Provided for non-commercial research and education use.
Not for reproduction, distribution or commercial use.

This article appeared in a journal published by Elsevier. The attached copy is furnished to the author for internal non-commercial research and education use, including for instruction at the authors institution and sharing with colleagues.

Other uses, including reproduction and distribution, or selling or licensing copies, or posting to personal, institutional or third party websites are prohibited.

In most cases authors are permitted to post their version of the article (e.g. in Word or Tex form) to their personal website or institutional repository. Authors requiring further information regarding Elsevier's archiving and manuscript policies are encouraged to visit:

http://www.elsevier.com/authorsrights

Author's personal copy

Omega 42 (2014) 75–87

Contents lists available at SciVerse ScienceDirect

Omega

journal homepage: www.elsevier.com/locate/omega

A robust block-chain based tabu search algorithm for the dynamic lot sizing problem with product returns and remanufacturing

Xiangyong Li [a,*], Fazle Baki [b], Peng Tian [c], Ben A. Chaouch [b]

[a] School of Economics & Management, Tongji University, Shanghai 200092, China
[b] Odette School of Business, University of Windsor, Windsor, Ontario N9B 3P4, Canada
[c] Antai College of Economics & Management, Shanghai Jiao Tong University, Shanghai 200052, China

ARTICLE INFO

Article history:
Received 1 December 2011
Accepted 21 March 2013
Processed by B. Lev
Available online 2 April 2013

Keywords:
Production planning and scheduling
Dynamic lot sizing
Product returns
Remanufacturing
Tabu search

ABSTRACT

This paper studies the dynamic lot sizing problem with product returns and remanufacturing (DLRR). Given demands and returns over a planning horizon, DLRR is to determine a production schedule of manufacturing new products and/or remanufacturing returns such that demand in each period is satisfied and the total cost (set-up cost plus holding cost of inventory) is minimized.

Since DLRR with general cost functions for set-ups of manufacturing and remanufacturing is NP-hard, we develop a tabu search to produce high-quality solutions. To generate a good initial solution, we use a block-chain based method where the planning horizon is split into a chain of blocks. A block may contain either a string of manufacturing set-ups, a string of remanufacturing set-ups, or both. Given the cost of each block, an initial solution corresponding to a best combination of blocks is found by solving a shortest-path problem. Neighboring operators aim at shifting integer variables for manufacturing and remanufacturing set-ups.

We evaluate our algorithm on 6480 benchmark problems and compare it with other available algorithms. Computational results demonstrate that our algorithm produces an optimal solution in 96.60% of benchmark problems, with an average deviation of 0.00082% from optimality and it is a state-of-the-art method for DLRR.

© 2013 Elsevier Ltd. All rights reserved.

1. Introduction

Traditional manufacturing is unsustainable because it has produced significant adverse environmental impacts. Manufacturing generates more than 60% of annual non-hazardous waste [1] and causes challenges including pollution, natural resource depletion and shortages, and therefore high cost of landfill space and virgin materials [2,3]. Now, increasing international legislation and social pressure requires manufacturers to reduce the environmental impacts of their products and production process. Furthermore, global and mounting competition also requires companies to reduce product price while maintaining product quality.

Remanufacturing is defined as a process of returning a used product to at least the original equipment manufacturer (OEM) performance specification from the customers' perspective and giving the resultant product a warranty that is at least equal to that of a newly manufactured equivalent [4,5]. Remanufacturing is different from other product recovery options such as repair, refurbishing, cannibalization, and recycling [5,6]. Among these recovery types, repair, refurbishing, and remanufacturing upgrade the used products. But remanufacturing obtains the largest upgrade in quality and/or technology. Remanufacturing is able to help companies address the legislative, environmental, and competitive pressures [7–9]. For instance, it can simultaneously improve competitiveness and limits environmental damage due to production by reducing production costs via reductions in processing and raw material usage. By integrating waste back into the production cycle, remanufacturing limits landfill and cost of waste disposal. Other benefits of remanufacturing include reductions in the level of virgin materials and energy used in production. As suggested, the weight of a remanufactured product may be obtained from used components, and such products have comparable quality to equivalent new products but require 50% to 80% less energy to produce [5]. In addition, remanufacturing may obtain 20% to 80% production savings in comparison to conventional manufacturing [10]. Remanufactured products include automotive and aircraft parts, compressors and electrical motors, office furniture, tires, toner cartridges, office equipment, machine tools, cameras and others [6,11]. Research papers on remanufacturing have grown extensively over the past two decades. For an overview, we refer the reader to [12–19].

During the past few years, there is an increasing and widespread attention to the operation system, where manufactured and remanufactured products are identical. In the literature,

* Corresponding author. Tel.: +86 21 65988469.
E-mail address: xyli@tongji.edu.cn (X. Li).

0305-0483/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.omega.2013.03.003

Author's personal copy

identical manufactured and remanufactured products are also referred to as serviceable products or serviceables [20]. In this paper, we consider the problem where manufactured and remanufactured products are identical.

The presence of both set-up cost and holding cost gives rise to the lot sizing problem, which aims to determine the optimal timing and quantity of production over a number of future periods. In general, the lot sizing problem can be categorized into static and dynamic. The static lot sizing problem has a constant demand and continuous time scale. This category includes the famous economic lot scheduling problem (ELSP) and the economic order quantity (EOQ) model. When the demand is dynamic and deterministic, and the time scale is discrete, we get a dynamic lot sizing problem. Both the static lot sizing problem and the dynamic lot sizing without returns have been widely studied in the field of production and inventory control. However, the dynamic lot sizing problem with returns and remanufacturing has received relatively less attention. When a manufacturer has the choice of both manufacturing and remanufacturing, the manufacturer has to plan and monitor both serviceable products and returns. Teunter et al. [20] introduce the dynamic lot sizing problem with returns and remanufacturing (DLRR), and point out a new challenge brought by the separate set-up costs for manufacturing and remanufacturing in addition to the holding costs for carrying serviceable products and returns.

Many production planning and decision problems are very hard to solve [9,21–23]. Proofs from complexity theory as well as computational experiments have also indicated that most lot sizing problems are hard to solve [24]. The existence of remanufacturing makes the dynamic lot sizing problem much more harder to solve. Teunter et al. [20] conjecture that the DLRR with separate set-up costs for manufacturing and remanufacturing is NP-hard. In a previous work, we have proven that this problem is an NP-hard combinatorial optimization problem [25]. Therefore, it is impractical to use exact algorithms to optimally solve the DLRR within an acceptable CPU time. Alternatively, it is interesting and valuable to quickly produce good, although not necessarily optimal, solutions by implementing metaheuristics.

To the best of our knowledge, there is no existing metaheuristic in the literature for the DLRR, although some heuristics have been proposed (see Section 2). In this paper, we intend to fill this gap. The purpose of this paper is to develop an efficient and robust tabu search (TS) based algorithm to produce high-quality solutions for the DLRR in terms of minimizing the total costs of setups and holding inventory. Henceforth, we refer to this new proposed tabu search approach as TS-DLRR. In recent years, tabu search has been applied with a high degree of success to a variety of hard combinatorial optimization problems including vehicle routing problems [26–29], lot sizing problems [30–32], crane scheduling problem [33], and network design problems [34,35]. As demonstrated, the overall performance of TS and other local search (LS) methods, to some extent, depends on the quality of the initial solution. For this purpose, we first discuss a new block-chain based method to get a starting solution, from which the TS-DLRR performs consecutive search in the solution space. The underlying principle is to view each feasible DLRR solution as a chain of blocks. The block-chain based method splits the planning horizon into a chain of blocks. Each block consists of a set of consecutive periods. A block may contain either a string of manufacturing set-ups, a string of remanufacturing set-ups, or both. If a block contains both manufacturing and remanufacturing set-ups, then the string of manufacturing set-ups precede the string of remanufacturing set-ups (see Section 4.3.2 for detailed definition). If we know the cost of each block, which is composed of set-up cost and holding cost of inventory, then a solution can be obtained using a shortest path algorithm. This initial solution corresponds to a best combination of blocks, of which the total cost of set-up and holding inventory is minimum. To compute the cost of each block, we propose a linear programming (LP) model, which can be quickly and efficiently solved by solvers such as CPLEX and LINGO. Note that although this LP model is a heuristic and does not guarantee integral values for the binary variables associated with manufacturing and remanufacturing set-ups (e.g., "1" denotes an occurrence of set-up, and "0" means no set-up), our computational experiments demonstrate that the solution to this LP model has the desirable property that when the 0–1 restrictions on the variables indicating whether a manufacturing/remanufacturing occurs or not in a period are relaxed, a large number of these variables assume integer values at the optimal solution. This was the case in most of the 6480 benchmark instances we tested. In the TS-DLRR, four simple, but very efficient neighboring operators are designed to explore the neighborhood of the current solution and to generate neighboring solutions. These operators generate neighbors of the current solution by shifting binary variables for set-ups of manufacturing, remanufacturing, or both. For instance, if a manufacturing set-up occurs in some period, this set-up may be eliminated altogether, or replaced with a manufacturing setup, or shifted to another period and so on. In detail, these operators include one-shift operator, two-shift operator, exchange operator, and cross two-shift operator. Further, as a diversification strategy, a restart based strategy is considered to force the search into previously unexplored regions in the search space, when no solution improvement is found within some consecutive iterations. Finally, a series of computational experiments are carried out to systematically evaluate the performance of the TS-DLRR by comparing it with other best available algorithms in the literature. Computational results demonstrate that our TS approach is the best method over a wide range of benchmark instances.

1.1. Contributions

In our view, the contributions of this paper include:

(1) Development of a robust tabu search algorithm for an important class of production planning problems called dynamic lot sizing problems with product returns and remanufacturing. Our algorithm includes the following novel features:
   - A block-chain based method is used to generate a high-quality initial solution by splitting the planning horizon into a chain of blocks. A block may contain either a string of manufacturing set-ups, a string of remanufacturing set-ups, or both. Given that the cost of each block is known, then this method generates an initial solution by finding a best combination of blocks, which minimizes the total cost of production and holding inventory.
   - A new LP model is proposed to compute the block cost, in which the timing of manufacturing, remanufacturing, or both is determined in the meantime.
   - Four operators including one-shift, two-shift, exchange, and cross two-shift operators are designed to generate neighboring solutions in the search space. The neighboring operators aim at shifting binary variables for set-ups of manufacturing and remanufacturing. In each neighboring solution, the optimal value of continuous variables of production quantities is determined by solving an LP.

(2) Implementation and evaluation of our algorithm using a set of 6480 benchmark instances. The computational results show that
   - The proposed LP model for computing block cost has a robust performance. In 66.54% of 505 440 blocks, this method has returned integer values for binary variables of manufacturing and remanufacturing set-ups.

Author's personal copy

- The block-chain based method is an efficient approach for generating initial solutions in our TS-DLRR. Over 1350 considered instances, this method produced an average deviation of 17.46% from optimality, and outperforms the ELSP based method (e.g., a deviation of 55.06%).
- Our TS-DLRR algorithm is a state-of-the-art method and produces an optimal solution in 96.60% of 6480 benchmark problems, with an average deviation of 0.00082% from optimality.

1.2. Outline

The remainder of this paper is organized as follows. In Section 2, we review the related work in the literature. In Section 3, we give a description of the DLRR. In Section 4, we detail the proposed tabu search for the DLRR. Next, in Section 5, we present computational results to assess our proposed algorithm. Finally, we conclude this paper in Section 6 with a discussion of future research directions.

2. Literature review

For a general review of the dynamic lot sizing problem, readers may refer to [36] and [37]. For ELSP with remanufacturing, refer to [38–40]. Here, we briefly review some papers related to the dynamic lot sizing problem with returns and remanufacturing.

Richter and Sombrutzki [41] discuss a dynamic lot sizing problem with returns and present a mixed integer programming (MIP) model. However, their analysis is restricted to a special case where there are enough product returns available at the beginning of the planning period to cover demands over the entire horizon. Therefore, manufacturing is not considered. This assumption makes it possible to use the zero-inventory property, which means that there exists an optimal solution, in which replenishment takes place in a given period only if the inventory in that period is zero. They point out that this special case can be solved using a Wagner–Whitin type algorithm. Richter and Weber [42] further extend the analysis by adding variable manufacturing and remanufacturing costs. Richter and Weber [42] present results only for the special case where the number of returns in the first period is at least as large as the total demand over the planning horizon. In other words, in this special case, manufacturing may be used only when the cost for holding returned products is very large.

Golany et al. [43] further generalize the problem considered in [41,42]. They remove the restriction on the number of returns and consider a disposal option in the model. They formulate the dynamic lot sizing problem with remanufacturing as a network problem and prove that this problem is NP-hard for general concave costs. For the special case with linear costs, they develop a polynomial-time algorithm.

Yang et al. [44] further extend the work done in [43] on the concave cost functions. They observe that optimal solutions are the extreme points of the feasible region and they develop a heuristic based on this observation.

Beltran and Krass [45] consider the lot sizing problem with returns and disposal option, but do not allow remanufacturing of return. Instead, the returns join the serviceable products directly. Beltran and Krass [45] show that it is able to consider solutions satisfying the zero-inventory property. With this property, they develop a dynamic programming algorithm to determine the optimal manufacturing and disposal decisions for special cases with concave cost function.

Note that the above five papers do not provide algorithms for the DLRR without restrictions on the returns and with set-up cost. The most significant work in this stream of research is that of Teunter et al. [20]. The authors provide a broad analysis on the dynamic lot sizing problem with returns and remanufacturing (DLRR) with the assumption that the demand and returns are dynamic and deterministic. They distinguish between two cases with different set-up cost schemes. The first case has a joint set-up cost for both manufacturing and remanufacturing. In the second case, there are separate set-up costs for manufacturing and remanufacturing. Based on the zero-inventory and remanufacturing-first properties, Teunter et al. [20] give a polynomial-time dynamic programming algorithm for the problem with a joint set-up cost, and conjecture that the problem with separate set-up costs is NP-hard. For both cases, they adapt three well-known heuristics, Silver-Meal (SM), Least Unit Cost (LUC), and Part Period Balancing (PPB), which were originally designed for the dynamic lot sizing problem without returns [36]. Teunter et al. [20] conduct a series of experiments to compare the performance of these three heuristics. Their results show that for both model settings, SM performs slightly better than LUC, and much better than PPB. SM has respectively produced solutions with an average deviation of 3.0% and 8.4% from optimality for the joint and separate set-up settings.

Pang et al. [46] extend the work of [20] and consider the dynamic lot sizing problem with a disposal option and a restriction on the capacities of manufacturing, remanufacturing, and disposal. The authors present different formulations for this problem and develop a dynamic programming algorithm, which solves some instances to optimality.

Schulz [47] uses some known results of the static lot sizing problem to generalize the adapted Silver-Meal method in [20]. As a result, the average percentage deviation from optimality can be slightly reduced from 7.5% to 6.1%. The author further considers an improvement procedure to reduce the average percentage deviation to 2.2%.

As previously stated, Teunter et al. [20] conjecture that the DLRR with separate set-up costs for manufacturing and remanufacturing is NP-hard. In a previous work, we prove that the DLRR is indeed NP-hard, present an alternative mixed integer programming model, and propose a dynamic programming based heuristic [25].

3. Problem description

In this paper, we study the dynamic lot sizing problem with returns and remanufacturing (DLRR), where there are fixed cost for set-ups and holding inventory. To begin our model statement, we assume that the planning horizon spans T periods. In each period i, there is D_i number of new products demanded and R_i number of returned products. As in [20], we also assume that the amount of demand and returned products is known in advance. It should be noted that although the process of returns is usually affected by various uncertainties, firms may apply forecasting techniques along with advanced information systems and even use modified MRP-like management techniques to obtain deterministic estimate for the quantities of returned products [12,13,43]. Without loss of generality, it is assumed that the initial inventories of both serviceables and returns are zero.

Given demand D_i and returns R_i in each period i in the planning horizon T, the DLRR is to determine optimal quantity Q_i^S of products manufactured, and quantity Q_i^R of returns remanufactured in each period i such that the total set-up cost plus holding cost of inventory is minimized.

Before presenting the formulation of the DLRR, we first introduce some notations:

- $\overline{T}$: the set of periods, i.e., $\overline{T}=\{1,2,\ldots,T\}$;
- K^S: set-up cost for manufacturing new products (or serviceables);

Author's personal copy

- K^R: set-up cost for remanufacturing returned products;
- h^S: unit holding cost for serviceables per unit period;
- h^R: holding cost for returns per unit period.

We define some rational decision variables:

- I_i^S: inventory level of serviceables left over at the end of period i;
- I_i^R: inventory level of returns left over at the end of period i;
- Q_i^S: quantity of products manufactured in period i;
- Q_i^R: quantity of products remanufactured in period i;

and two binary variables:

$$y_i^S = \begin{cases} 1, & \text{if new products are manufactured in period } i, \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

and

$$y_i^R = \begin{cases} 1, & \text{if returned products are remanufactured in period } i, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Now, we are ready to give the following straightforward MIP formulation (MIP1) for the DLRR

$$(\textbf{MIP1})\ \min \quad K^S \sum_{i\in\overline{T}} y_i^S + K^R \sum_{i\in\overline{T}} y_i^R + h^S \sum_{i\in\overline{T}} I_i^S + h^R \sum_{i\in\overline{T}} I_i^R \tag{3}$$

$$\text{s.t.} \quad I_0^S = I_0^R = 0, \tag{4}$$

$$I_i^S = I_{i-1}^S + Q_i^S + Q_i^R - D_i \quad \forall i\in\overline{T}, \tag{5}$$

$$I_i^R = I_{i-1}^R + R_i - Q_i^R \quad \forall i\in\overline{T}, \tag{6}$$

$$Q_i^S \le \left(\sum_{j=1}^{T} D_i\right) y_i^S \quad \forall i\in\overline{T}, \tag{7}$$

$$Q_i^R \le \left(\sum_{j=1}^{T} R_i\right) y_i^R \quad \forall i\in\overline{T}, \tag{8}$$

$$Q_i^S, Q_i^R, I_i^S, I_i^R \ge 0 \quad \forall i\in\overline{T}, \tag{9}$$

$$y_i^S, y_i^R \in \{0,1\} \quad \forall i\in\overline{T}. \tag{10}$$

This MIP formulation is widely used in the literature of lot sizing problem [20,47]. Eq. (3) optimizes the total cost composed of set-up cost and holding cost of inventory. Eqs. (5) and (6) compute the inventory of serviceables and returns in each period. Eqs. (7) and (8) state that any new product manufactured or remanufactured must give rise to a set-up.

4. The block-chain based tabu search algorithm

Our proposed solution approach is a block-chain based tabu search metaheuristic, in which a block-chain based method is introduced to generate high-quality initial solution, and four new neighboring operators are designed to explore the whole search space. After giving some notation, we then discuss TS as a general metaheuristic procedure (Section 4.2) and describe the overall structure of our TS metaheuristic for solving the DLRR.

4.1. Notation

We first list some new notation, which is used in describing algorithm TS-DLRR.

- P: data of a DLRR instance;
- D_i^S: demand in period i satisfied by manufacturing new products;
- D_i^R: demand in period i satisfied by remanufacturing returned products;
- $f(\sigma)$: the objective value of solution σ;
- σ^b: the best solution found since the start of the TS-DLRR;
- t_0: the number of iterations without solution improvement found;
- t_{no}: the maximum allowed iterations without solution improvement found;
- $A(\sigma)$: the set of admissible solutions obtained from σ.

4.2. TS as a general-purpose metaheuristic

Building upon his previous work, Glover [48] first proposed the tabu search that allows local search methods to overcome local optima. The basic principle underlying TS is to pursue LS whenever it encounters a local optima by allowing non-improving moves. The cycling back to previously obtained solutions is prevented by the use of short-term memories, called tabu lists, that record the recent history of the search. Some attributes of past solutions are registered and any solution possessing these attributes may not be considered, and temporarily declared tabu and are stored in the tabu list [27]. For detailed implementation of TS, readers may refer to [49].

4.3. The TS approach to our setting

The global structure of our TS implementation is represented as Algorithm 1. Starting from a feasible solution σ, our basic algorithm explores the neighborhood of the current solution and typically uses a best-improvement strategy to select the best neighbor in each step. To prevent the LS methods from immediately returning to a previously visited candidate solutions and avoid cycling, the TS-DLRR uses tabu lists to forbid moves to recently visited search positions. We detail the main components of the TS-DLRR as follows:

Algorithm 1. TS-DLRR: Tabu search for the DLRR.

```
input instance data P
initialize tabu lists T_i(i = 1,2,3,4)
σ=GenerateInitialSolution(P) (see Algorithm 2)
set σ^b : = σ, and t_0 : = 0
while (termination criterion not met) do
  A(σ) = GenerateAdmissibleNeighbor(σ, P)
  σ = SelectBestSolution(A(σ))
  if f(σ) < f(σ^b), then
    set σ^b : = σ, t_0 : = 0
  else
    set t_0 : = t_0 + 1
  end if
  record tabu for the current move in tabu lists and update tabu
  lists
  if t_0 > t_no, then
  implement diversification procedure, and set t_0 : = 0
  end if
end while
return the best-known solution σ^b
```

4.3.1. Representation and evaluation of a solution

In the TS-DLRR approach, we use a direct representation of a solution σ: we only include integer variables $y_i^S(\sigma)$ and

Author's personal copy

$y_i^R(\sigma)$ for set-ups of manufacturing and remanufacturing, respectively.

Each setting of the integer variables corresponds with an optimal value for rational variables of production quantities. This solution can be found by solving a linear programming model, which is formulation (MIP1) with the integer variables fixed. Henceforth, we denote MIP1 with integer variables fixed by solution σ as LPF(σ). Accounting for all problem constraints in the definition of the search space often restricts the searching process too much and can lead to mediocre solutions. To enhance the diversification, we allow the TS-DLRR approach to explore infeasible solutions. If solution σ results in an infeasible LPF(σ), we, for purpose of simplicity, set its objective $f(\sigma)$ to ∞.

4.3.2. GenerateInitialSolution(P)

The TS-DLRR starts from a feasible solution σ to explore its neighborhood. A high-quality initial solution might accelerate the course of the solution. As one of the main contributions, we propose a block-chain based method (function GenerateInitialSolution(P)) to get a high-quality DLRR solution below. Algorithm 2 gives the overall structure of function GenerateInitialSolution(P). The basic idea is that we view a feasible solution as a chain of blocks (see definition below) and an optimal solution corresponds to a best combination of blocks.

Algorithm 2. GenerateInitialSolution(P).

```
for each block (s,t)(s,t∈T̄)
  if formulation (LP_st) (see (15) for definition) is feasible, then
    solve formulation (LP_st) for block (s,t)
    if LP_st does not return an integer solution, then
      solve a modified formulation (LP_st) with integer x_ij^S and x_ij^R
  (see (11)–(14) for definition)
    end if
    store data of manufacturing and remanufacturing in block
  (s,t)
  else
    set f(σ) : = ∞
  end if
end for
σ = BuildSolution(block data)
return solution σ
```

Typically, Algorithm 2 works as follows. We first get the cost of each feasible block by solving an LP formulation. With block costs, we then create an auxiliary network with node set $N=\{0,1,2,\ldots,T\}$ (node 0 represents the beginning of period 1) and find a shortest path from node 0 to node T (see Algorithm 3). This shortest path corresponds to the best production schedule, i.e., a combination of blocks, which has a minimum cost of set-up and holding inventory. The performance of this block-chain based method is assessed in the next section.

Below, we detail the definition of block and block chain, how to compute block cost, and how to generate initial solution given block costs.

A block (s, t) is defined as a set of consecutive periods $s, s+1, \ldots, t$ such that it contains at least one period, in which a manufacturing or remanufacturing set-up occurs and if there exists a manufacturing set-up in period $i (s \le i \le t)$ and remanufacturing set-up in period $j (s \le j \le t)$, then $i \le j$. Fig. 1 illustrates examples of block patterns. In example (a), periods 1 to 6 form a block (1, 6): manufacturing is done in periods 1 and 3, which is followed by remanufacturing in period 4. However, in example (b), periods 1–5 do not constitute a block because remanufacturing in period 4 comes before manufacturing in period 5. Therefore,

a

Period i	1	2	3	4	5	6
$y_i^S(\sigma)$	1	0	1	0	0	0
$y_i^R(\sigma)$	0	0	0	1	0	0

b

Period i	1	2	3	4	5
$y_i^S(\sigma)$	1	0	0	0	1
$y_i^R(\sigma)$	0	0	0	1	0

Fig. 1. Two examples of block-chain based solutions.

periods 1–4 constitute a block (1, 4) and period 5 forms a block (5, 5) by itself. Alternatively, for this example, we may also consider three separate blocks: periods 1–3 as one block, period 4 as another, and period 5 as the third block. With this definition, each feasible solution corresponds to a chain of blocks, i.e., a combination of blocks.

Given a block (s, t), the key question is to determine when and how many new units to manufacture and when and how many returns to remanufacture so as to minimize the cost of block (s, t). To answer this question, we use a construction procedure based on the following idea. We divide demand in any given period into two partitions. One partition will be satisfied only by newly manufactured products, while the other partition will be met by remanufacturing only. We then formulate a linear programming model that incorporate this view of demand allocation. The model is subsequently solved to find the cost of block (s, t).

The beginning and ending inventory of serviceables and returns for block (s, t) are required inputs to the method. First, let us explain how these values are obtained. In a previous work [25], we show that for every feasible solution to the DLRR $I_i^S + I_i^R \ge \gamma_i$ and that there exists a solution to DLRR for which $I_i^S = I_0^S$ and $I_i^R = \gamma_i$, $\forall i$, where γ_i is computed as follows. We set $\gamma_0 = 0$ and then recursively compute $\gamma_i, \forall i \in \overline{T}$ as $\gamma_i = \max(0, \gamma_{i-1} + R_i - D_i)$.

In our construction procedure discussed below, we show that it is always possible to set the starting and ending inventories for block (s, t) to the following target values: $I_{s-1}^S = I_0^S$, $I_t^S = I_0^S$, $I_{s-1}^R = \gamma_{s-1}$ and $I_t^R = \gamma_t$.

For $i = s, \ldots, t$, let $D_i = D_i^S + D_i^R$, where D_i^S and D_i^R represent the parts of demand that are allocated to manufacturing and remanufacturing, respectively. The demands D_i^S and $D_i^R, \forall i \in \{s, \ldots, t\}$ are inputs to the linear program that computes the cost of the block. The demands D_i^S and D_i^R are determined as follows. Let $R(s,t) = \gamma_{s-1} - \gamma_t + \sum_{k=s}^{t} R_k$. $R(s,t)$ represents the total number of returns available for remanufacturing from periods s to t. For $i = s, \cdots t$, we define

$$D_i^R = \max\left(0,\ \min\left(D_i, R(s,t) - \sum_{k=i+1}^{t} D_k\right)\right), \text{and}$$

$$D_i^S = D_i - D_i^R$$

Next, for block (s, t), we define two specific periods u and v as follows. Let $u = \max_{s \le i \le t}\{i : R(s,t) < \sum_{k=i}^{t} D_k\}$. If there is no such i that satisfies the inequality, we set $u = s-1$. Period v is chosen as follows. if $u = s-1$, then $v = s$. But, if $u \ge s$ and $D_u^S = D_u$, then $v = u+1$; otherwise $v = u$.

Now, we divide block (s, t) into two segments. In the first segment, manufacturing is performed from period s to period u, where $s-1 \le u \le t$; if $u = (s-1)$, block (s, t) will not include any manufacturing. In the second segment, remanufacturing is performed from period v to period t, where $u \le v \le t+1$. If $v = (t+1)$, block (s, t) will have no remanufacturing.

The D_i^S and D_i^R values given above satisfy the following property.

Lemma 1. *Given two periods i and j such that $s \le i < j \le t$, we have (i) if $D_i^R > 0$, then $D_j^R = D_j$; (ii) if $D_j^R < D_j$, then $D_i^R = 0$, and (iii) $\sum_{k=s}^{t} D_k^R = R(s,t)$.*

Author's personal copy

Proof. (i) If $D_i^R > 0$, then $R(s,t) - \sum_{k=i+1}^{t} D_k > 0$. Adding D_{i+1} on both sides, we get $R(s,t) - \sum_{k=i+2}^{t} D_k > D_{i+1}$. Therefore, $D_{i+1}^R = D_{i+1}$. By repeating the process, we get $D_j^R = D_j, \forall i < j \le t$.

(ii) If $D_j^R < D_j$, then $R(s,t) - \sum_{k=j+1}^{t} D_k < D_j$. By subtracting D_j from both sides, we get $R(s,t) - \sum_{k=j}^{t} D_k < 0$. Therefore, $D_{j-1}^R = 0$. By repeating the process, we get $D_i^R = 0, \forall s \le i < j$.

(iii) By definition of $D_k{}^R$ and u, we get that $D_k^R = D_k, \forall k > u$; $D_u^R = R(s,t) - \sum_{k=u+1}^{t} D_k^R$; and $D_k^R = 0, \forall k < u$; Therefore, $\sum_{k=s}^{t} D_k^R = R(s,t)$. □

Now that we have described how block (s,t) can be structured, the next theorem shows that a feasible production schedule can always be found that satisfies the target inventory levels as mentioned above.

Theorem 1. *Given $I_{s-1}^S = I_0^S \ge 0$ and $I_{s-1}^R = \gamma_{s-1} \ge 0$, the production schedule given by $Q_i^S = D_i^S$ and $Q_i^R = D_i^R, \forall s \le i \le t$ satisfies inventory constraints $I_t^S = I_0^S$, $I_t^R = \gamma_t$, $I_i^S \ge 0$ and $I_i^R \ge 0, \forall s \le i \le t$.*

Proof. Since $D_i = D_i^S + D_i^R$, it follows from Eq. (5) that $I_i^S = I_{i-1}^S + D_i^S + D_i^R - D_i = I_{i-1}^S, \forall s \le i \le t$. Therefore, $I_t^S = I_{t-1}^S = \cdots = I_{s-1}^S = I_0^S \ge 0$.

Adding Eqs. (5) and (6), substituting $I_i^S = I_{i-1}^S$ and $D_i = D_i^S + D_i^R$ and simplifying, we get $I_i^R = I_{i-1}^R + R_i - D_i^R, \forall i = s, \ldots, t$. Adding the last equation $\forall i = s, \ldots, t$, cancelling out the common terms, and substituting $I_{s-1}^R = \gamma_{s-1}$ and $R(s,t) = \gamma_{s-1} - \gamma_t + \sum_{k=s}^{t} R_k$ we get $I_t^R = \gamma_t + R(s,t) - \sum_{k=s}^{t} D_k^R$. Now, from (1), $I_t^R = \gamma_t$. □

The above theorem shows a feasible production schedule that yields the target inventory. Below, we present a LP formulation that improves the cost of the block. Before presenting the LP formulation, we need to define some binary variables

(1) For al $s-1 \le i \le j < u$, we define

$$x_{ij}^S = \begin{cases} 1, & \text{if new products are manufactured in periods } i \text{ and } (j+1), \text{ but not in periods } (i+1), \ldots, j \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

(2) For all $s-1 \le i \le u$, we define

$$x_{iu}^S = \begin{cases} 1, & \text{if new products are manufactured in periods } i, \text{ but not in periods } (i+1), \ldots, u \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

(3) For all $v-1 \le i \le j < t$, we define

$$x_{ij}^R = \begin{cases} 1, & \text{if returns are remanufactured in periods } i \text{ and } (j+1), \text{ but not in periods } (i+1), \ldots, j \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

(4) For all $v-1 \le i \le t$, we define

$$x_{it}^R = \begin{cases} 1, & \text{if returns are remanufactured in periods } i, \text{ but not in periods } (i+1), \ldots, t \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

For any period $i \ge v$, if $\sum_{k=s}^{i} R_k \ge \sum_{k=i}^{j} D_k^R$ for $j \le t$, then the remanufacturing lot can be large enough to meet the demand allocated to periods i to j. Let $\varphi(i)$ be the largest $j \le t$ such that $\sum_{k=s}^{i} R_k \ge \sum_{k=i}^{j} D_k^R$. So, $x_{i,\varphi(i)+1}^R = x_{i,\varphi(i)+2}^R = \cdots = x_{i,t}^R = 0$.

We now present an LP formulation (LP_{st}) to compute the cost c_{st} associated with block (s, t)

$$(\mathbf{LP}_{st})\ c_{st} = \min K^S \sum_{i=s}^{u} \sum_{j=i}^{u} x_{ij}^S + K^R \sum_{i=v}^{t} \sum_{j=i}^{t} x_{ij}^R + h^S \sum_{i=s}^{t} I_i^S + h^R \sum_{i=s}^{t} I_i^R \tag{15}$$

$$\text{s.t. } I_{s-1}^S = I_0^S, \tag{16}$$

$$I_{s-1}^R = \gamma_{s-1}, \tag{17}$$

$$I_i^S = I_{i-1}^S + Q_i^S + Q_i^R - D_i \quad \forall i = s, s+1, \ldots, t, \tag{18}$$

$$I_i^R = I_{i-1}^R + R_i - Q_i^R \quad \forall i = s, s+1, \ldots, t, \tag{19}$$

$$Q_i^S = \sum_{j=i}^{u} x_{ij}^S \sum_{k=i}^{j} D_k^S \quad \forall i = s, s+1, \ldots, u, \tag{20}$$

$$Q_i^R = \sum_{j=i}^{t} x_{ij}^R \sum_{k=i}^{j} D_k^R \quad \forall i = v, v+1, \ldots, t, \tag{21}$$

$$\sum_{j=s-1}^{u} x_{(s-1)j}^S = 1, \tag{22}$$

$$\sum_{i=s-1}^{j} x_{ij}^S = \sum_{k=j+1}^{u} x_{(j+1)k}^S \quad \forall j = s-1, s, \ldots, u-1, \tag{23}$$

$$\sum_{i=s-1}^{u} x_{iu}^S = 1, \tag{24}$$

$$\sum_{j=v-1}^{t} x_{(v-1)j}^R = 1, \tag{25}$$

$$\sum_{i=v-1}^{j} x_{ij}^R = \sum_{k=j+1}^{\varphi(j+1)} x_{(j+1)k}^R \quad \forall j = v-1, v, \ldots, t-1, \tag{26}$$

$$\sum_{i=v-1}^{t} x_{it}^R = 1, \tag{27}$$

$$x_{ij}^R = 0 \quad \forall i = v, v+1, \ldots, t, \ \forall j > \varphi(i), \tag{28}$$

$$Q_i^S = 0 \quad \forall i = u+1, u+2, \ldots, t, \tag{29}$$

$$Q_i^R = 0 \quad \forall i = s, s+1, \ldots, v-1, \tag{30}$$

$$I_t^S = I_0^S, \tag{31}$$

$$I_t^R = \gamma_t, \tag{32}$$

$$0 \le x_{ij}^S \le 1 \quad \forall i, j = s-1, s, \ldots, u, \tag{33}$$

$$0 \le x_{ij}^R \le 1 \quad \forall i = v-1, v, \ldots, t, \tag{34}$$

Author's personal copy

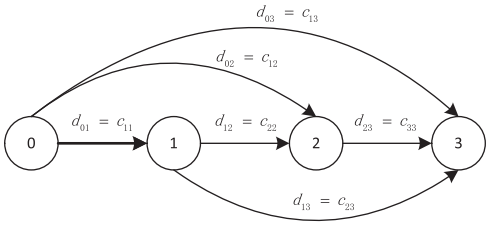

Fig. 2. Examples of directed network associated with the blocks.

$$Q_i^S, Q_i^R, I_i^S, I_i^R \geq 0 \quad \forall i = s, s+1, \ldots, t, \tag{35}$$

Eq. (15) minimizes the cost of block (s, t). Eqs. (18) and (19) compute the inventory of each period. Eqs. (20) and (21) compute the lot sizes. Eqs. (22)–(24) determine when to manufacture. Eq. (22) states that either there is no manufacturing, so $x_{(s-1)u}^S = 1$, or the first manufacturing occurs in period $(j+1)$, if $x_{(s-1)j}^S = 1$, $(s-1) \leq j \leq u$. Eq. (23) states that if a manufacturing lot covers some periods ending at period j, the next lot covers period $(j+1)$.

Eq. (24) states that either there is no manufacturing, so $x_{(s-1)u}^S = 1$ or a manufacturing in period i covers period u if $x_{iu}^S = 1, s \leq i \leq u$. Eqs. (25)–(27) determine when to remanufacture. Eq. (25) states that either there is no remanufacturing, so $x_{(v-1)t}^R = 1$, or the first remanufacturing occurs in period $(j+1)$, if $x_{(v-1)j}^R = 1$, $(v-1) \leq j \leq (t-1)$. Eq. (26) states that if a remanufacturing lot covers some periods ending at period j, the next lot covers period $(j+1)$. Eq. (27) states that either there is no remanufacturing, so $x_{(v-1)t}^R = 1$, or a remanufacturing in period i covers period t, if $x_{it}^S = 1, v \leq i \leq t$. Eq. (28) arise because if the returns accumulated up to period i is not sufficient to meet the demand from period i to j, then $x_{ij}^R = 0$. Eqs. (29) and (30) state that manufacturing is not done after period u and remanufacturing is not done before period v. Eqs. (31) and (32) ensure that the ending inventory of serviceable is I_0^S and ending inventory of returns is γ_t.

Note that this LP formulation does not guarantee integer values of x-variables. In other words, model (LP_{st}) does not necessarily result in integer values of y-variables in block (s, t). But the important advantage of the LP formulation given in (15)–(35) is that its solution yields integer values for the x-variables in a large number of test instances. In Algorithm 2, when formulation (LP_{st}) does not return integer values of x-variables, we use CPLEX to solve a modified formulation (LP_{st}) with integer x-variables (e.g., integer x_{ij}^S and x_{ij}^R). With integer solution to LP_{st}, we can easily determine values of y-variables, e.g., a positive Q_i^S implies $y_i^S = 1$.

Let $G^* = (N, A)$ be a directed network associated with the blocks, which is defined as node set $N = \{0, 1, 2, \ldots, T\}$ (node 0 means the beginning of period 1), and arc set $A = \{(i,j) : i, j \in N, i \leq j\}$. Arc (i, j) represents block $(i+1, j)$. The cost d_{ij} associated with arc (i, j) is equal to the block cost $c_{i+1,j}$. Fig. 2 gives an example of the auxiliary network that is created from a three-period DLRR.

Lemma 2. *Given that G^* is the directed network associated with the block data, the shortest path from 0 to T in G^* produces a best production schedule, i.e., a chain of blocks with smallest cost.*

Proof. Consider a path of $0 \to i_1 \to i_2 \cdots \to i_j \to i_{j+1} \to \cdots i_n \to T$. This corresponds to a chain of blocks $(1, i_1)$, (i_1+1, i_2), $\cdots$, (i_j+1, i_{j+1}), $\cdots$, (i_n+1, T). Following our previous definition, this block chain produces a feasible production schedule where the set-ups for manufacturing or remanufacturing are determined while computing each block cost. For instance, path $0 \to 2 \to 3$ in Fig. 2 represents a production schedule composed of blocks $(1, 2)$ and $(3, 3)$. The cost of the path can be computed by adding costs of those blocks included. This indeed determines the total set-up cost plus holding cost of inventory for this production schedule. Obviously, the shortest path defines a best production schedule for given block data. This completes our proof. □

With Lemma 2 and cost of each block, function BuildSolution (block data) in Algorithm 2 is to produce the initial solution σ.

4.3.3. *GenerateAdmissibleNeighbor* (σ, P)

This function uses operators to explore the neighborhood $N(\sigma)$ of the current solution σ and determine the subset $A(\sigma)$ of neighboring solutions. The neighborhood $N(\sigma)$ is a set of neighboring solutions in the search space, which is formally defined as

$N(\sigma) = \{\text{solutions } \sigma' \text{ obtained by applying neighboring operators to } \sigma\}.$

Following the above solution representation, we consider four neighboring operators, which involves shifting binary variables for set-ups of manufacturing and remanufacturing.

(1) One-shift operator. This neighboring operator aims at shifting set-up for manufacturing or manufacturing in one period. This neighborhood structure is explicitly defined as

$$N_1(\sigma) = \{\sigma' : \forall i \in \overline{T}, y_i^S(\sigma') = 1 - y_i^S(\sigma), \\ y_k^S(\sigma') = y_k^S(\sigma) \quad \forall k \in \overline{T} \backslash \{i\}\} \bigcup \{\sigma' : \forall i \in \overline{T}, \\ y_i^R(\sigma') = 1 - y_i^R(\sigma), y_k^R(\sigma') = y_k^R(\sigma) \quad \forall k \in \overline{T} \backslash \{i\}\}$$

(2) Two-shift operator. In each iteration, this operator simultaneously shifts set-ups in two periods for manufacturing or remanufacturing. This neighborhood structure is formally defined as

$$N_2(\sigma) = \{\sigma' : \forall i \neq j \in \overline{T}, y_i^S(\sigma') = 1 - y_i^S(\sigma), y_j^S(\sigma') = 1 - y_j^S(\sigma), \\ y_k^S(\sigma') = y_k^S(\sigma) \quad \forall k \in \overline{T} \backslash \{i, j\}\} \bigcup \{\sigma' : \forall i \neq j \in \overline{T}, y_i^R(\sigma') = 1 - y_i^R(\sigma), \\ y_j^R(\sigma') = 1 - y_j^R(\sigma), y_k^R(\sigma') = y_k^R(\sigma) \quad \forall k \in \overline{T} \backslash \{i, j\}\}$$

(3) Exchange operator. If in some period, only manufacturing or remanufacturing occurs, this operator exchanges set-ups for manufacturing and remanufacturing. This neighborhood structure can be formally defined as

$$N_3(\sigma) = \{\sigma' : \forall i \in \overline{T} \ni y_i^S(\sigma) \neq y_i^R(\sigma), y_i^S(\sigma') = y_i^R(\sigma), y_i^R(\sigma') = y_i^S(\sigma), \\ y_k^S(\sigma') = y_k^S(\sigma), y_k^R(\sigma') = y_k^R(\sigma) \quad \forall k \in \overline{T} \backslash \{i\}\}$$

(4) Cross two-shift operator. In each iteration, this operator respectively shifts the set-ups for manufacturing and remanufacturing, which may be in the same period, or different periods. We can formally define this neighborhood structure as

$$N_4(\sigma) = \{\sigma' : \forall i, j \in \overline{T}, y_i^S(\sigma') = 1 - y_i^S(\sigma), y_j^R(\sigma') = 1 - y_j^R(\sigma), \\ y_k^S(\sigma') = y_k^S(\sigma), y_k^R(\sigma') = y_k^R(\sigma) \quad \forall k \in \overline{T} \backslash \{i, j\}\}$$

With these four neighboring operators, we can explicitly define the neighborhood $N(\sigma)$ as

$$N(\sigma) = N_1(\sigma) \bigcup N_2(\sigma) \bigcup N_3(\sigma) \bigcup N_4(\sigma).$$

It is easy to check that the neighborhood $N(\sigma)$ consists of at most $(2T^2 + T)$ neighboring solutions. To obtain a trade-off between the solution quality and computational time, we allow

Author's personal copy

the TS-DLRR to explore all these solutions in $N(\sigma)$ since the size of neighborhood $N(\sigma)$ is not too large.

4.3.4. Tabus and tabu list

Tabus are one of the distinctive elements of TS when compared to other LS methods. Tabus are used to prevent cycling when moving away from local optima through non-improving moves. To achieve this, we declare tabu (disallowing) moves that reverse the effect of recent moves. Tabus are also useful to help the search move away from previously visited areas of the search space and this perform more extensive exploration in the search space. Tabus are stored in the tabu list, which is a short-term memory of the search.

In our application, the definitions of tabus depend on what neighboring operators are used. We use multiple tabu lists simultaneously and keep a separate tabu list for each type of neighboring operators. For one-shift move, we declare tabu shifting the set-ups back to previous status for θ iterations. θ is called the tabu tenure of a move. We store tabus of moves in a tabu list $T_1=(t^1_{i,j})$, where $i\in\overline{T}$ and $j=1$, 2, 3, and 4. Quantities $t^1_{i,1}$ and $t^1_{i,2}$ respectively record the status of shifting manufacturing back to non-manufacturing, and non-manufacturing to manufacturing in period i. $t^1_{i,3}$ and $t^1_{i,4}$ records the tabus of shifting remanufacturing. For two-shift move, we declare tabu changing set-up status of manufacturing in two periods or remanufacturing in two periods. Similarly, we also store tabus of moves in a tabu list $T_2=(t^2_{i,j})$ where $i\in\overline{T}$ and $j=1$, 2, 3, and 4. For exchange operator, we declare tabu simultaneously changing set-up status of both manufacturing and remanufacturing in one period. Accordingly, we define a tabu list $T_3=(t^3_{i,j})$ where $i\in\overline{T}$ and $j=1,2,3$, and 4. For cross two-shift operator, the tabu is defined as simultaneously changing set-ups status of one manufacturing and one remanufacturing back to previous status. The tabus of moves are stored in a tabu list $T_4=(t^4_{i,j})$ where $i\in\overline{T}$ and $j=1$, 2, 3 and 4.

Each tabu list is initialized to a list of all 0 s. In each search step, each element in the tabu list is updated by subtracting 1, and then the current move is declared tabu. In our implementation, we consider a random tabu tenure, i.e., randomly generate the tabu tenure of each move within interval $[\underline{\theta},\underline{\theta}+\Delta\theta]$ (the values of $\underline{\theta}$ and $\Delta\theta$ are determined by parameter calibration in Section 5.2). This random tabu scheme has been proven to be useful in metaheuristics for combinatorial optimization problems [26,29,50].

4.3.5. Aspiration criterion

Tabus are sometimes too restrictive and prohibit attractive moves, or they may lead to an overall stagnation of the search process. We use a so-called aspiration criterion, which will allow us to revoke (cancel) tabu status of some moves. We consider the simplest and most widely used aspiration criterion that allows a move, even if it is tabu, if it results in a solution σ with an objective value better than that of the current best-known solution σ^b.

4.3.6. SelectBestSolution($A(\sigma)$)

After all the candidate solutions are generated and evaluated, the function SelectBestSolution($A(\sigma)$) is then implemented to choose a non-tabu with the best objective value, or a tabu solution but satisfying the aspiration criterion. This best admissible solution is then used to update the tabu list and becomes a new solution for the next iteration. In addition, we might update the current best-known solution if this selected solution has a better objective value.

4.3.7. Diversification

The TS-DLRR approach likes other LS methods is easy to stagnate in a local optimal area: they tend to spend most, if not all, of their time in a restricted region of the search space. Here, we consider a diversification strategy to force the search into previously unexplored areas of the search space. That is a restart diversification strategy: if no solution improvement is found within t_{no} iterations, then we restart the search.

4.3.8. Termination condition

We terminate the TS-DLRR approach after a fixed number of iterations, i.e., 250 iterations in our implementation. Note that in complex termination schemes, the search is usually stopped after completing a sequence of phases [27].

5. Computational evaluation

In this section, we test the proposed TS-DLRR approach on a wide range of instances. The TS-DLRR approach was coded in the C language and complied on VC++ 2005. The LP solver in CPLEX 12.1 was called through CPLEX Callable Library C API 12.1 in our TS approach. The TS-DLRR approach was implemented on a PC with Intel CPU 2.2 GHz running Windows 7. In all the experiments, the computational time is reported in seconds.

5.1. Test instances

We attempt to compare our TS approach with other existing algorithms in the literature. These algorithms are all heuristics listed as follows:

- TBH: the adapted Silver-Meal heuristic proposed in [20].
- SCH: the Silver-Meal heuristic proposed in [47].
- TBH1: the TBH method with two improvement procedures proposed in [47].
- SCH1: the SCH method with two improvement procedures proposed in [47].

Schulz [47] has coded the above four methods and tested their performance on a set of 6480 benchmark problems. For the purpose of fair comparisons, we also test our proposed TS approach on the same instances and compare it with other alternative algorithms. Schulz [47] generated the benchmark problems as follows:

The instances have a 12-period planning horizon. The set-up costs for both manufacturing new products and remanufacturing returned products can be 200, 500, and 2000. The holding cost h^S for serviceables is set to 1. The holding cost h^R for carrying recoverable product can be 0.2, 0.5, and 0.8. The demand D_i in each period was randomly generated according to a normal distribution with a mean of ten units. The return R_i was also randomly generated according to a normal distribution with a mean of 30, 50, and 70. For these two normal distributions, instances were divided to small-variance case (the variance is set to 10%) and large-variance case (the variance is set to 20%). For each demand and return setting, 20 instances were randomly generated. Henceforth, we call these benchmark instances as Schulz's instances.

In addition to these benchmark instances, we also randomly generated six sets of 60 larger instances on which we tested the TS-DLRR and compared it with CPLEX. Each set includes ten instances. The planning horizon T ranges from 25 to 50 with a step of five. In all the larger instances, both demand D_i and return R_i are integers randomly generated from interval [10, 100], and set-up costs K^S and K^R are integers randomly generated from interval [200, 2000]. The holding cost h^S is set to 1, and h^R is randomly chosen from $\{0.2, 0.5, 0.8\}$.

Author's personal copy

5.2. Parameter calibration

As previously stated, our proposed TS approach has some parameters, which affect the algorithmic performance. We consider an automatic procedure, *F-Race*, to determine the best configurations of some main parameters. Birattari [51] gives the formal definition of parameter tuning and proposes a *F-Race* approach to solve the tuning problem. The *F-Race* approach is developed based on Friedman two-way analysis of variance by ranks, a statistical method for hypothesis testing. It can return the best configuration of a metaheuristic that performs statistically best with respect to some evaluation metric on a number of instances. For the detailed discussion of the *F-Race* approach, interested reader may refer to [51].

Table 1 gives the levels of main parameters of the TS-DLRR, which are tuned in this calibration study. In total, there are $5 \times 4 \times 5 = 100$ candidate configurations. We randomly generated 80 instances for parameter tuning. These tuning instances have similar properties to those benchmark instances in Section 5.1. The TS-DLRR approach with each candidate configuration was implemented to solve each of the 80 tuning instances for 8 seconds. The corresponding objective value of best-found solution is stored in a two-dimensional 80×100 array. With results under each configuration, we then implement the *F-Race* approach to tune the proposed TS approach. The best configuration is returned as $\underline{\theta} = 1$, $\Delta\theta = 9$, and $t_{no} = 15$.

Table 1
The set of candidate configurations.

$\underline{\theta}$	$\Delta\theta$	t_{no}
1	3	1
2	5	5
4	7	10
5	9	15
6		20

5.3. Performance of formulation LP_{st}

In Section 4.3, we propose an LP formulation to compute the block cost. As we point out, this formulation does not necessarily lead to integer values of binary variables for manufacturing and remanufacturing set-ups in each block. We first evaluate the performance of formulation LP_{st}. For all Schulz's instances, CPLEX 12.1 was implemented to solve formulation LP_{st} associated with each block (s,t). If formulation LP_{st} returns integer values for all y-variables associated with block (s,t), we call this block feasible, and infeasible, otherwise. For each instance, we compute the percentage of "feasible blocks" as $100 \times FB/[T(T+1)/2]$ where FB is the number of feasible blocks in each instance. Table 2 summarizes the percentage of feasible blocks, for which formulation LP_{st} returned integer values of y-variables. For each instance group, we consider three performance statistics: "Avg." the average percentage of feasible blocks in each instance group, "Std." is the standard deviation of percentage of feasible blocks, and "Max" is the maximum percentage of feasible blocks in each instance group.

The results in Table 2 show that, on average, formulation LP_{st} returned feasible set-ups of manufacturing and remanufacturing in approximately 67% of total 505 440 $(= 6480 \times T \times (T+1)/2 = 6480 \times 12 \times 6)$ blocks. The performance is slightly different among these instance groups. This formulation LP_{st} performed best in instances group with small return ratio, e.g., more than 87% of all blocks have integer values of y-variables. We also see that the performance of formulation LP_{st} is not good in instances group with large return ratio. In other instance groups, formulation LP_{st} has a similar and stable performance.

Table 2
Performance of formulation LP_{st} (in %) on Schulz's instances.

Item	Percentage of feasible blocks		
	Avg.	Std.	Max
All instances	66.54	16.36	94.87
Demand			
Small variance	65.95	17.22	93.59
Large variance	67.12	15.44	94.87
Returns			
Small variance	65.90	17.00	94.87
Large variance	67.17	15.68	93.59
Return ratio			
30%	87.23	3.69	94.87
50%	62.93	4.80	76.92
70%	49.45	5.78	73.08
K^S			
200	66.55	16.37	94.87
500	66.55	16.40	94.87
2000	66.50	16.33	94.87
K^R			
200	66.53	16.36	94.87
500	66.53	16.37	94.87
2000	66.54	16.37	94.87
h^R			
0.2	66.58	16.37	94.87
0.5	66.52	16.37	94.87
0.8	66.51	16.36	94.87

Table 3
Performance of the block-chain based method.

Set	K^R	K^S	h^R	ELSP based method		Block-chain based method	
				GAP	CPU	GAP	CPU
1	200	200	0.20	44.52	0.21	5.57	3.14
2	200	200	0.50	40.51	0.26	4.12	2.85
3	200	200	0.80	38.22	0.21	3.70	3.12
4	200	500	0.20	35.00	0.28	3.14	2.81
5	200	500	0.50	32.50	0.27	3.41	2.88
6	200	500	0.80	30.98	0.34	3.55	3.34
7	200	2000	0.20	22.55	0.15	0.25	2.53
8	200	2000	0.50	21.55	0.18	0.42	2.93
9	200	2000	0.80	20.47	0.18	0.66	2.78
10	500	200	0.20	67.94	0.23	21.08	2.79
11	500	200	0.50	62.74	0.24	16.87	2.67
12	500	200	0.80	58.77	0.21	15.41	3.16
13	500	500	0.20	57.25	0.26	9.56	2.77
14	500	500	0.50	53.85	0.25	8.78	3.22
15	500	500	0.80	51.11	0.31	8.58	3.59
16	500	2000	0.20	41.83	0.18	1.66	2.94
17	500	2000	0.50	40.30	0.32	1.53	3.93
18	500	2000	0.80	39.06	0.28	2.30	2.99
19	2000	200	0.20	90.74	0.19	64.21	3.07
20	2000	200	0.50	87.64	0.22	54.79	2.93
21	2000	200	0.80	84.55	0.24	46.29	3.16
22	2000	500	0.20	85.77	0.22	50.23	2.85
23	2000	500	0.50	82.83	0.20	42.59	2.89
24	2000	500	0.80	79.99	0.23	35.75	2.85
25	2000	2000	0.20	74.05	0.16	26.21	2.86
26	2000	2000	0.50	71.95	0.17	22.16	2.86
27	2000	2000	0.80	69.90	0.18	18.57	2.86
Average				55.06	0.23	17.46	2.99

Author's personal copy

Table 4
Detailed statistics of the block-chain based method.

Item	K^R			K^S			h^R		
	200	500	2000	200	500	2000	0.2	0.5	0.8
Average GAP	2.76	9.53	40.09	25.78	18.40	8.20	20.21	17.19	14.98

5.4. Performance of the block-chain based method

As previously stated, we use a block-chain based method to generate an initial solution in the TS-DLRR approach. In fact, this method can also be used as a heuristic for the DLRR. We evaluate the performance of this new method by comparing it with the economic lot scheduling problem (ELSP) based solution. In this

Table 5
Comparison with state-of-the-art algorithms in terms of percentage cost error.

Item	TBH (%)			TBH1 (%)			SCH (%)		
	Avg.	Std.	Max	Avg.	Std.	Max	Avg.	Std.	Max
All instances	7.5	7.9	49.2	6.9	7.9	49.2	6.1	7.6	47.3
Demand									
Small variance	7.2	7.9	43.6	6.6	7.9	43.5	6.0	7.6	47.3
Large variance	7.8	8.0	49.2	7.2	8.0	49.2	6.1	7.5	43.9
Returns									
Small variance	7.3	7.8	47.2	6.8	7.8	47.2	6.1	7.6	47.3
Large variance	7.7	8.0	49.2	7.1	8.0	49.2	6.1	7.5	46.3
Return ratio									
30%	5.5	5.5	31.3	4.9	5.4	31.3	3.7	4.5	28.5
50%	8.5	9.4	40.1	8.0	9.3	39.8	7.3	8.2	41.8
70%	8.4	8.0	49.2	8.0	8.0	49.2	7.2	8.7	47.3
K^S									
200	4.3	4.5	20.2	3.5	4.0	20.2	3.4	3.6	17.6
500	5.4	5.2	25.1	4.8	4.9	23.7	3.9	3.9	19.3
2000	12.8	9.9	49.2	12.6	9.9	49.2	10.9	10.4	47.3
K^R									
200	10.9	9.1	49.2	10.0	9.4	49.2	6.6	7.8	40.2
500	7.9	6.6	34.7	7.3	6.6	34.7	8.1	8.2	47.3
2000	3.7	6.0	29.4	3.6	5.9	29.4	3.5	5.7	25.7
h^R									
0.2	5.9	8.0	42.9	5.8	8.0	42.9	5.3	8.0	47.3
0.5	7.5	7.7	49.2	7.0	7.7	49.2	6.5	7.6	42.4
0.8	9.1	7.7	44.4	8.1	7.8	44.4	6.3	7.0	40.3

Item	SCH1 (%)			TS-DLRR (%)			
	Avg.	Std.	Max	Avg.	Std.	Max	% OPT
All instances	2.2	2.9	24.3	0.00082	0.00508	0.09338	96.60
Demand							
Small variance	2.1	2.8	18.9	0.00089	0.00521	0.06367	96.17
Large variance	2.4	3.0	24.3	0.00075	0.00494	0.09338	97.04
Returns							
Small variance	2.2	2.9	21.1	0.00086	0.00506	0.05574	98.39
Large variance	2.3	2.9	24.3	0.00078	0.00509	0.09338	96.82
Return ratio							
30%	1.2	1.8	12.1	0.00070	0.00486	0.05047	97.31
50%	2.3	2.7	16.2	0.00067	0.00424	0.04926	96.94
70%	3.3	3.5	24.3	0.00108	0.00598	0.09338	95.56
K^S							
200	2.3	2.6	13.5	0.00159	0.00746	0.09338	94.26
500	2.1	2.5	12.8	0.00081	0.00440	0.05023	95.93
2000	2.3	3.4	24.3	0.00007	0.00113	0.03232	99.63
K^R							
200	1.9	2.1	11.8	0.00151	0.00740	0.09338	95.05
500	3.4	3.2	19.1	0.00076	0.00439	0.03908	96.53
2000	1.4	2.9	24.3	0.00019	0.00157	0.02420	98.24
h^R							
0.2	1.7	2.5	21.1	0.00064	0.00484	0.09338	96.99
0.5	2.3	3.0	24.3	0.00081	0.00488	0.04532	96.71
0.8	2.8	3.0	20.6	0.00102	0.00547	0.06367	96.11

Author's personal copy

ELSP based solution, returns in each period are first remanufactured to meet the demands. For each instance, we first create an MIP formulation for this ELSP where we remanufacture all returns in each period, and then use CPLEX 12.1 to solve this MIP formulation to get an ELSP based solution. Henceforth, we refer to this method as the ELSP based method. Note that for some test instances, the ELSP formulation might not return a feasible integer solution to the original DLRR. Such instances were not included in this comparison study.

These two methods were implemented to solve 27 sets of 1350 Schulz's instances in total. The comparison results are summarized in Table 3. For 50 instances in each group, we use two performance measures to evaluate each approach: "GAP" denotes the average relative gap between the objective values of the optimal solution (OPT) and best integer solution (UB) returned by each approach. For each instance, the "GAP" is computed as (UB-OPT)/UB $\times$ 100.

The results in Table 3 clearly demonstrate that the block-chain based method significantly outperforms the ELSP based method. On these instances with different properties, the block-chain based method has returned much less GAP values. Over all these 1350 instances, the ELSP based method reached an average GAP of 55.06%, whereas our block-chain based method got a GAP of 17.46% (about one third). For some instances (e.g., K^R=200, K^S=2000, and h^R=0.20), the block-chain based method returned very good initial solution with average GAP less than 0.3%. In addition, both methods performed so fast to get integer solutions. From the results in Table 3, we can further observe that the performance of the block-chain method depends on the parameters of set-up cost and inventory holding cost. Table 4 summarizes the detailed statistics of the block-chain method in terms of different parameters. The results in Table 4 show that the block-chain based method has a best and worst performance in the cases with K^R=200 and K^R=2000, respectively. The performance evolution patterns change with K^R, K^S, and h^R. For example, the block-chain method has got an increasing performance as K^R decreases from 2000 to 200. However, the average GAP is monotonically decreasing in both K^S and h^R.

5.5. Comparison with state-of-the-art algorithms

We are now ready to compare the performance of our TS-DLRR algorithm with other alternatives from the literature. For each method, we measure the performance by the percentage cost error, which is defined as the percentage gap between the total cost and an optimal solution. For each benchmark instance, the percentage cost error is computed as (UB-OPT)/UB $\times$ 100 where UB is the objective value of best solution returned by the TS-DLRR. We used CPLEX 12.1 to get objective value (OPT) of an optimal solution to each benchmark instance.

The comparison results are summarized in Table 5. For each instance group, we use three performance measures to evaluate each approach: "Avg." is the average percentage cost error for all the instances in each group, "Std." is the standard deviation of errors, and "Max" represents the maximum error in each instance group. In addition, "%OPT" is the percentage of instances to which the TS-DLRR approach has produced an optimal solution.

The results in Table 5 clearly demonstrate that our TS-DLRR approach outperforms other four alternative algorithms from the literature. Over all these 6480 instances, the TS-DLRR approach has achieved an average cost error of 0.00082%, which is much less than that for TBH (7.5%), TBH1 (6.9%), SCH (6.1%), and SCH1 (2.2%). Our proposed algorithm has produced an optimal solution in 96.60% of benchmark problems. Furthermore, the TS-DLRR approach has generated different evolution patterns in different instance groups. Both K^S and K^R have same influence on the performance of the TS-DLRR approach. For example, the average cost errors decrease in both K^S and K^R. However, parameter h^R has a different effect, e.g., as h^R increases, the average cost errors increase.

Table 6
Computational times of the TS-DLRR.

Item	Computational times		
	Avg.	Std.	Max
All instances	53.68	71.26	490.38
Demand			
Small variance	58.68	76.78	490.38
Large variance	48.69	64.90	484.33
Returns			
Small variance	54.48	72.87	490.38
Large variance	52.80	69.41	484.33
Return ratio			
30%	56.51	81.13	490.38
50%	51.75	65.51	475.93
70%	52.80	65.96	484.33
K^S			
200	54.79	76.83	484.33
500	65.12	77.81	490.38
2000	41.14	54.69	413.68
K^R			
200	65.38	73.01	478.91
500	61.65	75.74	490.38
2000	34.02	59.85	484.33
h^R			
0.2	49.50	66.77	475.93
0.5	55.18	73.39	484.33
0.8	56.37	73.25	490.38

The computational times of the TS-DLRR approach are reported in Table 6. The results in Table 6 show that on average, the TS-DLRR approach is an efficient algorithm, which can solve all these instances within average 54 s. It took only average 34.02 s to solve 2160 instances with K^R=2000. Seemingly, it is relatively easier to solve instances with larger K^R. This is also indicated by the cost errors in Table 5.

5.6. Results on larger instances

Next, we examine how the TS-DLRR approach performs on larger instances. The TS-DLRR was run to solve 90 instances. We terminated TS-DLRR after 70 iterations and set the neighborhood size to 1000. To carry out a comparison, CPLEX was implemented to optimally solve each instance. Table 7 summarizes the computational results. As previously stated, "UB" means the objective value of the best solution returned by the TS-DLRR approach. "GAP" represents the average relative gap between the objective value of optimal solution (OPT) and the UB. "CPU" represents the computational time required by the approach to find the best integer solution. The computational results in Table 7 have obviously demonstrated the NP-hardness of the DLRR. With increase of instance size, the problem becomes very hard to solve. For example, CPLEX required 27 197 s to solve instance 4 in set 6. On average, it took 11 151.62 s for CPLEX to find optimal solutions to these ten instances with 50 planning periods. The results also demonstrate the performance of the TS-DLRR approach on larger instances. Within acceptable computation time, the TS-DLRR approach has solved these larger instances with a very small deviation from optimality. For these ten 50-period instances, the proposed approach reached an average percentage cost error of 0.8860% within around 742 s.

Author's personal copy

Table 7
Results on larger instances.

T	No.	CPLEX		TS-DLRR			T	No.	CPLEX		TS-DLRR		
		OPT	CPU	UB	CPU	GAP			OPT	CPU	UB	CPU	GAP
25	1	8615.5	0.46	8615.5	143.72	0	30	1	5929.6	2.76	5929.6	745.68	0
	2	8615.5	0.66	8615.5	452.70	0		2	12074	11.89	12074.0	457.02	0
	3	9036.8	0.64	9036.8	33.53	0		3	10496	18.17	10498.0	92.79	0.0191
	4	11394.5	0.51	11394.5	162.90	0		4	12016	5.97	12034.0	102.02	0.1496
	5	8660	0.47	8665.5	262.33	0.0635		5	13418.2	12.04	13510.8	126.17	0.6854
	6	6047.8	0.78	6051.6	260.82	0.0635		6	5531.5	45.14	5532.3	530.39	0.0137
	7	11871.5	0.48	11883.5	237.30	0.1010		7	12963	10.71	13047.8	224.96	0.6502
	8	9353.5	0.48	9365.5	25.12	0.1281		8	5385	10.33	5385.0	55.04	0
	9	11871.5	0.74	11873.1	184.61	0.0133		9	5385	39.66	5432.6	54.94	0.8768
	10	10586.2	0.47	10586.2	706.07	0		10	11232	47.38	11343.5	595.33	0.9833
Average			0.57		246.91	0.0369	Average			20.41		298.44	0.3378
35	1	18389	130.25	18389.0	1102.66	0	40	1	11759.6	7521.14	11841.0	389.41	0.6871
	2	14724.4	20.02	14724.4	348.12	0		2	12081.5	6913.21	12168.2	1096.00	0.7122
	3	14724.4	29.54	14724.4	348.02	0		3	15232.5	5553.34	15356.7	942.02	0.8088
	4	15753.6	28.25	15753.6	503.74	0		4	15232.5	5540.92	15516.4	448.79	1.8298
	5	12518.5	7.78	12549.5	254.86	0.2470		5	14135	5339.77	14149.4	492.27	0.1020
	6	17829.5	126.43	18015.5	346.00	1.0324		6	14135	5256.10	14149.4	116.12	0.1020
	7	18545	63.07	18746.5	310.96	1.0749		7	18219.2	1547.94	18427.7	858.84	1.1315
	8	13691	470.72	13844.0	946.84	1.1052		8	17196	1417.20	17396.9	925.78	1.1549
	9	10030.5	397.12	10158.5	656.53	1.2600		9	19755.8	1264.29	19916.0	914.98	0.8044
	10	12755.5	15.43	12897.9	866.15	1.1038		10	19755.8	1184.43	19978.8	443.97	1.1161
Average			128.86		568.39	0.5823	Average			4153.83		662.82	0.8449
45	1	11269.2	8142.56	11305.4	441.60	0.3202	50	1	16295.2	4592.90	16479.1	610.32	1.1158
	2	18038	1294.96	18187.0	720.39	0.8193		2	15634	26807.12	15646.5	1107.38	0.0800
	3	21727.4	2765.76	21952.9	1040.51	1.0272		3	25946.2	14805.43	26227.1	731.64	1.0711
	4	11763	3539.58	11903.2	491.03	1.1778		4	15634	27197.06	15811.2	569.26	1.1206
	5	11763	3613.92	11903.2	834.97	1.1778		5	21094.5	6223.83	21307.7	1022.77	1.0004
	6	17998	3680.24	18028.5	554.43	0.1691		6	22461.4	7641.33	22485.2	395.35	0.1058
	7	17998	3741.50	18249.6	549.54	1.3789		7	16807.4	6485.45	16979.1	628.90	1.0114
	8	24549	5464.85	24790.5	1189.05	0.9742		8	26821.2	5835.53	27097.8	974.52	1.0208
	9	22274.5	8765.82	22523.3	679.88	1.1048		9	24576	7958.32	24827.1	549.14	1.0114
	10	23811	5589.65	23902.3	541.88	0.3819		10	13093	3969.21	13268.6	822.86	1.3231
Average			4659.88		704.33	0.8531	Average			11151.62		741.21	0.8860

6. Conclusions

In today's circular economy, remanufacturing has become a promising and economic practice in the production and manufacturing. In this paper, we study the dynamic lot sizing problem with product returns and remanufacturing (DLRR). In the planning horizon, both manufacturing new products and remanufacturing returned products can be used to meet the demands. This lot sizing problem with general set-up cost functions for manufacturing and remanufacturing is an NP-hard problem. We develop a new tabu search metaheuristic to produce high-quality solution. This new algorithm includes some new features. To generate a high-quality initial solution, we present a block-chain based method, which splits the planning horizon into a chain of blocks. A block is a set of periods with either a string of manufacturing set-ups, a string of remanufacturing set-ups, or both. A new LP formulation is used to get the block cost. With cost of each block, we then find the best production schedule (best combination of blocks) by solving a shortest path problem. This block-chain based method can also be implemented as a heuristic for solving the DLRR. The neighboring operators are defined as shifting integer variables for set-ups of manufacturing and remanufacturing in the planning horizon. We carried out a computational study to evaluate the efficiency of the proposed tabu search metaheuristic. The computational results first show that the block-chain method can generate high-quality initial solutions, compared with the economic lot sizing based method. Over a set of 6480 benchmark instances, we experimentally compare the performance of the proposed TS method with other available methods from the literature and show that our TS approach outperforms other algorithms. Our proposed algorithm has produced an optimal solution in 96.60% of benchmark problems, with an average deviation of 0.00082% from optimality. We also demonstrated that the proposed approach was a robust method for larger DLRR instances. In summary, our proposed algorithms are applicable to the DLRR with a joint set-up cost as well as separate set-up costs for manufacturing and remanufacturing.

In the future work, an interesting research avenue is to study other metaheuristics and exact algorithms for the DLRR with general set-up cost function.

Acknowledgments

We would like to express our sincere appreciation to two anonymous referees and editors for their suggestions in improving this manuscript. The research of the first author has been partially supported by the Natural Science Foundation of China (Grant no. 71101105), NSFC major program (Grant no. 71090404/71090400), Shanghai Pujiang program, research fund for doctoral program of higher education of China (Grant no. 20110072120013), Shanghai philosophical and social science program (Grant no. 2011EGL004), and the Fundamental Research Funds for the Central Universities.

References

[1] Nasr N, Varel E. Lifecycle analysis and costing in an environmentally conscious manufacturing environment, In: APICS remanufacturing symposium proceedings, Dayton, OH; 1996. p. 44–7.

Author's personal copy

[2] Hwang SN, Chen C, Chen Y, Lee H-S, Shen P-D. Sustainable design performance evaluation with applications in the automobile industry: focusing on inefficiency by undesirable factors. Omega 2013;41:553–8.
[3] Sueyoshi T, Goto M. Returns to scale vs. damages to scale in data envelopment analysis: an impact of U.S. clean air act on coal-fired power plants. Omega 2013;41:164–75.
[4] Ijomah W. A model-based definition of the generic remanufacturing business process, PhD thesis, The University of Plymouth, UK; 2002.
[5] Ijomah WL, Mcmahon CA, Hammond GP, Newman ST. Development of robust design-for-remanufacturing guidelines to further the aims of sustainable development. International Journal of Production Research 2007;45:4513–36.
[6] Thierry M, Solomon M, van Nunen JAEE, van Wassenhove LN. Strategic issues in product recovery management. California Management Review 1995;37:114–35.
[7] Guide VDR, Teunter RH, Van Wassenhove LN. Matching demand and supply to maximize profits from remanufacturing. Manufacturing & Service Operations Management 2003;5:303–16.
[8] Ray S, Boyaci T, Aras N. Optimal prices and trade-in rebates for durable, remanufacturable products. Manufacturing & Service Operations Management 2005;7:208–28.
[9] Zhou SX, Tao Z, Chao X. Optimal control of inventory systems with multiple types of remanufacturable products. Manufacturing & Service Operations Management 2011;13:20–34.
[10] Lund R. Remanufacturing: the experience of the united states and implications for developing countries, Technical report, World Bank; 1984.
[11] Lund R. The remanufacturing industry: hidden giant, Technical report, Boston University; 1996.
[12] Guide VDR, Jayaraman V, Srivastava R. Production planning and control for remanufacturing: a state-of-the-art survey. Robotics and Computer-Integrated Manufacturing 1999;15:221–30.
[13] Guide VDR, Jayaraman V, Srivastava R, Benton WC. Supply-chain management for recoverable manufacturing systems. Interfaces 2000;30:125–42.
[14] Guide VDR, Van Wassenhove LN. Managing product returns for remanufacturing. Production and Operation Management 2001;10:142–55.
[15] Guide VDR, Van Wassenhove LN. The evolution of closed-loop supply chain research. Operations Research 2009;57:10–8.
[16] Blackburn JD, Guide VDRJ, Souza GC, Van Wassenhove LN. Reverse supply chains for commercial returns. California Management Review 2004;46:6–22.
[17] Srivastava SK. Green supply-chain management: a state-of-the-art literature review. International Journal of Management Reviews 2007;9:53–80.
[18] Barker TJ, Zabinsky ZB. A multicriteria decision making model for reverse logistics using analytical hierarchy process. Omega 2011;39:558–73.
[19] Wu CH. OEM product design in a price competition with remanufactured product. Omega 2013;41:287–98.
[20] Teunter RH, Bayindir ZP, Van Den Heuvel W. Dynamic lot sizing with product returns and remanufacturing. International Journal of Production Research 2006;44:4377–400.
[21] Kaminsky P, Swaminathan JM. Effective heuristics for capacitated production planning with multiperiod production and demand with forecast band refinement. Manufacturing & Service Operations Management 2004;6:184–94.
[22] Groër C, Golden B, Wasil E. The consistent vehicle routing problem. Manufacturing & Service Operations Management 2009;11:630–43.
[23] Brusco MJ, Steinley D. Neighborhood search heuristics for selecting hierarchically well-formulated subsets in polynomial regression. Naval Research Logistics 2010;57:33–44.
[24] Jans R, Degraeve Z. Meta-heuristics for dynamic lot sizing: a review and comparison of solution approaches. European Journal of Operational Research 2007;177:1855–75.
[25] Baki M, Chaouch A, Abdul-Kader W. A heuristic solution procedure for the dynamic lot size problem with remanufacturing and product recovery, Working paper, University of Windsor; 2012.
[26] Gendreau M, Laporte G, Seguin R. A tabu search heuristic for the vehicle routing problem with stochastic demands and customers. Operations Research 1996;44:469–77.
[27] Gendreau M. An introduction to tabu search. In: Glover F, Kochenberger G, editors. Handbook of metaheuristics. Boston, USA: Kluwer Academic Publishers; 2003. p. 37–54.
[28] Li X, Tian P, Leung SCH. An ant colony optimization metaheuristic hybridized with tabu search for open vehicle routing problems. Journal of the Operational Research Society 2009;60:1012–25.
[29] Li X, Tian P, Aneja YP. An adaptive memory programming metaheuristic for the heterogeneous fixed fleet vehicle routing problem. Transportation Research Part E 2010;46:1111–27.
[30] Robinson P, Narayanan A, Sahin F. Coordinated deterministic dynamic demand lot-sizing problem: a review of models and algorithms. Omega 2009;37:3–15.
[31] Gopalakrishnan M, Ding K, Bourjolly JM, Mohan S. A tabu-search heuristic for the capacitated lot-sizing problem with set-up carryover. Management Sciences 2001;47:851–63.
[32] Hung YF, Chen CP, Shih CC, Hung MH. Using tabu search with ranking candidate list to solve production planning problems with setups. Computers & Industrial Engineering 2003;45:615–34.
[33] Lim A, Rodrigues B, Xiao F, Zhu Y. Crane scheduling with spatial constraints. Naval Research Logistics 2004;51:386–406.
[34] Drezner Z, Salhi S. Using hybrid metaheuristics for the one-way and two-way network design problem. Naval Research Logistics 2002;49:449–63.
[35] Gendreau M, Laporte G, Parent I. Heuristics for the location of inspection stations on a network. Naval Research Logistics 2000;47:287–303.
[36] Silver E, Pyke D, Peterson R. Inventory management and production planning and scheduling. New York: Wiley; 1998.
[37] Brahimi N, Dauzere-Peres S. Single item lot sizing problems. European Journal of Operational Research 2006;168:1–16.
[38] Tang O, Teunter R. Economic lot scheduling problem with returns. Production and Operation Management 2006;15:488–97.
[39] Teunter R, Kaparis K, Tang O. Multi-product economic lot scheduling problem with separate production lines for manufacturing and remanufacturing. European Journal of Operational Research 2008;191:1241–53.
[40] Teunter R, Tang O, Kaparis K. Heuristics for the economic lot scheduling problem with returns. International Journal of Production Economics 2009;118:323–30.
[41] Richter K, Sombrutzki M. Remanufacturing planning for the reverse Wagner/Whitin models. European Journal of Operational Research 2000;121:304–15.
[42] Richter K, Weber J. The reverse Wagner/Whitin model with variable manufacturing and remanufacturing cost. International Journal of Production Economics 2001;71:447–56.
[43] Golany B, Yang J, Yu G. Economic lot-sizing with remanufacturing options. IIE Transactions 2001;33:995–1003.
[44] Yang J, Golany B, Yu G. A concave-cost production planning problem with remanufacturing options. Naval Research Logistics 2005;52:443–58.
[45] Beltran J, Krass D. Dynamic lots sizing with returning items and disposals. IIE Transactions 2002;34:437–48.
[46] Pang Z, Tang J, Liu O. Capacitated dynamic lot sizing problems in closed-loop supply chain. European Journal of Operational Research 2009;198:810–21.
[47] Schulz T. A new Silver-Meal based heuristic for the single-item dynamic lot sizing problem with returns and remanufacturing. International Journal of Production Research 2011;49:2519–33.
[48] Glover F. Future paths for integer programming and links to artificial intelligence. Computers & Operations Research 1986;13:533–49.
[49] Glover F, Laguna M. Tabu search. Boston: Kluwer Academic Publishers; 1997.
[50] Li X, Aneja YP. A branch-and-cut approach for the minimum-energy broadcasting problem in wireless networks. INFORMS Journal on Computing 2012;24:443–56.
[51] Birattari M. The problem of tuning metaheuristics as seen from a machine learning perspective, PhD thesis, Universite Libre De Bruxelles; 2005.