are less than 1% and 0.5%. HB 's average optimality gap is less than 0.1%, and those of HB^1 and HB^2 are less than 0.05%. We can thus conclude that the proposed heuristics are well-suited for problem instances with time-variant costs. The results also provide interesting insights into the performance heuristics with respect to the instance size (see Table 2 and Table 4). We observe that the average optimality gaps of HA and HB , and their improved variants are relatively higher for 52-period instances as compared with 12-period instances, whereas their maximum optimality gaps are relatively lower. We also see that the cost improvement due to extending the definition of regeneration networks is relatively smaller for large instances. This can be observed by comparing HA and HA^1 as well as HB and HB^1 . We can attribute this result to the fact that this improvement scheme mainly focuses on

Table 3. Results on *Svar*-instances.

Parameters	Average						Standard Deviation						Maximum					
	HA	HA ¹	HA ²	HB	HB ¹	HB ²	HA	HA ¹	HA ²	HB	HB ¹	HB ²	HA	HA ¹	HA ²	HB	HB ¹	HB ²
Demands																		
Small variance	1.20	0.59	0.34	0.06	0.04	0.03	1.98	1.05	0.83	0.27	0.24	0.17	13.17	8.46	7.00	3.85	3.85	2.87
Large variance	1.33	0.64	0.36	0.06	0.04	0.03	2.14	1.12	0.88	0.30	0.21	0.19	13.98	7.30	6.87	3.73	3.34	3.34
Returns																		
Small variance	1.29	0.64	0.37	0.07	0.04	0.03	2.06	1.12	0.91	0.30	0.24	0.19	11.29	8.46	7.00	3.85	3.85	3.34
Large variance	1.25	0.59	0.33	0.06	0.03	0.02	2.07	1.05	0.80	0.27	0.20	0.17	13.98	7.30	6.12	3.01	3.01	2.87
Return ratio																		
0.3	0.95	0.46	0.28	0.10	0.07	0.04	1.59	0.82	0.67	0.36	0.29	0.20	9.86	4.55	4.36	3.85	3.85	3.34
0.5	1.11	0.51	0.26	0.05	0.03	0.02	1.92	0.94	0.71	0.25	0.18	0.17	10.81	6.65	5.44	2.95	2.45	2.45
0.7	1.74	0.87	0.51	0.04	0.02	0.02	2.50	1.37	1.10	0.22	0.17	0.17	13.98	8.46	7.00	2.88	2.88	2.88
K^m																		
200	2.01	0.84	0.44	0.11	0.04	0.01	2.49	1.31	1.01	0.40	0.27	0.16	13.17	8.46	7.00	3.85	3.85	3.34
500	1.24	0.65	0.38	0.03	0.03	0.02	2.04	1.08	0.88	0.17	0.17	0.16	13.98	6.33	5.80	2.45	2.45	2.45
2000	0.55	0.36	0.23	0.05	0.05	0.05	1.18	0.73	0.62	0.22	0.22	0.22	13.57	6.87	6.87	2.88	2.88	2.88
K^r																		
200	1.27	0.62	0.35	0.06	0.04	0.03	2.34	1.31	1.08	0.36	0.34	0.26	13.57	8.46	6.57	3.85	3.85	3.34
500	1.24	0.54	0.32	0.07	0.01	0.00	2.00	1.01	0.79	0.29	0.08	0.06	11.55	7.09	7.00	2.95	1.70	1.70
2000	0.61	0.27	0.13	0.01	0.01	0.01	1.56	0.72	0.55	0.16	0.16	0.16	13.98	6.87	6.87	2.88	2.88	2.88
h^r/h^s																		
0.2	0.84	0.30	0.18	0.13	0.05	0.02	1.74	0.81	0.64	0.42	0.29	0.19	11.29	6.33	5.43	3.85	3.85	3.34
0.5	1.23	0.59	0.33	0.02	0.02	0.02	2.01	1.02	0.80	0.13	0.13	0.12	11.55	7.20	6.57	2.63	2.63	2.63
0.8	1.73	0.96	0.54	0.04	0.04	0.04	2.31	1.28	1.04	0.22	0.22	0.22	13.98	8.46	7.00	2.88	2.88	2.88
All Instances	1.27	0.62	0.35	0.06	0.04	0.03	2.07	1.09	0.86	0.28	0.22	0.18	13.98	8.46	7.00	3.85	3.85	3.34

Table 4. Results on *SKM*-instances.

Parameters	Average						Standard Deviation						Maximum					
	HA	HA ¹	HA ²	HB	HB ¹	HB ²	HA	HA ¹	HA ²	HB	HB ¹	HB ²	HA	HA ¹	HA ²	HB	HB ¹	HB ²
K^m																		
200	1.76	1.36	1.04	0.10	0.09	0.03	0.96	0.75	0.69	0.17	0.17	0.11	3.75	2.85	2.64	0.58	0.58	0.58
500	1.05	0.85	0.71	0.04	0.03	0.01	0.51	0.48	0.48	0.10	0.10	0.02	2.22	2.04	1.79	0.42	0.42	0.08
2000	0.40	0.28	0.21	0.02	0.01	0.01	0.31	0.25	0.22	0.05	0.05	0.03	0.95	0.86	0.71	0.25	0.25	0.14
K^r																		
200	1.29	1.10	0.87	0.05	0.05	0.00	1.02	0.82	0.72	0.13	0.13	0.01	3.75	2.85	2.64	0.48	0.48	0.05
500	1.24	0.95	0.71	0.08	0.07	0.02	0.83	0.61	0.55	0.13	0.12	0.06	3.64	2.36	2.21	0.42	0.42	0.24
2000	0.66	0.44	0.38	0.03	0.02	0.02	0.50	0.39	0.40	0.10	0.10	0.10	1.72	1.45	1.45	0.58	0.58	0.58
h^r/h^s																		
0.2	1.18	0.79	0.52	0.15	0.13	0.04	0.96	0.72	0.56	0.17	0.17	0.11	3.75	2.66	2.19	0.58	0.58	0.58
0.5	1.03	0.87	0.74	0.01	0.01	0.00	0.82	0.75	0.68	0.03	0.03	0.02	2.92	2.85	2.64	0.16	0.16	0.14
0.8	0.99	0.83	0.70	0.00	0.00	0.00	0.79	0.61	0.56	0.00	0.00	0.00	3.64	2.32	2.17	0.00	0.00	0.00
All Instances	1.07	0.83	0.65	0.05	0.05	0.02	0.85	0.69	0.60	0.12	0.12	0.07	3.75	2.85	2.64	0.58	0.58	0.58

the end of horizon, which carries a smaller fraction of the total cost in large instances. The heuristics show a notable performance for large instances. *HA*'s average optimality gap is less than 1.5%, and that of *HA*¹ and *HA*² are less than 1%. *HB* as well as *HB*¹ and *HB*² all average optimality gaps less than 0.05%. Also, the maximum optimality gaps of *HA* and its improved variants are less than 5% and *HB* and its improved variants are less than 1%. These suggest that proposed heuristics perform well also on large instances and even yield a better worst-case performance.

7.3. Comparison with benchmark heuristics

To the best of the authors' knowledge, polynomial-time heuristics available in the literature are those introduced by Teunter *et al.* (2006), Schulz (2011), and Fazle Baki *et al.* (2014). Teunter *et al.*'s (2006) heuristics are adaptations of well-known lot sizing heuristics, and their numerical analysis suggests that, among others, the Silver-Meal-based heuristic yields significantly better results. Schulz's (2011) heuristics

generalize Teunter *et al.*'s Silver-Meal-based heuristic by introducing several improvement steps. These result in a significantly better performance. Fazle Baki *et al.*'s (2014) heuristic follows a different approach. It exploits the block structure of feasible solutions and constructs a heuristic solution by means of a dynamic program. We shall use Teunter *et al.*'s (2006) Silver-Meal heuristic *SM*, Schulz's (2011) improved Silver-Meal heuristic *SM*₄⁺ (best among all variants presented), and Fazle Baki *et al.*'s (2014) constructive heuristic *BL* as benchmarks.

The benchmark heuristics have already been tested on *TBvH*-instances and *S*-instances. In particular, *SM* and *BL*'s optimality gaps are available for *TBvH*-instances (Teunter *et al.*, 2006; Fazle Baki *et al.*, 2014) and *SM* and *SM*₄⁺'s optimality gaps are available for *S*-instances (Schulz, 2011). For *TBvH*-instances, *SM* and *BL*'s average optimality gaps are respectively 8.3% and 4.3%. For *S*-instances, *SM* and *SM*₄⁺'s average optimality gaps are respectively 7.5% and 2.2%. These figures are much higher than the average optimality gaps of our heuristics. For *TBvH*- and *S*-instances, *HA* and

improved variants average optimality gaps less than 1%, and *HB* and improved variants even less than 0.05%. In addition, the worst-case performance of our heuristics is also significantly better as compared with benchmark heuristics. For instance, SM_4^+ , which appears to be the best among the benchmarks, has a maximum optimality gap of 24.3% in *S*-instances, whereas for *HA* and improved variants it is less than 10% and for *HB* and improved variants it is less than 5%. Therefore, it is fair to conclude that the proposed heuristics significantly outperform earlier heuristics in terms of average and worst-case performance.

8. Conclusion

In this study, we provided structural results of an optimal solution of *ELSR*, and, building on these properties, developed a class of polynomial-time heuristics. We demonstrated that heuristics that emanate from our approach yield almost optimal solutions and significantly outperform earlier heuristics in the literature.

The work presented in this manuscript can be extended in several different ways. First, the problem definition can be extended to account for more general cost structures. Following the general convention in the literature, we assumed that production costs are comprised solely of fixed setup costs and that holding costs are linear. These assumptions are by no means necessary for the presented structural properties to hold, as the optimality of extreme point solutions and decomposition of the solution into regeneration networks immediately extends to general concave-cost flows. However, relaxing them increases the complexity of the subproblems that appear in our decomposition approach, due to the computational complexity of manufacturing-only and remanufacturing-only lot sizing problems with general concave costs being quadratic (rather than linear) in the number of periods (Wagelmans *et al.*, 1992; Hwang and van den Heuvel, 2012). It is important to note that the problem with general concave costs is significantly more challenging from a practical point of view, as it cannot be directly modeled and solved as a mixed-integer linear program. To the best of our knowledge, our heuristics are the only methods currently available in the literature that can account for this variant of the problem. Second, the structural properties of an extreme point solution presented in this article can be used to strengthen the mathematical programming formulations of the problem. For instance, it could be possible to improve the time-efficiency of the mixed-integer formulations of Fazle Baki *et al.* (2014) and Retel Helmrich *et al.* (2014). Third, the proposed heuristics can be embedded in a mathematical programming framework. The performance of our heuristics are extremely competitive, however, they require dedicated routines to generate and solve subproblems and put together a solution by solving a dynamic program. Thus, embedding these heuristics into an efficient mathematical programming formulation that can be solved by off-the-shelf solvers would significantly improve their applicability.

Acknowledgments

We thank the editor and two anonymous referees for their valuable comments, which helped improve this paper.

Notes on contributors

Onur A. Kilic is an assistant professor of operations research and operations management at the Department of Operations, University of Groningen. He holds an MSc in operations research from Hacettepe University and a Ph.D. in operations management from the University of Groningen. His research focuses on mathematical modeling and optimization with an emphasis on applications to inventory management, production management, and scheduling.

Wilco van den Heuvel is an associate professor of operations research at the Econometric Institute, Erasmus University Rotterdam. He holds an MSc in econometrics & management science and a Ph.D. in operations research from Erasmus University Rotterdam. His research interests include developments of operations research techniques with applications in production planning, supply chain coordination and cooperative game theory.

References

- Ahuja, R.K., Magnanti, T.L. and Orlin, J.B. (1993) *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, NJ.
- Ali, S.A.S., Doostmohammadi, M., Akartunal, I.K. and van der Meer, R. (2018) A theoretical and computational analysis of lot-sizing in remanufacturing with separate setups. *International Journal of Production Economics*, **203**, 276–285.
- Cunha, J.O. and Melo, R.A. (2016) A computational comparison of formulations for the economic lot-sizing with remanufacturing. *Computers & Industrial Engineering*, **92**, 72–81.
- De Brito, M.P., Dekker, R. and Flapper, S.D.P. (2005) Reverse logistics: A review of case studies, in *Distribution Logistics: Advanced Solutions to Practical Problems*, Springer, Berlin, Germany, pp. 243–282.
- Fazle Baki, M., Chaouch, B.A. and Abdul-Kader, W. (2014) A heuristic solution procedure for the dynamic lot sizing problem with remanufacturing and product recovery. *Computers & Operations Research*, **43**, 225–236.
- Golany, B., Yang, J. and Gang, Y. (2001) Economic lot-sizing with remanufacturing options. *IIE Transactions*, **33**(11), 995–1003.
- Guide, V.D.R. and van Wassenhove, L.N. (2009) The evolution of closed-loop supply chain research. *Operations Research*, **57**(1), 10–18.
- Hwang, H.-C. and van den Heuvel, W. (2012) Improved algorithms for a lot-sizing problem with inventory bounds and backlogging. *Naval Research Logistics*, **59**(3–4), 244–253.
- Li, X., Baki, F., Tian, P. and Chaouch, B.A. (2014) A robust block-chain based tabu search algorithm for the dynamic lot sizing problem with product returns and remanufacturing. *Omega*, **42**(1), 75–87.
- Parsopoulos, K.E., Konstantaras, I. and Skouri, K. (2015) Metaheuristic optimization for the single-item dynamic lot sizing problem with returns and remanufacturing. *Computers & Industrial Engineering*, **83**, 307–315.
- Retel Helmrich, M.J., Jans, R., van den Heuvel, W. and Wagelmans, A.P.M. (2014) Economic lot-sizing with remanufacturing: Complexity and efficient formulations. *IIE Transactions*, **46**(1), 67–86.
- Richter, K. and Sombrutzki, M. (2000) Remanufacturing planning for the reverse Wagner/Whitin models. *European Journal of Operational Research*, **121**(2), 304–315.
- Richter, K. and Weber, J. (2001) The reverse Wagner/Whitin model with variable manufacturing and remanufacturing cost. *International Journal of Production Economics*, **71**(1), 447–456.
- Schulz, T. (2011) A new Silver–Meal based heuristic for the single-item dynamic lot sizing problem with returns and remanufacturing. *International Journal of Production Research*, **49**(9), 2519–2533.

- Sifaleras, A., Konstantaras, I. and Mladenović, N. (2015) Variable neighborhood search for the economic lot sizing problem with product returns and recovery. *International Journal of Production Economics*, **160**, 133–143.
- Silver, E.A., Pyke, D.F. and Peterson, R. (1998) *Inventory Management and Production Planning and Scheduling*, Wiley New York NY.
- Teunter, R.H., Bayindir, Z.P. and van den Heuvel, W. (2006) Dynamic lot sizing with product returns and remanufacturing. *International Journal of Production Research*, **44**(20), 4377–4400.
- Wagelmans, A., van Hoesel, S. and Kolen, A. (1992) Economic lot sizing: An $O(n \log n)$ algorithm that runs in linear time in the Wagner-Whitin case. *Operations Research*, **40**, 145–156.
- Wagner, H.M. and Whitin, T.M. (1958) Dynamic version of the economic lot size model. *Management Science*, **5**, 89–96.
- Yang, J., Golany, B. and Yu, G. (2005) A concave-cost production planning problem with remanufacturing options. *Naval Research Logistics*, **52**(5), 443–458.
- Zangwill, W.I. (1968) Minimum concave cost flows in certain networks. *Management Science*, **14**, 429–450.