Computers & Industrial Engineering 83 (2015) 307–315

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

Metaheuristic optimization for the Single-Item Dynamic Lot Sizing problem with returns and remanufacturing

K.E. Parsopoulos [a], I. Konstantaras [b,*], K. Skouri [c]

[a] Department of Computer Science & Engineering, University of Ioannina, GR-45110 Ioannina, Greece
[b] Department of Business Administration, University of Macedonia, GR-54636 Thessaloniki, Greece
[c] Department of Mathematics, University of Ioannina, GR-45110 Ioannina, Greece

A R T I C L E I N F O

Article history:
Received 30 July 2014
Received in revised form 1 December 2014
Accepted 19 February 2015
Available online 9 March 2015

Keywords:
Lot Sizing
Inventory optimization
Remanufacturing
Differential Evolution
Metaheuristics

A B S T R A C T

The use of metaheuristics for solving the Single-Item Dynamic Lot Sizing problem with returns and remanufacturing has increasingly gained research interest. Recently, preliminary experiments with Particle Swarm Optimization revealed that population-based algorithms can be competitive with existing state-of-the-art approaches. In the current work, we thoroughly investigate the performance of a very popular population-based algorithm, namely Differential Evolution (DE), on the specific problem. The most promising variant of the algorithm is experimentally identified and properly modified to further enhance its performance. Also, necessary modifications in the formulation of the corresponding optimization problem are introduced. The algorithm is applied on an abundant test suite employed in previous studies. Its performance is analyzed and compared with a state-of-the-art approach as well as with a previously investigated metaheuristic algorithm. The results suggest that specific DE variants can be placed among the most efficient approaches, thereby enriching the available algorithmic artillery for tackling the specific type of problems.

1. Introduction

The field of *Reverse Logistics* contains all logistics processes beginning with the take-back of used products from customers up to the stage of making them reusable products or their disposal. Reverse Logistics activities have received increasing attention within Logistics and Operations Management over the last years both from theoretical and practical point of view. One reason for this is the more rigid environmental legislation and the growing environmental concerns.

In most countries environmental regulations are in place, rendering manufacturers responsible for the whole life cycle of their product. A common example of these regulations is the take-back obligations after usage (Fleischmann et al., 1997). Another reason is the economic benefits of reusing products rather than disposing them. Reverse Logistics can bring direct gains to companies by dwindling on the use of raw materials, adding value with recovery, as well as reducing disposal costs, which have significantly increased in recent years due to depletion of incineration and land filling capacities. Environmental regulations, “green image” policies due to growing environmental concerns, as well as the potential economical benefits of product recovery, have pushed manufacturers to integrate product recovery management with their manufacturing process. Two very good recent review papers on Reverse Logistics supply chain management are Govindan, Soleimani, and Kannan (2015) and Stindt and Sahamie (2014).

Recovery processes are generally classified into the following five types: repair, refurbishing, remanufacturing, cannibalization, and recycling. *Remanufacturing*, which is the topic of the present work, is the process that brings used products up to quality standards that are as rigorous as those of new products. A remanufactured product is a returned product that a manufacturer puts through its manufacturing process (or remanufactures) in order to restore it to a good-as-new condition. It shall be distinguished from refurbished products, which are returned products that are tested and usually have some parts replaced if the manufacturer deems this necessary to restore the product to working condition. Remanufacturing is a typical example for economically attractive reuse activities, since it transforms used products into like-new products.

After disassembling the returned product, modules and parts are extensively inspected and problematic parts are repaired or, if not possible, replaced with new parts. These operations allow a considerable amount of value incorporated in the used product to be

* Corresponding author.
E-mail addresses: kostasp@cs.uoi.gr (K.E. Parsopoulos), ikonst@uom.gr (I. Konstantaras), kskouri@uoi.gr (K. Skouri).

http://dx.doi.org/10.1016/j.cie.2015.02.014
0360-8352/© 2015 Elsevier Ltd. All rights reserved.

regained. Remanufactured products have usually the same quality as the new products and are sold for the same price but they are less costly. A recent review paper in pricing of new and remanufactured products and production planning is given in Steeneck and Sarin (2013). Also, some manufacturers offer the same warranty and service options on remanufactured products as they do on new ones. Typical examples of remanufacturable products include mostly high-value components such as aircraft or automobile engines, aviation equipment, medical equipment, office furniture, machine tools, copiers, computers, electronics equipment, toner cartridges, cellular telephones, and single-use cameras (Fleischmann et al., 1997; Guide, Jayaraman, & Srivastava, 1999; Thierry, Salomon, van Numen, & van Wassenhove, 1995).

Inventory management and control is one of the key decision-making areas while managing product returns. *Dynamic* or *Economic Lot Sizing* (ELS), i.e., determining production orders over a number of future periods in which demand is dynamic and deterministic, is one of the most extensively researched topics in inventory control. However, the ELS problem with remanufacturing options (ELSR), as an alternative for manufacturing, has received quite a bit of attention in the Reverse Logistics literature. A very good recent review paper that offers a general overview of the existing quantitative models for the ELSR problem is Akcali and Cetinkaya (2011).

In the ELSR problem, known quantities of used products are returned from customers in each period over a finite planning horizon. There is no demand for these returned products themselves, but they can be remanufactured such that they become as good as new. Customer demand can then be fulfilled from two sources, namely newly manufactured and remanufactured items. Since both can be used to serve customers, they are referred to as *serviceables* and so the retailer maintains separate inventories for serviceables and returned used product. When ordering a newly manufactured or remanufactured product, the retailer incurs a fixed setup cost. In addition, in each period the retailer incurs holding costs for storing serviceables and returned product in inventory. Thus, in ELSR problem the traditional trade-off between set-up and holding costs is extended with remanufacturing set-up cost and holding cost for returns.

Up-to-date, different variants of the ELSR problem have been studied. In Richter and Sombrutzki (2000), the classical Wagner–Whitin model (Wagner & Whitin, 1958) is extended by introducing a remanufacturing process. It was shown that there exists an optimal solution that is a zero-inventory policy. Also, a dynamic programming algorithm to determine the periods where products are manufactured and remanufactured, was proposed. Richter and Weber (2001) extended the previous models by introducing variable manufacturing and remanufacturing costs and proved the optimality of a policy starting with remanufacturing before switching to manufacturing.

A variant of ELSR with disposal of returned used products at a cost was considered in Golany, Yang, and Yu (2001), and it was shown that this problem is NP-complete under general concave production and holding costs. The same setting was studied in Yang, Golany, and Yu (2005), where a polynomial-time heuristic was developed to solve the problem. Pineyro and Viera (2009) proposed and evaluated a set of inventory policies designed for the ELSR problem, under the assumption that remanufacturing used items is more suitable than disposing of them and producing new items. A Tabu Search approach was proposed, aiming at finding a near-optimal solution. Teunter, Bayindir, and Van den Heuvel (2006) studied ELSR with separate setup and joint setup for manufacturing and remanufacturing. For the case of joint setup cost, they provided an exact polynomial-time dynamic programming algorithm. They also studied and compared the computational performance of modified versions of three well-known heuristics, namely *Silver-Meal* (SM), Least Unit Cost, and Part Period Balancing, for the separate and joint setup cost cases. Helmrich, Jans, Den Heuvel, and Wagelmans (2014) showed that both models studied in Teunter et al. (2006), with separate and joint setup costs, are NP-hard problems and also they proposed and compared several alternative mixed-integer programming formulations of both problems. Ahiska and Kurtul (2014) studied an inventory control problem for a periodic review stochastic hybrid manufacturing/remanufacturing system with two products and substitution.

The multi-product economic lot scheduling problems with returns, in the case of separate production lines for manufacturing and remanufacturing, was studied in Teunter, Kaparis, and Tang (2008). The authors proposed a mixed integer programming model to solve the problem for a fixed cycle time, which can be combined with a cycle time search to find an optimal solution. In Teunter, Tang, and Kaparis (2009) the ELSR problem was considered with two sources of production: manufacturing of new items and remanufacturing of returned items. For both cases, a mixed integer programming formulation was presented for a fixed cycle time, and simple heuristics were proposed for the determination of the optimal solution.

In Zanoni, Segerstedt, and Tang (2012) the multi-product ELSR problem was further analyzed, extending the scheduling policy from the common cycle to a basic period policy. A simpler scheduling policy was introduced, which can be solved with near-optimal solutions and has the potential to improve the cost performance in the system. Schulz (2011) proposed a generalization of the SM-based heuristic introduced in Teunter et al. (2006) for the separate setup cost case. The enhanced SM variants exhibited significantly better performance in terms of the average percentage error from the optimal solution.

Recently, both trajectory-based and population-based metaheuristics were used to tackle the ELSR problem. A Tabu Search (TS) algorithm was proposed in Li, Baki, Tian, and Chaouch (2013), while the Particle Swarm Optimization (PSO) algorithm was investigated in Moustaki, Parsopoulos, Konstantaras, Skouri, and Ganas (2013). In addition, recent works on the Wagner–Whitin and relevant inventory optimization problems (Piperagkas, Voglis, Tatsis, Parsopoulos, & Skouri, 2011; Piperagkas, Konstantaras, Skouri, & Parsopoulos, 2012) demonstrated the potential of effectively solving these problems by using modern population-based optimization algorithms, namely PSO, Differential Evolution, and Harmony Search. Although most of the studied algorithms were primarily designed for real-valued optimization problems, proper modifications in their operation as well as in formulation of the problem can render them applicable also on integer and mixed integer problems, such as the one under consideration. The reported good performance triggered our interest in further studying such algorithms on the ELSR problem.

In the present work, we considered a state-of-the-art population-based algorithm, namely *Differential Evolution* (DE). In the past, DE has been successfully applied on mixed integer engineering design problems. Recently, it was shown to be clearly superior than another popular algorithm of the same type, namely Genetic Algorithms (GAs), while its solutions were shown to lie also very close to exact Branch-and-Bound methods (Ponsich & Coello Coello, 2011). Moreover, the DE operators are based on difference vectors and they significantly differ from the corresponding GA binary operators (see also Feoktistov, 2006).

The performance of DE was assessed on the test suite proposed in Schulz (2011). The algorithm was also compared with the established SM-based variants from Schulz (2011), which constitute part of the state-of-the-art for this kind of problems. Our aim was to probe the potential of DE to serve as promising alternative for tackling the ELSR problem, enriching the available algorithmic artillery. For this reason, the basic variants of DE were thoroughly tested and the most promising ones were distinguished, along with their parameter settings. Additional performance-enhancing techniques such as restarting and 1-step local search were also considered. To the best of our knowledge, with the exception of the two very recent papers of TS and PSO that were mentioned above, there are no other existing metaheuristic optimization algorithms for the ELSR in the literature.

The rest of the paper is organized as follows: Section 2 provides the basic formulation of the problem. In Section 3, the DE algorithm is described and some of its essential properties are discussed. Section 4 exposes the necessary modifications in the formulation of the problem. The experimental setting and results are reported in Section 5, and the paper concludes with Section 6.

2. Original model formulation

The original ELSR problem considered in our study, consists of the Dynamic Lot Sizing model with both remanufacturing and manufacturing setup costs, as it was introduced in Teunter et al. (2006) and studied in Schulz (2011). This problem emerged as an extension of the original Wagner–Whitin problem (Wagner & Whitin, 1958) and considers a manufacturer that produces a single product over a finite planning horizon. At each time period, there is a known demand for the product as well as a number of returned items that can be completely remanufactured and sold as new. If the remanufactured items that are stored in inventory are inadequate to satisfy the demand, an additional number of items is manufactured. The aim is to determine the exact number of remanufactured and manufactured items per time period, in order to minimize the total holding and setup cost under various operational constraints.

In order to formally describe the considered model, we will henceforth use the following notation that closely follows the presentation of Schulz (2011):

t	time period, $t = 1, 2, \ldots, T$
D_t	demand for time period t
R_t	number of returned items in period t that can be completely remanufactured and sold as new
h^R	holding cost for the recoverable items per unit time
h^M	holding cost for the manufactured items per unit time
z_t^R	number of items that are eventually remanufactured in period t
z_t^M	number of manufactured items in period t
K^R	remanufacturing setup cost
K^M	manufacturing setup cost
y_t^R	inventory level of items that can be remanufactured in period t
y_t^M	inventory level of ready-to-ship items in period t

Now we can define the main cost optimization problem as follows (Schulz, 2011):

$$\min\ C = \sum_{t=1}^{T} \left(K^R \gamma_t^R + K^M \gamma_t^M + h^R y_t^R + h^M y_t^M \right), \tag{1}$$

where

$$\gamma_t^R = \begin{cases} 1, & \text{if } z_t^R > 0, \\ 0, & \text{otherwise}, \end{cases} \qquad \gamma_t^M = \begin{cases} 1, & \text{if } z_t^M > 0, \\ 0, & \text{otherwise}, \end{cases} \tag{2}$$

are binary decision variables denoting the initiation of a remanufacturing or manufacturing lot, respectively. Naturally, the model is accompanied by a number of constraints (Schulz, 2011):

$$y_t^R = y_{t-1}^R + R_t - z_t^R, \quad y_t^M = y_{t-1}^M + z_t^R + z_t^M - D_t, \quad \forall t = 1, 2, \ldots, T, \tag{3}$$

$$z_t^R \leqslant Q\gamma_t^R, \quad z_t^M \leqslant Q\gamma_t^M, \quad \forall t = 1, 2, \ldots, T, \tag{4}$$

$$y_0^R = y_0^M = 0, \quad \gamma_t^R, \gamma_t^M \in \{0, 1\}, \quad \forall t = 1, 2, \ldots, T, \tag{5}$$

$$y_t^R, y_t^M, z_t^R, z_t^M \geqslant 0, \quad \forall t = 1, 2, \ldots, T. \tag{6}$$

The constraints defined in (3) guarantee the inventory balance taking into consideration the incoming and outcoming items. Eq. (4) guarantees that fixed costs are introduced whenever a new lot is initiated. The parameter Q is a sufficiently large number; in Schulz (2011) it is suggested the use of the total demand during the planning horizon. Finally, (5) and (6) ensure that inventories are initially empty and all parameters assume only reasonable values.

Some interesting properties of the considered model were identified in Teunter et al. (2006). Firstly, there is a possibility of attaining optimal solutions that do not adhere to the zero-inventory property. Secondly, although theoretically the (mixed integer) problem can be solved to optimality, there is strong evidence that it is NP-hard. This conjecture was stated in Teunter et al. (2006) and verified by the findings in Schulz (2011), offering motivation for the use of (meta) heuristic algorithms such as the adapted SM, TS, and PSO. Also, it triggered our interest in further investigating the problem with the DE algorithm, which is described in the following section.

3. Differential Evolution

Differential Evolution (DE) is a population-based metaheuristic algorithm, initially introduced by Storn and Price (1997). It employs a group, called a *population*, of search points, called *individuals*, to iteratively probe the search space. The search operators of DE are based on linear combinations of difference vectors produced from individuals in the population, as well as on recombination procedures that roughly resemble those of Evolutionary Algorithms (Spears, 2000). During the past decade, DE has gained increasing popularity due to its efficiency in a wide range of applications (Chakraborty, 2008; Neri & Tirronen, 2010; Qing, 2009).

Consider the n-dimensional global optimization problem,

$$\min_{\mathbf{x} \in X \subset \mathbb{R}^n} C(\mathbf{x}).$$

DE employs a population of N individuals,

$$S = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\},$$

where N is a user-defined *population size*, and its value depends on the available computational resources as well as on the problem at hand. Each individual is an n-dimensional vector in the search space $X \subset \mathbb{R}^n$,

$$\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{in})^\top \in X, \quad i \in I = \{1, 2, \ldots, N\}.$$

The population is randomly initialized, usually following a uniform distribution over the search space.

The positions of the individuals are updated according to three procedures that sample individuals from the population, combine them to produce new candidate solutions, and select the best among the existing and the new ones. These procedures are called *mutation*, *crossover*, and *selection*, respectively, and they are described below.

3.1. Basic search procedures and operators

Mutation

A mutation operator produces a new vector $\mathbf{v}_i$ for each individual $\mathbf{x}_i$, $i \in I$, by combining some of the individuals of the population. There is a variety of mutation operators reported in the DE literature. Usually, the following five operators are considered:

$$(\text{DE1}): \mathbf{v}_i(t+1) = \mathbf{x}_g(t) + F\left(\mathbf{x}_{r_1}(t) - \mathbf{x}_{r_2}(t)\right), \quad (7)$$
$$(\text{DE2}): \mathbf{v}_i(t+1) = \mathbf{x}_{r_1}(t) + F\left(\mathbf{x}_{r_2}(t) - \mathbf{x}_{r_3}(t)\right), \quad (8)$$
$$(\text{DE3}): \mathbf{v}_i(t+1) = \mathbf{x}_i(t) + F\left(\mathbf{x}_g(t) - \mathbf{x}_i(t) + \mathbf{x}_{r_1}(t) - \mathbf{x}_{r_2}(t)\right), \quad (9)$$
$$(\text{DE4}): \mathbf{v}_i(t+1) = \mathbf{x}_g(t) + F\left(\mathbf{x}_{r_1}(t) - \mathbf{x}_{r_2}(t) + \mathbf{x}_{r_3}(t) - \mathbf{x}_{r_4}(t)\right), \quad (10)$$
$$(\text{DE5}): \mathbf{v}_i(t+1) = \mathbf{x}_{r_1}(t) + F\left(\mathbf{x}_{r_2}(t) - \mathbf{x}_{r_3}(t) + \mathbf{x}_{r_4}(t) - \mathbf{x}_{r_5}(t)\right), \quad (11)$$

where t denotes the iteration counter; $F \in (0, 1]$ is a (typically) fixed user-defined parameter; g denotes the index of the best individual in the population, i.e.,

$$g = \arg\min_{j \in I} C(\mathbf{x}_j),$$

and $r_k \in I,\ k = 1, 2, \ldots, 5$, are mutually different, randomly selected indices that differ also from the index i. Obviously, in order to enable all mutation operators it must hold that $N > 5$.

Recombination

After mutation, the recombination operator is applied on the generated vector $\mathbf{v}_i$, producing a *trial vector*,

$$\mathbf{u}_i = (u_{i1}, u_{i2}, \ldots, u_{in})^\top,$$

as follows:

$$u_{ij}(t+1) = \begin{cases} v_{ij}(t+1), & \text{if } \mathcal{R} \leqslant CR \text{ or } j = RI(i), \\ x_{ij}(t), & \text{otherwise}, \end{cases} \quad (12)$$

where $j = 1, 2, \ldots, n$; $\mathcal{R}$ is a uniformly distributed random variable in the range $[0, 1]$; $CR \in (0, 1]$ is a user-defined *crossover constant*; and $RI(i)$ is an index randomly selected from the set $\{1, 2, \ldots, n\}$. A different value of $\mathcal{R}$ is used for each i and j, while the use of $RI(i)$ guarantees that at least one of the components of the mutated vector $\mathbf{v}_i$ is inherited to the trial vector $\mathbf{u}_i$.

Selection

Finally, the trial vector $\mathbf{u}_i$ is compared against the original individual $\mathbf{x}_i$ and the best between them is included in the population of the next generation:

$$\mathbf{x}_i(t+1) = \begin{cases} \mathbf{u}_i(t+1), & \text{if } C(\mathbf{u}_i(t+1)) < C(\mathbf{x}_i(t)), \\ \mathbf{x}_i(t), & \text{otherwise}. \end{cases} \quad (13)$$

The new population iteratively undergoes the same procedure as above, until a user-defined stopping criterion is fulfilled. This is usually related to the quality of the achieved solutions or the available computational resources.

3.2. Tackling integer optimization problems

The original DE algorithm was designed for solving continuous optimization problems. However, numerical evidence indicates that DE can be successfully applied even on integer or mixed integer optimization problems (Piperagkas et al., 2012). The simplest technique to achieve it, is the rounding of the vector components to their nearest integer after mutation and recombination. Thus, before the selection phase of (13), the trial vector undergoes rounding to the nearest integer as follows:

$$\mathbf{u}_i = (\bar{u}_{i1}, \bar{u}_{i2}, \ldots, \bar{u}_{in})^\top,$$

where

$$\bar{u}_{ij} = \lfloor u_{ij} + 0.5 \rfloor,$$

and $\lfloor \cdot \rfloor$ stands for the floor function. Due to the simplicity and negligible computational cost of the rounding technique, we adopted it also in the present work.

3.3. Exploration vs exploitation

Up-to-date, there is no theoretical finding to support a widely acceptable parameter setting of DE. Various settings have been used in the literature (Epitropakis, Tasoulis, Pavlidis, Plagianakos, & Vrahatis, 2011) but their performance appears to be strongly dependent on the employed mutation operator and the considered problem. Nevertheless, by their nature, the mutation operators that involve the best individual of the population are expected to be more efficient in exploitation, while the ones that use randomly selected individuals exhibit better exploration properties.

In order to verify these properties, we considered an experimental scenario that resembles the actual one described later in Section 5. Specifically, we considered a population of $N = 60$ individuals of dimension $n = 24$. The population was randomly generated within the search space $X = [0, 1000]^{24}$. We considered five DE variants, one for each mutation operator in (7)–(11). For each variant, we generated 1000 offspring per individual, and measured their average distances from the best of the population. This procedure was repeated for all pairwise combinations of $F,\ CR \in \{0.3, 0.5, 0.7\}$. Then, we compared the different DE variants with respect to their offspring's average distances from the best, as a measure of their potential for exploration.

Specifically, there were 45 triplets (OP, F, CR) with $OP \in \{\text{DE1}, \ldots, \text{DE5}\}$ and $F,\ CR \in \{0.3, 0.5, 0.7\}$. Each triplet was compared against all the others. A triplet scored a "win" whenever it achieved a higher average distance than another one, with statistical significance 99%. The statistical significance was checked using the Wilcoxon rank sum test.

Fig. 1 displays the number of wins per variant and combination of (F, CR). Clearly, the variants based on DE2 and DE5 exhibit the highest average offspring distances from the best individual. This is in line with the intuitive expectation that using randomly selected individuals in mutation, promotes exploration. Indeed, DE2 and DE5 are the two operators that use only randomly selected individuals as we can see in (8) and (11). These two variants were shown to be more efficient than the rest also in previous works (Piperagkas et al., 2012) where integer problems were considered.

Almost identical observations were made when, instead of the distance from the best, the diversity of the offspring itself was used as a measure of its potential for exploration. The diversity was assessed on the basis of the average standard deviation of the offspring vectors per direction component.

Besides the choice of mutation operator, the values of F and CR affect DE's performance. Although mutation operators have inherent randomization (e.g., random selection of vector indices), the difference vectors are deterministically produced when F assumes fixed values. However, better diversity can be achieved if F is taken randomly. This is illustrated in Fig. 2, where we show the mutated

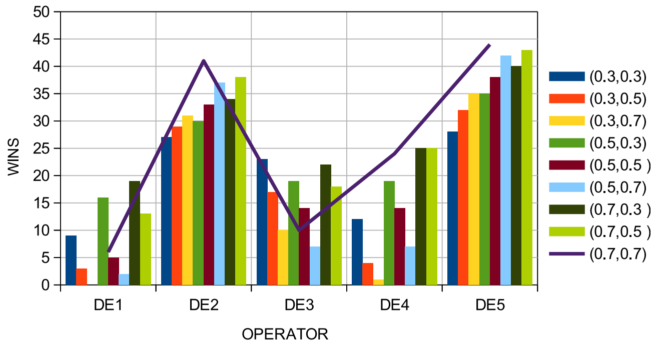

Fig. 1. Number of wins per DE variant and combination (F, CR).

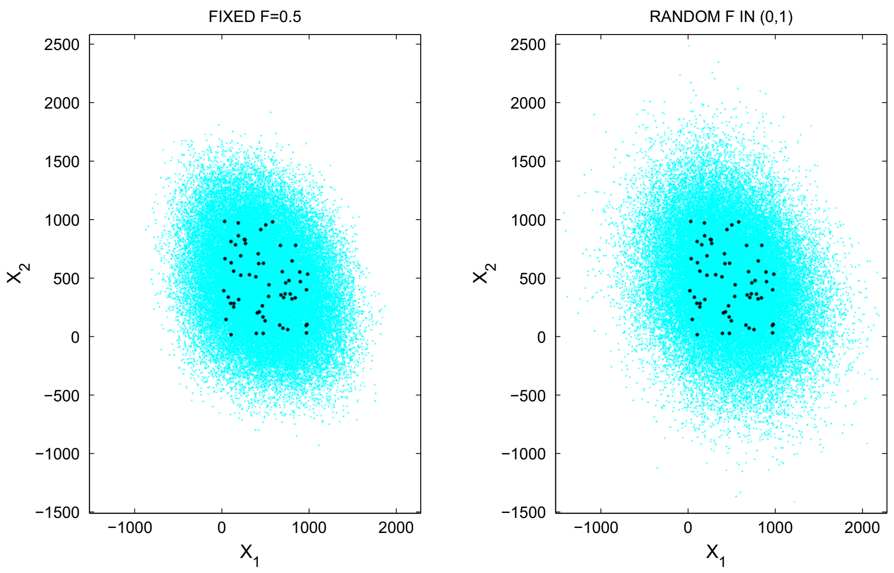

Fig. 2. Vectors produced using mutation operator DE5 for fixed $F = 0.5$ (left) and random F in the range $(0, 1)$ (right).

vectors produced by DE5 for fixed $F = 0.5$ as well as for random F uniformly distributed in $(0, 1)$ (i.e., with mean value equal to 0.5). Specifically, we produced 1000 mutated vectors (denoted as cyan dots) for each one of the 60 individuals (denoted as black stars) of a randomly generated, 2-dimensional population. It shall be noted that both the population and the mutated vectors were rounded to the nearest integer, in order to take into consideration the effect of rounding in the diversity of the produced offspring.

As we see in Fig. 2, the dispersion of the mutated vectors was higher for the case of random F. The standard deviation of the vectors produced with random F was increased by almost 10% per direction component. For this reason, it can be considered as a good starting point for experimentation. This was also verified in our experiments, as it will be reported later in Section 5. Since F is typically taken in the range $(0, 1]$, it is a reasonable choice to consider,

$$F \sim \mathcal{U}(0, 1),$$

where $\mathcal{U}$ stands for the Uniform distribution, and assume a different, random value of F for each vector component in the mutation operators of (7)–(11).

4. Modeling and application issues

In our study, the decision variables of the considered optimization problem (see Section 2) are the integer quantities z_t^R and z_t^M of the remanufactured and manufactured items, respectively, per time period. The rest of the problem's parameters can be derived from these values, as will be shown below. Thus, for a planning horizon of T time periods, the corresponding optimization problem is of dimension $n = 2 \times T$. If we consider the general form $\mathbf{x} = (x_1, x_2, \ldots, x_n)^\top$ of an individual in DE, we can use the following mapping between its components and the problem's decision variables:

$$\begin{array}{ccccccc} x_1 & x_2 & x_3 & x_4 & \cdots & x_{2T-1} & x_{2T} \\ \updownarrow & \updownarrow & \updownarrow & \updownarrow & & \updownarrow & \updownarrow \\ z_1^R & z_1^M & z_2^R & z_2^M & \cdots & z_T^R & z_T^M \end{array} \tag{14}$$

Thus, each individual constitutes a different setting of the decision variables.

For a given decision vector, the parameters γ_t^R and γ_t^M of (1) were computed directly from (2), while y_t^R and y_t^M were computed with the recursive relations of (3), thereby becoming dependent on the decision variables. In order to retain the integrity of the original model and the corresponding constraints defined in (1)–(5), our modified model consisted of the original minimization problem in the form of (1), while the constraints were redefined as follows:

$$0 \leqslant x_t^R \leqslant \sum_{\tau=1}^{t} R_\tau, \quad 0 \leqslant x_t^M \leqslant \sum_{\tau=t}^{T} D_\tau, \quad \forall t = 1, 2, \ldots, T. \tag{15}$$

These constraints were used to bound the decision variables and, consequently, the individuals within bounded search ranges. Additionally, the following constraints were imposed:

$$\sum_{t=1}^{T} (x_t^R + x_t^M) = \sum_{t=1}^{T} D_t, \tag{16}$$

$$\sum_{\tau=1}^{t} (x_\tau^R + x_\tau^M) \geqslant \sum_{\tau=1}^{T} D_\tau, \quad \forall t = 1, 2, \ldots, T-1, \tag{17}$$

$$\sum_{\tau=1}^{t} R_\tau \geqslant \sum_{\tau=1}^{t} x_\tau^R, \quad \forall t = 1, 2, \ldots, T, \tag{18}$$

$$y_t^R, y_t^M \geqslant 0, \quad \forall t = 1, 2, \ldots, T. \tag{19}$$

The original problem of (1) along with the representation of (14) and the constraints of (15)–(19), constitute the model that was used in the present study. Notice that the modified model contains a total of $Q = 6 \times T$ constraints.

Another crucial issue that required special treatment was the constraint-handling strategy in order to avoid infeasible solutions. For this purpose, a simple yet effective penalty function was used to guide the individuals towards feasible regions of the search space by penalizing the infeasible ones. The penalty function had the general form:

$$\widetilde{C}(\mathbf{x}) = C(\mathbf{x}) + \sum_{i=1}^{Q} |P_i(\mathbf{x})|, \tag{20}$$

where the penalty term $P_i(\mathbf{x})$ was the amount of violation of the i-th constraint by the decision vector $\mathbf{x}$. In addition to the penalty function, we adopted a strategy that was shown to be successful in previous works (Piperagkas et al., 2011; Piperagkas et al., 2012). It consists of the following rules that determine the most desirable one between any two decision vectors:

(a) If both vectors are feasible, the one with the smallest objective value is selected.
(b) If both vectors are infeasible, the one with the smallest total penalty is selected.
(c) Between a feasible and an infeasible vector, the feasible one is always preferable.

These rules were applied whenever an individual was compared to its trial vector, in order to promote the inclusion of feasible candidate solutions against infeasible ones in the population. It shall be underlined that we did not require the algorithm to be initialized with a feasible population.

5. Experimental assessment

The DE algorithm was applied on the established test suite used in Schulz (2011), which is an extended version of the one introduced in Teunter et al. (2006). This is the only established test suite and consists of a full factorial study of various problem instances with a common planning horizon of $T = 12$ time periods. The setup costs K^M and K^R, as well as the holding costs h^R, assumed three different values each. The demands and returns were drawn from normal distributions with both large and small deviations. The mean of the returns' distribution assumed also three different values (return ratios). The exact configuration of the employed test suite is reported in Table 1. For each combination of problem parameters, the test suite contains 20 different problem instances (Schulz, 2011), resulting in a total number of 6480 different problem instances.

The optimal value per problem instance is provided in the test suite. The main optimization goal is to achieve the lowest possible *percentage error from the optimal solution* within a prespecified computational budget (maximum number of function evaluations). The percentage error is computed as follows:

$$\frac{C_{\text{best}} - C^*}{C^*} \times 100, \tag{21}$$

where C_{best} is the best solution value detected by the algorithm and C^* is the global minimum of the problem. This performance measure was used also in previous works (Schulz, 2011; Moustaki et al., 2013), and it was adopted here to facilitate comparisons between algorithms. Similarly to relevant works with stochastic population-based algorithms (Moustaki et al., 2013), for each problem instance there were 30 independent experiments conducted and the achieved percentage errors were statistically analyzed with respect to their mean, standard deviation, and maximum value.

Table 1
Parameter values for the employed test suite (Schulz, 2011).

Parameter description	Value(s)
Setup costs	$K^M,\ K^R \in \{200, 500, 2000\}$
Holding cost for ready-to-ship products	$h^M = 1$
Holding cost for recoverable products	$h^R \in \{0.2, 0.5, 0.8\}$
Demand for time period t	$D_t \sim \mathcal{N}(\mu_D, \sigma_D^2)$
	$\mu_D = 100$
	$\sigma_D = 10\%$ of μ_D (small variance)
	$\sigma_D = 20\%$ of μ_D (large variance)
Returns for time period t	$R_t \sim \mathcal{N}(\mu_R, \sigma_R^2)$
	$\mu_R = 30\%, 50\%, 70\%$ of μ_D
	$\sigma_R = 10\%$ of μ_R (small variance)
	$\sigma_R = 20\%$ of μ_R (large variance)

Table 2
Parameter values for the DE variants.

Parameter description	Value(s)
Number of experiments per problem instance	30
Population size	$N = 60$
Maximum number of function evaluations	10^8
DE mutation operator	DE1–DE5
Parameter values	$F,\ CR \in \{0.3, 0.5, 0.7\}$ (first round)
	$F \sim \mathcal{U}(0, 1)$ (for DE5 only)

Our experimentation with DE was initiated with five variants based on the mutation operators DE1-DE5 of (7)–(11). Henceforth we will denote these variants simply as DE1-DE5. For each variant, we considered all 9 parameter combinations of fixed $F,\ CR \in \{0.3, 0.5, 0.7\}$. Therefore, we had a total of 45 algorithmic instances. For each one, 30 independent experiments were conducted per problem instance, resulting in a total of $45 \times 30 \times 6480 = 8{,}748{,}000$ independent experiments. In a second round of experiments, the case of random F was considered. All the employed DE parameters are summarized in Table 2. The same setting as in Moustaki et al. (2013) was adopted to facilitate comparisons with the PSO approach, as well as with the established SM variants reported in Schulz (2011).

The preliminary experiments with fixed parameter values, revealed that DE5 with $F = 0.5$ and $CR = 0.3$ was the most efficient setting (we omit detailed report of all the preliminary results for the sake of compact presentation). This was a premature yet clear indication that successful runs were related to the exploration properties of the algorithm (recall discussion in Section 3.3). Also, we observed that sometimes the algorithm was spending a considerable number of function evaluations to detect a better solution that was marginally different than its current best. This deficiency was attributed to the rounding of the vector components to the same integer value for some iterations (search stagnation).

For this reason, we adopted an 1-step local search on the best individual whenever a new one was found. Specifically, as soon as a new best was detected, all its component values were perturbed by ± 1 one-by-one. The new vectors were evaluated and, if one was better than the best, it was replacing it in the population. The local search adds a computational overhead of $2 \times n$ function evaluations per point of application. For this reason, it was applied once, only on every new best individual. Nevertheless, it provided considerable performance gain. On the other hand, regularly restarting the algorithm did not offer significant benefits in our preliminary experiments and, thus, it was abandoned. The DE5 with the 1-step local search will be henceforth denoted as DE5$_F$.

Besides the DE5$_F$ variant, we also considered a DE5 variant with random parameter F, motivated from the observations in Section 3.3. Specifically, we considered a uniformly distributed $F \sim \mathcal{U}(0, 1)$ in (11), which assumes a different value per vector component. Also, the better performance of smaller CR values in our preliminary experiments, offered incentives for further investigation of even smaller values. Thus, we experimented with the values $CR = 0.2$, 0.1, and 0.05. The results revealed that $CR = 0.1$ was the most promising value. In practice, CR controls the number of component values of the trial vector that are inherited to the new one (see (13)). Small values allow only a few of the components of the trial vector to be included in the new one, while the rest come from the current point of the population.

Thus, the good performance of DE5 with smaller values of CR can be attributed to the smaller changes in the components of each vector. For the case of $CR = 0.1$, only 2 or 3 out of the 24 components of each new candidate solution are expected to differ in $\mathbf{x}_i(t+1)$ from the original point $\mathbf{x}_i(t)$ after the selection procedure of (13). This is also in line with the 1-step local search procedure described above, where small perturbations of the best individual could offer performance benefits to the algorithm. The described DE5 variant with random $F \sim \mathcal{U}(0,1)$ and $CR = 0.1$, will be henceforth denoted as DE5$_R$. It shall be noted that the value $CR = 0.1$ was also considered with DE5$_F$, but exhibited slightly inferior performance.

The results of the most promising variants, namely DE5$_F$ and DE5$_R$, are reported in Tables 3–5, for all problem instances and for the problem parameters of Table 1. Specifically, we report the mean, standard deviation, and maximum percentage error from the optimal solution for both approaches. For comparison purposes, the corresponding quantities of the Silver-Meal (SM) variants that are reported in Schulz (2011), as well as for the Particle Swarm Optimization (PSO) that are reported in Moustaki et al. (2013), are also included in the tables. The results for the TS approach in Li et al. (2013) were not used for comparisons, since the percentage error from the optimal solution in Li et al. (2013) is not computed according to (21), but using the formula $(C_{\text{best}} - C^*)/C_{\text{best}} \times 100$, as reported in Li et al. (2013). Thus, they are incomparable with the rest of the algorithms. The best value(s) per row is boldfaced in our tables.

A first evaluation of the results clearly reveals the superiority of two algorithms, namely SM$_4^+$ and DE5$_R$. Although the overall mean value of percentage error from the optimal solution is marginally better for SM$_4^+$ (2.2% against 2.3%) as shown in the first line of Table 3, DE5$_R$ follows closely, achieving the smallest maximum value (22.5% against 24.3%). Also, the standard deviations of the two algorithms are slightly different (2.9% against 3.1%). The reported values suggest that the observed performance differences are within the limit of statistical error and may be negligible.

A closer examination of the tables, also shows the potential of DE5$_R$ to outperform or closely follow SM$_4^+$, especially for higher values of the problem parameters. Indicatively, we can refer to the case of large variance for demand and returns in Table 3, as well as the cases of higher values of return ratio, K^M, and K^R, in Tables 4 and 5. In most of these cases, SM$_4^+$ seemed to have declining

Table 3
Percentage cost error of DE5$_F$ and DE5$_R$ for all problem instances, as well as for different demand and returns variance. The corresponding results for SM variants (Schulz, 2011) and PSO (Moustaki et al., 2013) are also reported. The best values per line are boldfaced.

		SM$_2$ (%)	SM$_4$ (%)	SM$_2^+$ (%)	SM$_4^+$ (%)	PSO (%)	DE$_F$ (%)	DE5$_R$ (%)
All instances	Mean	7.5	6.1	6.9	**2.2**	4.3	3.4	2.3
	StD	7.9	7.6	7.9	**2.9**	4.5	5.0	3.1
	Max	49.2	47.3	49.2	24.3	49.8	39.2	**22.5**
Demand (small variance)	Mean	7.2	6.0	6.6	**2.1**	4.4	3.4	2.5
	StD	7.9	7.6	7.9	**2.8**	4.6	4.9	3.2
	Max	43.6	47.3	43.5	**18.9**	49.8	33.3	22.4
(large variance)	Mean	7.8	6.1	7.2	**2.4**	4.1	3.3	**2.4**
	StD	8.0	7.5	8.0	3.0	4.5	5.1	**2.9**
	Max	49.2	43.9	49.2	24.3	48.3	39.2	**23.9**
Returns (small variance)	Mean	7.3	6.1	6.8	**2.2**	4.3	3.4	2.5
	StD	7.8	7.6	7.8	**2.9**	4.6	4.9	3.3
	Max	47.2	47.3	47.2	**21.1**	46.7	39.2	22.4
(large variance)	Mean	7.7	6.1	7.1	**2.3**	4.2	3.4	**2.3**
	StD	8.0	7.5	8.0	2.9	4.5	5.0	**2.8**
	Max	49.2	46.3	49.2	**24.3**	49.8	33.3	24.5

Table 4
Percentage cost error of DE5$_F$ and DE5$_R$ for different return ratio and K^M values. The corresponding results for SM variants (Schulz, 2011) and PSO (Moustaki et al., 2013) are also reported. The best values per line are boldfaced.

			SM$_2$ (%)	SM$_4$ (%)	SM$_2^+$ (%)	SM$_4^+$ (%)	PSO (%)	DE$_F$ (%)	DE5$_R$ (%)
Return ratio	30%	Mean	5.5	3.7	4.9	**1.2**	3.5	3.3	2.2
		StD	5.5	4.5	5.4	**1.8**	3.1	5.0	3.0
		Max	31.3	28.5	31.3	**12.1**	45.5	28.2	25.5
	50%	Mean	8.5	7.3	8.0	**2.3**	4.1	3.5	2.4
		StD	9.4	8.2	9.3	**2.7**	4.0	5.2	2.9
		Max	40.1	41.8	39.8	**16.2**	34.0	32.5	18.3
	70%	Mean	8.4	7.2	8.0	3.3	5.1	3.3	**2.9**
		StD	8.0	8.7	8.0	3.5	5.9	4.7	**3.4**
		Max	49.2	47.3	49.2	24.3	49.8	39.2	**22.1**
K^M	200	Mean	4.3	3.4	3.5	**2.3**	4.0	3.3	2.9
		StD	4.5	3.6	4.0	**2.6**	3.1	3.9	3.1
		Max	20.2	17.6	20.2	**13.5**	45.5	24.4	17.5
	500	Mean	5.4	3.9	4.8	**2.1**	4.5	2.4	2.7
		StD	5.2	3.9	4.9	**2.5**	4.1	2.5	2.9
		Max	25.1	19.3	23.7	**12.8**	27.5	16.0	14.9
	2000	Mean	12.8	10.9	12.6	2.3	4.4	4.4	**1.8**
		StD	9.9	10.4	9.9	3.4	5.9	7.1	**3.2**
		Max	49.2	47.3	49.2	24.3	49.8	39.2	**23.5**

Table 5
Percentage cost error of DE5$_F$ and DE5$_R$ for different K^R and h^R values. The corresponding results for SM variants (Schulz, 2011) and PSO (Moustaki et al., 2013) are also reported. The best values per line are boldfaced.

			SM$_2$ (%)	SM$_4$ (%)	SM$_2^+$ (%)	SM$_4^+$ (%)	PSO (%)	DE$_F$ (%)	DE5$_R$ (%)
K^R	200	Mean	10.9	6.6	10.0	**1.9**	5.7	3.8	2.8
		StD	9.1	7.8	9.4	**2.1**	5.5	4.0	3.2
		Max	49.2	40.2	49.2	**11.8**	49.8	24.4	19.3
	500	Mean	7.9	8.1	7.3	3.4	3.8	1.9	**1.7**
		StD	6.6	8.2	6.6	3.2	4.1	2.0	**2.0**
		Max	34.7	47.3	34.7	19.1	37.4	12.6	**11.6**
	2000	Mean	3.7	3.5	3.6	**1.4**	3.3	4.4	1.6
		StD	6.0	5.7	5.9	2.9	3.5	7.1	**2.7**
		Max	29.4	25.7	29.4	24.3	45.5	39.2	**22.1**
h^R	0.2	Mean	5.9	5.3	5.8	**1.7**	4.5	3.0	1.8
		StD	8.0	8.0	8.0	**2.5**	5.2	5.3	**2.5**
		Max	42.9	47.3	42.9	**21.1**	49.8	35.6	21.7
	0.5	Mean	7.5	6.5	7.0	**2.3**	4.3	3.3	**2.3**
		StD	7.7	7.6	7.7	3.0	4.5	5.0	**2.9**
		Max	49.2	42.4	49.2	24.3	45.5	39.2	**23.6**
	0.8	Mean	9.1	6.3	8.1	**2.8**	4.0	3.7	3.0
		StD	7.7	7.0	7.8	**3.0**	3.9	4.5	3.3
		Max	44.4	40.3	44.4	20.6	42.9	32.5	**19.5**

performance, with DE5$_R$ exhibiting complementary behavior. The mean and standard deviation of the percentage error for both algorithms is graphically illustrated also in Fig. 3 to facilitate optical interpretation. Based on our model formulation presented in Section 4, larger values of the parameters correspond to larger search ranges for the DE. Taking into consideration that the individuals in DE's population are produced such that the constraints of (15) are *de facto* satisfied, we can infer that DE's good performance related to higher search ranges is intuitively sound, because higher search ranges result in smaller number of infeasible vectors (that are eventually restricted on the boundary of the search space) per iteration of the algorithm.

Regarding the time complexity of the DE variants, it ranged from a few seconds (in the cases where the optimal solution was attained) up to 2.5 min for the cases where the number of function evaluations was exceeded. The reported times are indicative, since they heavily depend on the implementation, the hardware, and the machine's load at the time of execution. All times refer to execution on Ubuntu Linux servers with Intel® Core™ i7 processors, 8 GB RAM, occupying all the available cores, while the source code was written in C++ (compiler `GCC ver. 4.6.3`). In our case, no effort was paid to further optimize the running time of the algorithms by using special optimization flags or software for the compilation of our source code. It shall be noted that, besides its efficiency, DE requires only minor implementation effort due to its simplicity and the lack of complicated procedures.

Our final concern was the scaling property of the DE5$_R$ approach. The established test suite that was used in the previous experiments contains only problems of $T = 12$ periods. However, following the practice in similar studies (e.g., Li et al., 2013) we produced a set of problem instances for three different time periods, namely $T = 24$, 48, and 52. Each problem instance was produced by properly extending the corresponding problem of Schulz's test suite. The extended parameter vectors (for demand, returns, return ratio, etc.) were produced by using the same probability distributions as in Schulz's test suite. The distinguished DE5$_R$ approach was applied on the new test problems according to the experimental framework of the main investigation for $T = 12$.

In the new experiments, we were interested in probing the relative performance scaling of our approach with respect to that of CPLEX. For this reason, a fixed execution time of 500 s was considered, and both CPLEX and DE5$_R$ were run on all test problems. Finally, the percentage error between DE's solution and the one obtained by CPLEX, averaged over all problem instances, were derived. The results are graphically illustrated in Fig. 4. As we can see, there is a clear logarithmic trend in the corresponding curves both for the mean and standard deviation of the percentage error between the solutions of CPLEX and DE5$_R$. This implies a smooth scaling capability for DE5$_R$.

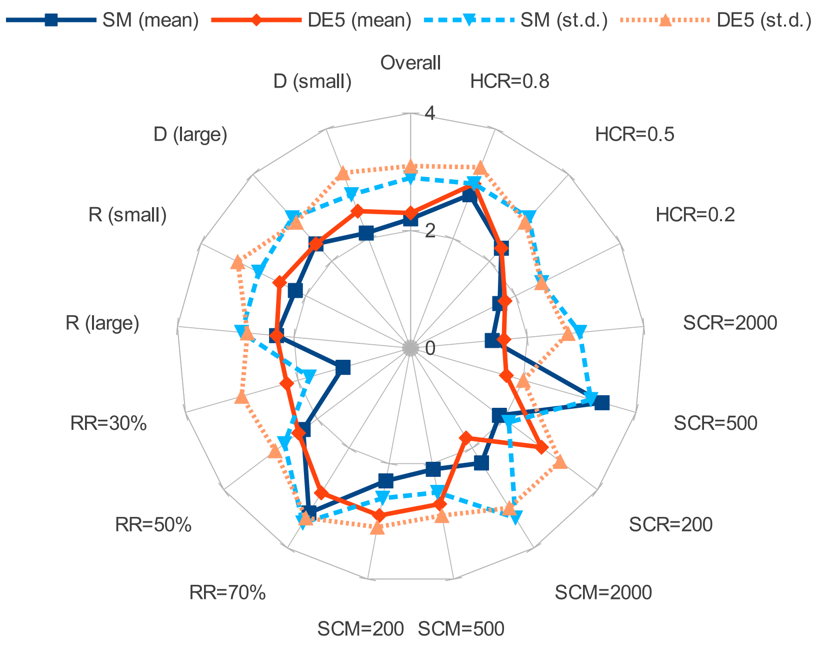

Fig. 3. Mean and standard deviation of the percentage error from optimal solution for SM$_4^+$ and DE5$_R$, for all problem cases. Labels are D: demand, R: returns, RR: return ratio, SCM: setup cost manufacturing, SCR: setup cost remanufacturing, and HCR: holding cost remanufacturing.

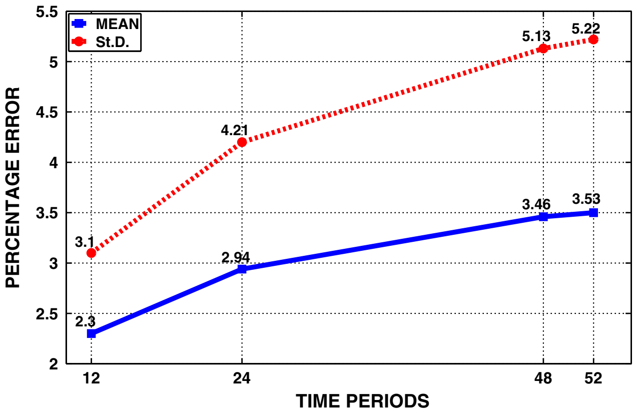

Fig. 4. Performance scaling for the DE5$_R$ algorithm.

6. Conclusions and future research

In the current work, we thoroughly investigate the performance of a very popular population-based algorithm, namely Differential Evolution (DE), on the Single-Item Dynamic Lot Sizing problem with returns and remanufacturing. The most promising variant of the algorithm was identified and properly modified to further enhance its performance. Also, necessary modifications in the formulation of the corresponding optimization problem were introduced. The algorithm was applied on an established test suite employed in previous studies. Its performance was analyzed and compared with state-of-the-art approaches, as well as with previously investigated (meta) heuristics.

The results suggest that DE is very competitive to the most efficient approaches, and it can be considered as a promising alternative for solving the considered problems. The distinguished DE variants were compared against established variants of the adapted SM algorithm and its enhanced versions, as well as against recently proposed PSO variants. The results are aligned (in terms of cost performance) with results from previous studies (Piperagkas et al., 2011, 2012), supporting the use of modern population-based optimization algorithms to tackle Dynamic Lot Sizing problems under various modifications.

Future work will contribute towards the direction of developing more refined versions in order to further enhance their performance on the specific problem type. Specialized operators as well as parallel implementations can offer the desirable performance boost. Also since the quality of return items plays an important role in Reverse Logistics, it would be interesting to study the DLSR problem assuming quality considerations of returned items. Another topic for further research would be the study of DLSR problem under stochastic demand and returns and to develop and test appropriate nature inspired or heuristics algorithms for its solution. Finally the present model can be extended to the case of two kinds of finished goods (one from fresh raw materials and the other one from returned items) with different selling prices in the market.

References

Ahiska, S. S., & Kurtul, E. (2014). Modeling and analysis of a product substitution strategy for a stochastic manufacturing/remanufacturing system. *Computers & Industrial Engineering, 72*(1), 1–11.

Akcali, E., & Cetinkaya, S. (2011). Quantitative models for inventory and production planning in closed-loop supply chains. *International Journal of Production Research, 49*(8), 2373–2407.

Chakraborty, U. K. (Ed.). (2008). *Advances in differential evolution. Studies in computational intelligence* (Vol. 143). Springer.

Epitropakis, M. G., Tasoulis, D. K., Pavlidis, N. G., Plagianakos, V. P., & Vrahatis, M. N. (2011). Enhancing differential evolution utilizing proximity-based mutation operators. *IEEE Transactions on Evolutionary Computation, 15*(1), 99–119.

Feoktistov, V. (2006). *Differential evolution: In search of solutions*. Springer.

Fleischmann, M., Bloemhof-Ruwaard, J. M., Dekker, R., van der Laan, E. A., van Numen, J. A. E. E., & van Wassenhove, L. N. (1997). Quantitative models for reverse logistics: A review. *European Journal of Operational Research, 103*(1), 1–17.

Golany, B., Yang, J., & Yu, G. (2001). Economic lot-sizing with remanufacturing options. *IIE Transactions, 33*(11), 995–1003.

Govindan, K., Soleimani, H., & Kannan, D. (2015). Reverse logistics and closed-loop supply chain: A comprehensive review to explore the future. *European Journal of Operational Research, 240*(3), 603–626.

Guide, V. D. R., Jr., Jayaraman, V., & Srivastava, R. (1999). Production planning and control for remanufacturing: a state-of-the-art survey. *Robotics and Computer Integrated Manufacturing, 15*(3), 221–230.

Helmrich, M. J. R., Jans, R., Den Heuvel, W., & Wagelmans, A. P. M. (2014). Economic lot-sizing with remanufacturing: Complexity and efficient formulations. *IIE Transactions, 46*(1), 67–86.

Li, X., Baki, F., Tian, P., & Chaouch, B. A. (2013). A robust bock-chain based tabu search algorithm for the dynamic lot sizing problem with product returns and remanufacturing. *Omega, The International Journal of Management Science, 42*(1), 75–87.

Moustaki, E., Parsopoulos, K. E., Konstantaras, I., Skouri, K., & Ganas, I. (2013). A first study of particle swarm optimization on the dynamic lot sizing problem with product returns. In *XI Balkan conference on operational research (BALCOR 2013)* (pp. 348–356). Belgrade, Serbia.

Neri, F., & Tirronen, V. (2010). Recent advances in differential evolution: A survey and experimental analysis. *Artificial Intelligence Review, 33*(1–2), 61–106.

Pineyro, P., & Viera, O. (2009). Inventory policies for the economic lot-sizing problem with remanufacturing and final disposal options. *Journal of Industrial and Management Optimization, 5*(2), 217–238.

Piperagkas, G. S., Voglis, C., Tatsis, V. A., Parsopoulos, K. E., & Skouri, K. (2011). Applying PSO and DE on multi-item inventory problem with supplier selection. In *The 9th metaheuristics international conference (MIC 2011)* (pp. 359–368), Udine, Italy.

Piperagkas, G. S., Konstantaras, I., Skouri, K., & Parsopoulos, K. E. (2012). Solving the stochastic dynamic lot-sizing problem through nature-inspired heuristics. *Computers & Operations Research, 39*(7), 1555–1565.

Ponsich, A., & Coello Coello, C. A. (2011). Differential evolution performances for the solution of mixed-integer constrained process engineering problems. *Applied Soft Computing, 11*(1), 399–409.

Qing, A. (2009). *Differential evolution: Fundamentals and applications in electrical engineering*. Wiley-IEEE Press.

Richter, K., & Sombrutzki, M. (2000). Remanufacturing planning for the reverse Wagner/Whitin models. *European Journal of Operational Research, 121*(2), 304–315.

Richter, K., & Weber, J. (2001). The reverse Wagner/Whitin model with variable manufacturing and remanufacturing cost. *International Journal of Production Economics, 71*(1–3), 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.

Spears, W. M. (2000). *Evolutionary algorithms: The role of mutation and recombination*. Springer.

Steeneck, D. W., & Sarin, S. C. (2013). Pricing and production planning for reverse supply chain: a review. *International Journal of Production Research, 51*(23–24), 6972–6989.

Stindt, D., & Sahamie, R. (2014). Review of research on closed loop supply chain management in the process industry. *Flexible Services and Manufacturing Journal, 26*(1–2), 268–293.

Storn, R., & Price, K. (1997). Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization, 11*, 341–359.

Teunter, R. H., Bayindir, Z. P., & Van den Heuvel, W. (2006). Dynamic lot sizing with product returns and remanufacturing. *International Journal of Production Research, 44*(20), 4377–4400.

Teunter, R., Kaparis, K., & Tang, O. (2008). Multi-product economic lot scheduling problem with separate production lines for manufacturing and remanufacturing. *European Journal of Operational Research, 191*(4), 1241–1253.

Teunter, R., Tang, O., & Kaparis, K. (2009). Heuristics for the economic lot scheduling problem with returns. *International Journal of Production Economics, 118*(1), 323–330.

Thierry, M. C., Salomon, M., van Numen, J., & van Wassenhove, L. N. (1995). Strategic issues in product recovery management. *California Management Review, 37*(2), 114–135.

Wagner, H. M., & Whitin, T. M. (1958). Dynamic version of the economic lot size model. *Management Science, 5*(1), 88–96.

Yang, J., Golany, B., & Yu, G. (2005). A concave-cost production planning problem with remanufacturing options. *Naval Research Logistics, 52*(5), 443–458.

Zanoni, S., Segerstedt, A., & Tang, O. (2012). Multi-product economic lot scheduling problem with manufacturing and remanufacturing using a basic period policy. *Computers & Industrial Engineering, 62*(4), 1025–1033.