Int J Adv Manuf Technol (2015) 80:31–45
DOI 10.1007/s00170-015-6993-6

ORIGINAL ARTICLE

Novel Pareto-based meta-heuristics for solving multi-objective multi-item capacitated lot-sizing problems

Vahid Hajipour[1] · **AmirSaman Kheirkhah**[2] · **Madjid Tavana**[3,4] · **Nabil Absi**[5]

Received: 19 November 2014 / Accepted: 4 March 2015 / Published online: 18 March 2015

Abstract Capacitated production lot-sizing problems (LSPs) are challenging problems to solve due to their combinatorial nature. We consider a multi-item capacitated LSP with setup times, safety stocks, and demand shortages plus lost sales and backorder considerations for various production methods (i.e., job shop, batch flow, or continuous flow among others). We use multi-objective mathematical programming to solve this problem with three conflicting objectives including: (i) minimizing the total production costs; (ii) leveling the production volume in different production periods; and (iii) producing a solution which is as close as possible to the just-in-time level. We also consider lost sales, backorders, safety stocks, storage space limitation, and capacity constraints. We propose two novel Pareto-based multi-objective meta-heuristic algorithms: multi-objective vibration damping optimization (MOVDO) and a multi-objective harmony search algorithm (MOHSA). We compare MOVDO and MOHSA with two well-known evolutionary algorithms called the non-dominated sorting genetic algorithm (NSGA-II) and multi-objective simulated annealing (MOSA) to demonstrate the efficiency and effectiveness of the proposed methods.

Keywords Lot-sizing · Computational Intelligence · Multi-objective optimization · Harmony search · Vibration damping optimization · MOSA · NSGA-II

✉ Madjid Tavana
tavana@lasalle.edu

Vahid Hajipour
v.hajipour@basu.ac.ir

AmirSaman Kheirkhah
kheirkhah@basu.ac.ir

Nabil Absi
absi@emse.fr

[1] Young Researchers and Elite Club, Qazvin Branch, Islamic Azad University, Qazvin, Iran

[2] Industrial Engineering Department, Bu-Ali Sina University, Hamedan, Iran

[3] Business Systems and Analytics Department, Lindback Distinguished Chair of Information Systems and Decision Sciences, La Salle University, Philadelphia, PA 19141, USA

[4] Business Information Systems Department, Faculty of Business Administration and Economics, University of Paderborn, 33098 Paderborn, Germany

[5] Department of Manufacturing Sciences and Logistics, Ecole des Mines de Saint-Etienne, 13541 Gardanne, France

1 Introduction

Lot-sizing problems (LSPs) are production planning problems with multiple setups between production lots. There is a tradeoff between the cost of producing a product in every period and the inventory costs associated with producing the product in large quantities with fewer setups. Therefore, the goal of LSP's is to determine the time periods for production as well as the production quantities that minimize production, setup, and inventory holding costs while satisfying demand. The capacitated LSPs are combinatorial optimization problems whose objectives are to find production plans that minimize production, setup, and inventory costs while meeting the demands for the items over a finite planning horizon without delays. The multi-objective multi-item capacitated LSPs are non-deterministic polynomial-time (NP)-hard problems since the well-known mono-objective version of the problem [1] is NP-hard [2]. The algorithms used for solving the capacitated LSPs include exact methods, specialized heuristics, and mathematical programming-based heuristics. Karimi et al. [3]

and Robinson et al. [4] have evaluated the mathematical models and algorithms for capacitated LSPs.

We introduce a mixed-integer-programming formulation for the capacitated LSPs with three sets of conflicting objectives: (i) minimize the total cost of production, inventory, shortage, and safety stock; (ii) level the production volume in different production periods; and (iii) maintain the production level as close as possible to the just-in-time production level. We propose two novel meta-heuristic algorithms based upon vibration damping optimization (VDO; i.e., multi-objective VDO or MOVDO) and the harmony search algorithm (HSA; i.e., multi-objective HAS or MOHSA). We compare MOVDO and MOHSA with two well-known evolutionary algorithms, the non-dominated sorting genetic algorithm (NSGA-II), and multi-objective simulated annealing (MOSA), in order to demonstrate the efficiency and effectiveness of the proposed methods. In the following sub-sections, we first describe how the LSP has been applied in production planning and then present a brief review of the relevant multi-objective optimization literature.

1.1 Lot-sizing problem

Production planning involves transforming raw materials into final goods while satisfying demands and minimizing costs. The LSP is a well-known optimization problem often applied to production planning problems with time-varying demand over a multi-period planning horizon. LSPs are inherently complex problems with multiple and conflicting objectives. For example, increasing the multi-item offerings may have unintended consequences as a result of a company's ability to satisfy their customer demands.

Production planning typically encompasses long-, medium-, and short-term planning. Long-term planning involves strategic decisions such as product, equipment, facility location, and resource planning. Medium-term planning involves tactical decisions such as material requirement planning, production planning and control, and lot-sizing decisions during the planning period. Short-term planning involves operational decisions such as daily scheduling, planning, and forecasting [3]. In this study, we concentrate on medium-term production planning with an emphasis on lot-sizing decisions.

Since the seminal papers by Wagner and Whitin [5] and Manne [6], a great deal of research has been done on LSPs. Single-item LSPs have received a considerable amount of attention because of their relative simplicity and their importance as a sub-problem of some more complex LSPs (see Brahimi et al. [7] for a complete review of the single-item LSPs).

Production planning models generally involve multiple items, restrictive capacities, and significant setup times. Kazan et al. [8] have suggested three lot-sizing methodologies. Two of the methods are modified versions of the Wagner–Whitin [5] algorithm and the Silver–Meal [9] heuristic. They also proposed a new mixed-integer linear programming formulation.

Shortages are commonly assumed in lot-sizing problems. Shortages correspond to demands that cannot be satisfied, which can be either lost or backordered. Loparic et al. [10] address single-item uncapacitated LSPs with sales instead of fixed demands and lower bounds on stock variables. They propose valid inequalities to strengthen the linear programming relaxation. Aksen et al. [11] introduced a profit maximization version of the Wagner–Whitin [5] algorithm for deterministic uncapacitated single-item LSPs with lost sales. Absi and Kedad-Sidhoum [12] proposed a multi-item capacitated LSP with setup times and safety stock in which demand can be totally or partially lost. They also presented mixed-integer programming heuristics based on a planning horizon decomposition strategy to find a feasible solution. Following the introduction of their multi-item capacitated LSP, Absi and Kedad-Sidhoum [13] extended their model by considering shortage costs. Moreover, Absi and Kedad-Sidhoum [14] proposed a multi-item capacitated lot-sizing model with setup times, safety stock shortage costs, and demand shortage costs. To solve their model, they first proposed a Lagrangian relaxation of the resource capacity constraints and then solved the resulting sub-problem using a dynamic programming algorithm. Absi et al. [15] proposed a multi-item capacitated LSP with setup times and lost sales. They proposed a non-myopic heuristic based on a probing strategy and a refining procedure to find feasible solutions. They also proposed a meta-heuristic based on the adaptive large neighborhood search principle to improve their solution.

1.2 Multi-objective optimization

Chen and Thizy [2] proved that multi-item capacitated LSPs with setup times are NP-hard problems. Many researchers have attempted to solve multi-item capacitated LSPs very close to optimality [16, 17]. However, they were not successful in solving large-scale problems because they could not anticipate the number of cutting planes or the number of iterations that were required in a branch-and-bound approach. Due to the complexity of the problem, numerous meta-heuristic approaches have been proposed to solve large-scale multi-item capacitated LSPs [18–21]. In addition, several solution procedures involving simultaneous optimization of multiple objectives are proposed to find the Pareto solution

sets [22]. NSGA-II [23] and MOSA [24] are two meta-heuristics that are commonly used to find the Pareto front solutions in NP-hard multi-objective problems.

Rezaei and Davoodi [25] developed two multi-objective mixed-integer non-linear models for solving multi-period LSPs involving multiple products and multiple suppliers. The first model represents situations where shortages are not allowed while the second one represents situations where the total demand during the stock-out period is backordered. They considered three conflicting objectives (i.e., cost, quality, and service level) and used NSGA-II to find the best Pareto fronts. Karimi-Nasab and Aryanezhad [26] proposed a novel multi-objective model for the production smoothing problem on a single-stage facility where some of the operating times could be determined in a time interval. Karimi-Nasab and Konstantaras [27] presented a new multi-objective production planning model and a random search heuristic with a reasonable computation time.

Mehdizadeh and Tavakkoli-Moghaddam [28] proposed the VDO algorithm which is based on the concept of vibration damping in mechanical vibration. They first utilized the VDO algorithm to solve the parallel machine scheduling problem. In addition, Mehdizadeh et al. [29] proposed a hybrid VDO algorithm to solve the multi-facility stochastic-fuzzy capacitated location–allocation problem. Furthermore, Mousavi et al. [30] developed a special type of the VDO algorithm to solve the capacitated multi-facility location–allocation problem with probabilistic customers' locations and demands. They developed the multi-objective version of VDO as another way to solve the proposed mathematical model. HSA, a music-inspired algorithm, is simple in concept and has just a few parameters. It is easy to implement and has been successfully applied to different problems including the Sudoku puzzle [31], mechanical structure design [32], pipe network optimization [33], inventory models [34], and facility location [35]. However, to the best of our knowledge, it has not been employed to solve NP-hard LSPs. We propose two novel Pareto-based multi-objective meta-heuristic algorithms called MOVDO and MOHSA for solving NP-hard LSPs. We compare the proposed algorithms with two widely known Pareto-based meta-heuristic algorithms called NSGA-II and MOSA to demonstrate the efficiency and effectiveness of the proposed algorithms.

The rest of the paper is organized as follows. In Sect. 2, we present the mathematical details of the mixed-integer-programming formulation proposed in this study. In Sect. 3, we introduce MOVDO, MOHSA, MOSA, and NSGA-II for solving the mixed-integer-programming problems. In Sect. 4, we present a computational, statistical, and graphical comparison of the results for the four Pareto-based meta-heuristic algorithms. Finally, in Sect. 5, we present our conclusions and future research directions.

2 Problem formulation

In this section, we first discuss the problem and its attributes, and then formulate the proposed multi-objective non-linear programming model. The proposed multi-objective non-linear programming model captures various real-world characteristics of multi-item LSP's such as production line equilibrium limitations, capacity limitations, as well as shortages and safety stocks. The model also considers different production methods (i.e., job shop, batch flow, or continuous flow among others) that can be utilized in the manufacturing process and attempts to determine the most suitable method. We consider the following

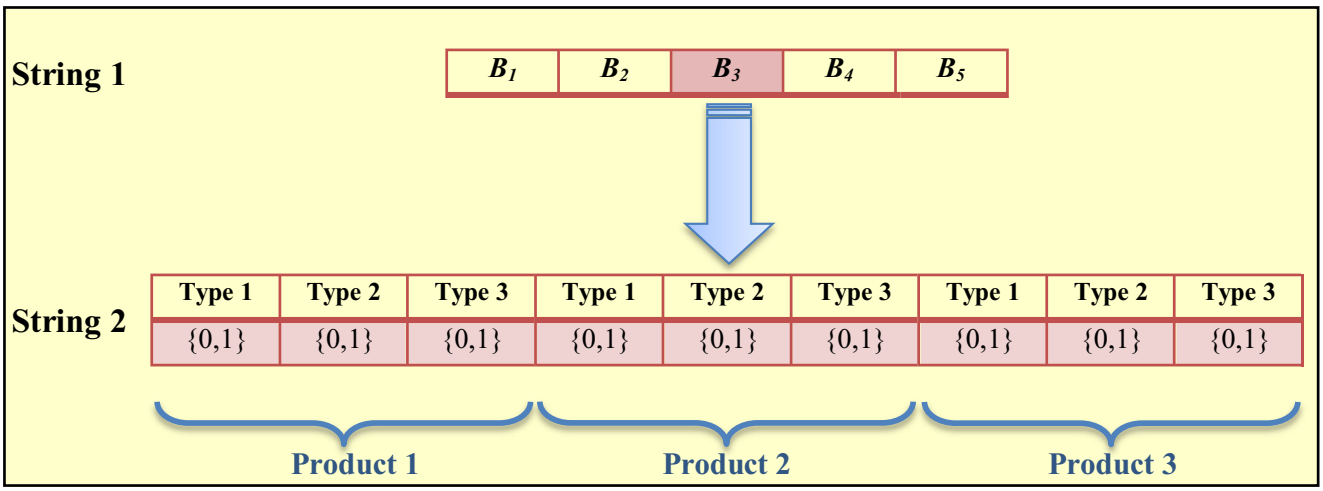

Fig. 1 Solution representation

assumptions, parameters, and decision variables associated with the model:

2.1 Assumptions

- Demand is assumed deterministic.
- Shortage is considered as a backorder and a lost sale, proportionally.
- Safety stocks and shortages on safety stocks are considered.
- Shortage and inventory costs are considered at the end of the planning horizon.
- Storage capacity limitations are considered.
- Raw material resources are capacitated.
- Inventory and shortage quantities are zero at the beginning of the planning horizon.
- Shortage quantity is zero at the end of the planning horizon.

2.2 Parameters

T Number of periods in the planning horizon (t=1, …, T)
N Number of items (i=1, …, N)
J Number of production methods (j=1, …, J)
d_{it} Demand for item i in period t
ξ Percentage of shortages backordered.
π_{it} Unit shortage cost of a lost sale of item i in period t
φ_{it} Unit backorder shortage cost of item i in period t
γ_{it}^{-} Unit safety stock shortage cost of item i in period t
L_{it} Safety stock value of item i in period t
δ_{it} Safety stock variation between two consecutive periods
α_{ijt} Unit production cost of item i for production method j in period t
β_{ijt} Setup cost of item i by production method j in period t
γ_{it}^{+} Unit holding cost of item i in period t
C_t Available capacity in period t
v_{it} Amount of resources necessary to produce item i in period t. Note that, in this model, $v_{i,1,t}=v_{i,2,t}=\ldots=v_{i,J,t}=v_{it}$.
M A large number
w_i Required space for a unit of item i
f_{ij} Lost resource when product i is produced by production method j
F_t Available storage capacity in period t

2.3 Decision variables

x_{ijt} Quantity of item i produced by production method j in period t
y_{ijt} A binary variable equal to 1 if item i is produced by production method j in period t
r_{it} Shortage for item i at period t
s_{it}^{+} Overstock shortage variables for item i in period t
s_{it}^{-} Safety stock shortage variables for item i in period t

2.4 The non-linear integer mathematical formulation

$$\text{Min } Z_1 = \sum_{i=1}^{n}\sum_{t=1}^{T}\Big[\Big(\sum_{j=1}^{J}\left(\alpha_{ijt}\cdot x_{ijt}+\beta_{ijt}\cdot y_{ijt}\right)\Big)+\xi\cdot\varphi_{it}\cdot r_{it} + (1-\xi).\pi_{it}\cdot r_{it}+y_{it}^{+}\cdot s_{it}^{+}+y_{it}^{-}\cdot s_{it}^{-}\Big] \quad (1)$$

$$\text{Min } Z_2 = \sum_{i-1}^{n}\sum_{j=1}^{J}\sum_{t=1}^{T-1}\left(x_{ij,t+1}-x_{ijt}\right)^{2} \quad (2)$$

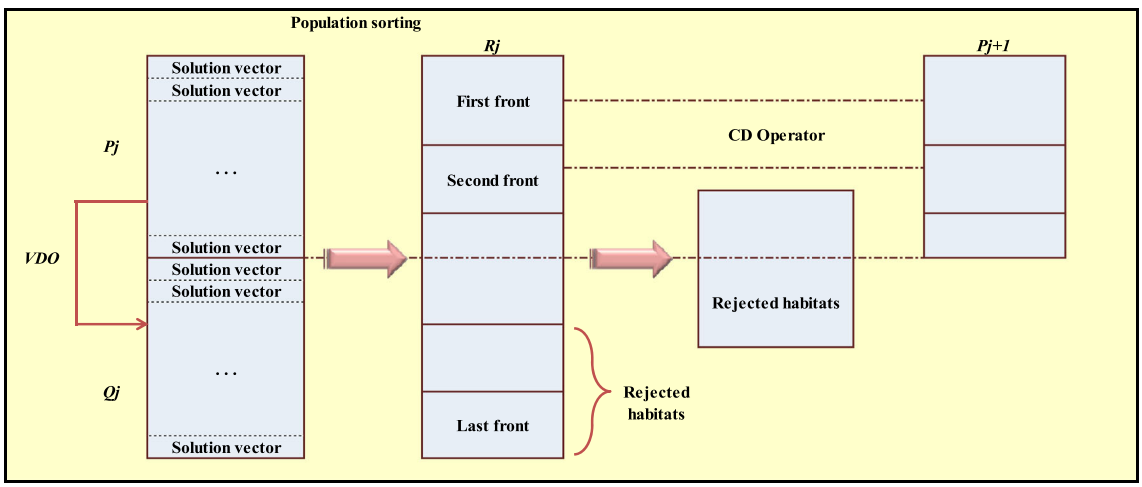

Fig. 2 Evolutionary process of the MOVDO method

Fig. 3 MOVDO pseudo-code

```
Begin;
 Input: nPop (Population Number), γ (Damping Coefficient) and σ (Raleigh Distribution Constant);
 Initialize (X; A; L and t, t=1);
 Evaluate Solutions

 Perform fast non-dominate sorting (FNDS) and calculate ranks
 Calculate crowding distance (CD)
 Sort population according to ranks and CDs

 For j=1: nPop
   P_j = Population
   For i=1:L
       Y = PERTURB(X); {Generate Neighborhood Solution}
       Δ= E(Y) − E(X);
       If Δ≤ 0 Or (1 − e^(−A²/(2σ²)) > Random(0,1))
       Then  X = Y ; {Accept the Movement if Dominated Final Pareto Solution}
       End if

       Update (A and t ;  A=A_0*e^(−γt/2), t = t+1)
       Until (Stop-Criterion)
 End for

 Q_j = New Population
 R_j = P_j • Q_j
 Perform fast non-dominate sorting (FNDS) on R_j and calculate ranks
 Calculate crowding distance (CD) of R_j
 Sort Population according to ranks and CDs on R_j
 Create P_(j+1) as Size as Population Size (Population= P_(j+1))

End for
End
```

$$\text{Min } Z_3 = \sum_{i=1}^{n}\sum_{j=1}^{J}\sum_{t=1}^{T}\left(x_{ijt} - d_{it}\right)^2 \tag{3}$$

s.t.

$$s_{i,t-1}^{+} - s_{i,t-1}^{-} - \xi.r_{i,t-1} + r_{it} + \sum_{j=1}^{J} x_{ijt} = d_{it} + \delta_{it} + s_{it}^{+} - s_{it}^{-} \quad \forall\, i = 1,\ 2,\ \ldots,\ n\ ;\ t = 1,\ 2,\ \ldots,\ T-1 \tag{4}$$

$$s_{i,t-1}^{+} - s_{i,t-1}^{-} - \xi \cdot r_{i,t-1} + \sum_{j=1}^{J} x_{ijt} = d_{it} + \delta_{it} + s_{it}^{+} - s_{it}^{-} \quad \forall\, i = 1,\ 2,\ \ldots,\ n\ ;\ t = T \tag{5}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{J}\left(v_{it} \cdot x_{ijt} + f_{ij} \cdot y_{ijt}\right) \le C_t \quad \forall\, t = 1,\ 2,\ \ldots,\ T \tag{6}$$

$$x_{ijt} \le M \cdot y_{ijt} \quad \forall\, i = 1,\ 2,\ \ldots,\ n\ ;\ j = 1,\ 2,\ \ldots,\ J\ ;\ t = 1,\ 2,\ \ldots,\ T \tag{7}$$

$$r_{it} \le d_{it} \quad \forall\, i = 1,\ 2,\ \ldots,\ n\ ;\ t = 1,\ 2,\ \ldots,\ T \tag{8}$$

$$s_{it}^{-} \le L_{it} \quad \forall\, i = 1,\ 2,\ \ldots,\ n\ ;\ t = 1,\ 2,\ \ldots,\ T \tag{9}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{J} w_i \cdot x_{ijt} \le F_t \quad ;\ \forall\, t = 1,\ 2,\ \ldots,\ T \tag{10}$$

$$x_{ijt}, r_{it}, s_{it}^{+}, s_{it}^{-} \ge 0\ ;\ y_{ijt} \in \{0,1\} \quad \forall\, i = 1,\ 2,\ \ldots,\ n\ ;\ j = 1,\ 2,\ \ldots,\ J\ ;\ t = 1,\ 2,\ \ldots,\ T \tag{11}$$

- Objective (1) is used to minimize the total cost, including unit production costs with different production methods, inventory costs, overtime costs, shortage costs, and setup costs.

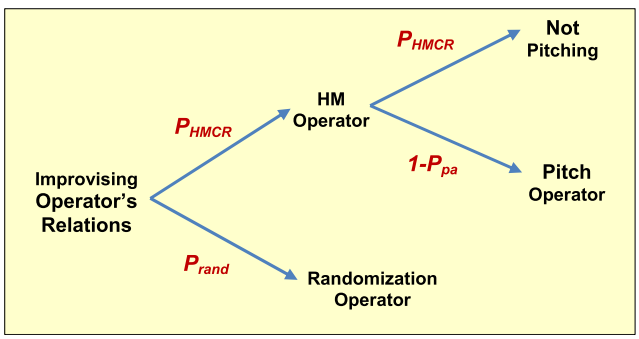

Fig. 4 Relationship between different HSA probabilities

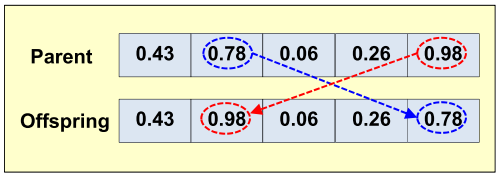

Fig. 5 Pitch-adjusting operator example

- Objective (2) is used to level the production volume in different production periods.
- Objective (3) is used to ensure that the model produces results close to the just-in-time levels.
- Constraints (4) are used to control the inventory balances throughout the planning horizon.
- Constraints (5) are used to satisfy the condition that shortages are not allowed.
- Constraints (6) are used to satisfy the condition that overall consumption may not exceed the available capacity.
- Constraints (7) are used to satisfy the condition that the quantity produced must not exceed a maximum production level M_{it} which is assumed to be the minimum between the total demand requirement for item i on section (t,T) of the planning horizon and the highest quantity of item i that could be produced considering the capacity constraints. M_{it} is then equal to:

$$\text{Min}\left\{\sum_{t'=t}^{T} d_{it'}, (C_t - \sum_{j=1}^{J} f_{ij})/v_{it}\right\}$$

- Constraints (8) and (9) are used to satisfy the upper bounds on the demand shortage and the safety stock shortage for item i in period t, respectively,
- Constraints (10) are used to satisfy the storage space limitation.
- Constraints (11) are used to specify that the variable domains for x_{ijt}, r_{it}, s_{it}^{+}, and s_{it}^{-} are non-negative and that y_{ijt} is a binary variable.

3 Pareto-based meta-heuristics

In this section, four Pareto-based meta-heuristic algorithms called MOVDO, MOHSA, MOSA, and NSGA-II are introduced for solving the mixed-integer-programming formulation proposed in this study. Consider a multi-objective model with a set of conflicting objectives $f(\vec{x}) = [f_1(\vec{x}), \ldots, f_m(\vec{x})]$ subject to $g_i(\vec{x}) \le 0$, $i = 1, 2, \ldots, c$, $\vec{x} \in X$, where $\vec{x}$ denotes n-dimensional vectors that can have real, integer, or even Boolean values and X is the feasible region. Then, for a minimization model, we say solution $\vec{a}$ dominates solution $\vec{b}$ $\left(\vec{a}, \vec{b} \in X\right)$ if:

1. $f_i(\vec{a}) \le f_i\left(\vec{b}\right)$, $\forall i = 1, 2, \ldots, m$; and
2. $\exists i \in \{1, 2, \ldots, m\} : f_i(\vec{a}) < f_i\left(\vec{b}\right)$

Furthermore, Pareto solutions or Pareto fronts are a set of solutions that cannot dominate each other. A suitable Pareto front has two features: (1) good convergence and (2) good solution diversity. A wide range of multi-objective Pareto-based algorithms are developed to achieve the Pareto-

Fig. 6 HSA pseudo-code

Parameter setting: $\boldsymbol{P_{HMCR}}$, $\boldsymbol{P_{pa}}$, $\boldsymbol{P_{rand}}$, Pop. Size, Outer Loop Num., Inner loop Num.
Initialization: Generating harmonies as size as HM
Evaluation: Evaluate harmonies
Sort the memory according to the harmonies value increasingly

For *i*=1: Out. Loop Num.
  For *j*=1: Inner loop Num.
    Generate $\boldsymbol{Rand} \in [\boldsymbol{0}, \boldsymbol{1}]$
    If *Rand*< $\boldsymbol{HMCR}$
      $\boldsymbol{H}$ = Choose a solution randomly
      Generate $\boldsymbol{Rand} \in [\boldsymbol{0}, \boldsymbol{1}]$
        If *Rand*< $\boldsymbol{P_{pa}}$
          $\boldsymbol{H}$ =Pitch the solution $\boldsymbol{H}$
        Else
          The solution keeps unchanged
        End if
    Else
      $\boldsymbol{H}$ = Improvise a solution randomly
    End if

    Update the HM (accept $\boldsymbol{H}$, if it is better than the worst solution of HM)
  End for

  Keep CPU time and best value of harmonies of HM
End for

optimal front. This front is expected to have the most convergence and the highest diversity [23].

3.1 The MOVDO algorithm

VDO is a meta-heuristic algorithm based on the concept of vibration damping in mechanical vibration improvisation in music [28]. In this paper, a multi-objective version of the VDO algorithm is applied to the production planning problems. The details of this algorithm are explained in the following sub-sections.

3.1.1 Solution representation

In this paper, we focus on the single-strand (A) solution type, whose length is equal to the total number of periods. Each part of the string itself contains a strand (B_t) whose length equals the total number of different production methods. On the other hand, the components of string B_t can define the production plan, depending on the type of decision variable. A schematic solution set for a specific problem with three products and three different production methods for each of the products in five periods is shown in Fig. 1.

However, since some constraints are likely to be violated, they are penalized using the method given in Yeniay and Ankare [36]. In other words, infeasible solutions are found using Eq. (12).

$$P(x) = M \times \left(\left(\frac{g(x)}{b} \right) - 1 \geq 0 \right) \tag{12}$$

where M, $g(x)$, and $P(x)$ represent a large number, the constraint under consideration, and the penalty value, respectively. This equation ensures that more violations receive bigger penalties for a constraint such as $g(x) \leq b$. Moreover, penalty values are considered for all three objective functions according to an additive function given in Eq. (13).

$$F(x) = \begin{cases} f(x) & ; \quad x \in \text{feasible region} \\ f(x) + P(x) & ; \quad x \notin \text{feasible region} \end{cases} \tag{13}$$

where $f(x)$ is the objective function value of chromosome x.

3.1.2 MOVDO main loop

In vibration theory, the concept of vibration is based upon the theory of oscillation. If the damping is small, it has very little influence on the natural frequencies of the system, and hence, the calculation for the natural frequencies is approximated based on the case of no damping. In the VDO algorithm, at high amplitudes, the scope of a solution becomes larger and the probability of obtaining a new solution increases. Therefore, when the amplitude is reduced, the probability of obtaining a new solution decreases, and then the system stops from the amplitude state [28, 30].

Using the analogy between an optimization problem and the vibration damping process, the states of the oscillation system represent feasible solutions of the optimization problem, the energies of the states correspond to the objective function value computed at those solutions, the minimum energy state corresponds to the optimal solution to the problem, and rapid quenching can be viewed as local optimization.

The VDO algorithm starts by generating random solutions in the search space. The algorithm parameters (i.e., initial amplitude (A_0), max of iteration at each amplitude (L), damping coefficient (γ), and standard deviation (σ)) are then initialized. The solutions are then evaluated by means of the objective function value (OFV). The algorithm contains two main loops. The first loop generates a solution randomly using the Microsoft Excel linear programming Solver add-in and then a

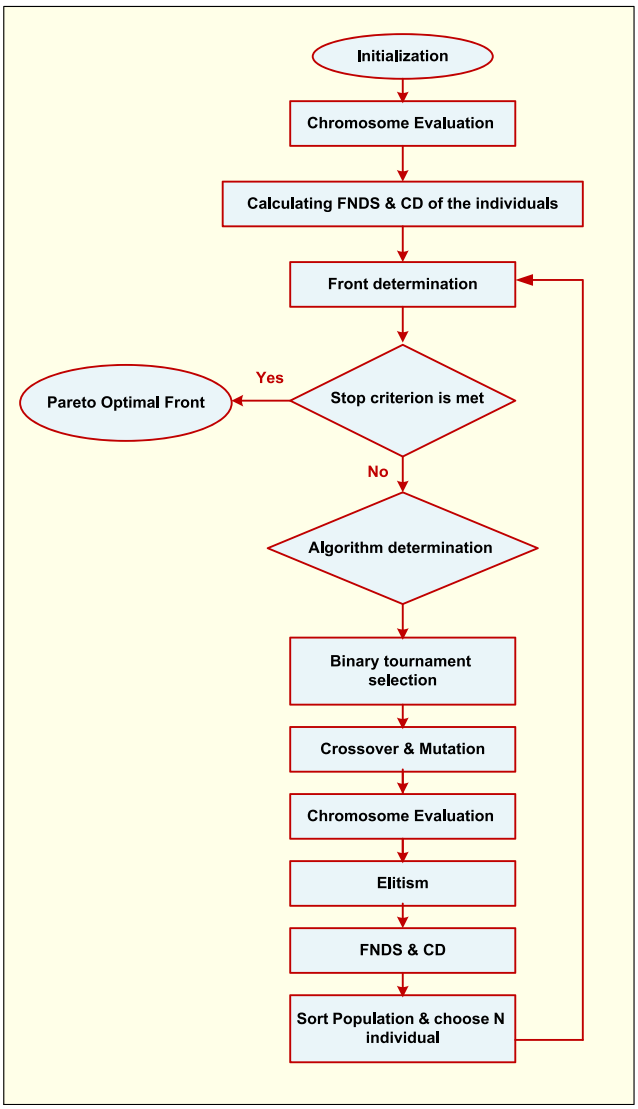

Fig. 7 NSGA-II flowchart

new solution is obtained and chosen as the best one using a neighborhood structure. However, similar to the SA algorithm, the solution with a lower OFV can be selected with regards to the Rayleigh distribution function. In fact, the new solution is accepted if Δ= OFV (new solution)−OFV (current solution)< 0. Otherwise, if $\Delta > 0$, we generate a random number r between (0, 1). The current solution is selected with respect to the following criteria:

$$r < 1-\exp\left(-\frac{A^2}{2\sigma^2}\right) \tag{14}$$

The second loop adjusts the amplitude, which is used for reducing the amplitude at each iteration. The algorithm is finished when the stopping criterion is met.

$$A_t = A_0 \exp\left(-\frac{\gamma t}{2}\right) \tag{15}$$

After a brief illustration of the VDO algorithm, we introduce the first proposal for applying a multi-objective version of the VDO algorithm called MOVDO to solve and manage Pareto-optimal solutions. For this purpose, we apply two main concepts of multi-objective meta-heuristics; namely, fast non-dominated sorting (FNDS) and the crowding distance (CD), to compare the solutions. In FNDS, R initial populations are compared and sorted. Initially, all chromosomes in the first non-dominated front are found. Since all objective functions in the mathematical model are to be minimized, the chromosomes are chosen using the concept of domination. Then, in order to find the chromosomes in the next non-dominated front, the solutions of the previous fronts are disregarded temporarily. This procedure is repeated until all solutions are set into fronts.

After sorting the populations in different fronts, a CD measure is defined to evaluate the solution fronts of the populations in terms of the relative density of individual solutions [23]. Consider Z and f_k; k=1,2,…,M as the number of non-dominated solutions in a particular front (F) and the objective functions, respectively. In addition, let $d_i^{'}$ and $d_j^{'}$ be the value of the CD on the solution i and j, respectively. Then, the CD is obtained using the following steps:

(a) Set $d_i^{'}$=0 for i=1, 2, …, Z,

(b) Sort all objective functions f_k ; k=1,2,…,M in ascending order,

Fig. 8 MOSA pseudo-code

```
Parameter setting: Tf, T0, nGen, β, frontmax
Initialization: Generate initial solutions
Evaluation: Evaluate initial solutions
Perform non-dominate sorting and calculate ranks
Calculate crowding distance (CD)
Sort population according to ranks and CDs

Pt=population
For  it=1: num.it
    if T< Tf
    break
   end
    for i=1:popsize
        St(i)= perform neighborhood structure on the solution i of the population based on swap strategy
    end

   Perform non-dominate sorting and calculate ranks (St)
   Calculate crowding distance (CD) (St)
   Sort population according to ranks and CDs (St)

   for i=1:popsize
      if ~Dominates (Pt(i), St (i))
          Qt (i)= St (i)
     else
         delta=Cost Pt (i)- Cost  St (i)
         p=exp(-delta/T(it))
         if rand<p
              Qt  (i)= St (i)
         end
      end
   end

   Rt = Pt ∪Qt
   Perform non-dominate sorting and calculate ranks (Rt)
   Calculate crowding distance (CD) (Rt)
   Sort population according to ranks and CDs (Rt)

   if  size( Rt )> frontmax
       Pt =Select  frontmax number of the solution
       non-dominate sorting and calculate ranks  ( Pt )
       Calculate crowding distance  ( Pt)
   else
       Pt =  Rt
   end
Update T
End
```

Table 1 Algorithm parameter values and tuning procedures

Multi-objective algorithms	Algorithm parameters	Parameter descriptions	Optimum values
MOVDO	*nPop*	Number of population	5
	A_0	Initial amplitude	6
	L	Max of iteration at each amplitude	40
	σ	Standard deviation	1.5
	γ	Damping coefficient	0.05
MOHSA	P_{hmcr}	Harmony memory considering rate	0.75
	P_{pa}	Pitch-adjusting probability	0.3
	Outloop	Outer loop	50
	In. loop	Inner loop	100
	nPop	Number of population	50
MOSA	T_0	Initial temperature	500
	Popsize	Number of population	5
	Ngen	Maximum number of generation	500
	β	Temperature reduction rate	0.99
NSGA-II	*nPop*	Number of population	25
	P_c	Crossover probability	0.6
	P_m	Mutation probability	0.01
	Ngen	Maximum number of generation	100

(c) The CD for the end solutions in each front ($d_1^{'}$ and $d_2^{'}$) are $d_1^{'} = d_z^{'} \rightarrow \infty$, and

(d) The crowding distances for $d_j^{'}; j = 2, 3, \ldots, (Z-1)$ are $d_j^{'} = d_j^{'} + \left(f_{k_{j+1}} - f_{k_{j-1}}\right)$.

To select individuals for the next generation, the crowded tournament selection operator "≻" is applied [22] according to the following steps:

Step 1. Randomly choose n individuals from the population.

Step 2. A non-dominated rank order is obtained for each individual and the CDs of the solutions having equal non-dominated rank are calculated.

Step 3. Select the solutions with the least rank order. Moreover, if more than one individual share the lowest ranking, the individual with the highest CD should be selected.

In other words, the comparison criterion in the MOVDO algorithm solutions can be written as follows: if $r_x < r_y$ or $r_x = r_y$ and $d_x^{'} > d_y^{'}$, then $x \succ y$ where r_x and r_y are the ranks and d_x and d_y are the CDs. We assume a polynomial neighborhood structure for the selected chromosome.

After applying the aforementioned concepts and operators, the populations of the parents and the offspring are combined to ensure the elitism. Since the combined population size is naturally higher than the original population size N, we perform non-domination sorting once more. Chromosomes with higher ranks are selected and added to the populations until the population size becomes N. The last front also consists of the population based on the crowding distance. The algorithm stops when a predetermined number of iterations (or any stopping criteria) is reached and only feasible solutions are kept in the final Pareto front.

3.1.3 Evolution process of MOVDO

The process is started by initializing the initial population of the solution vectors P_j. Then, the new operators are

Table 2 Input parameters of the generated test problems

Test problem number	Number of items (N)	Number of production methods (J)	Number of periods (T)
1	2	2	3
2	3	2	5
3	3	3	5
4	5	2	6
5	5	3	6
6	5	2	12
7	10	2	12
8	10	3	5
9	12	2	12
10	15	2	5
11	15	3	12
12	20	3	12

Table 3 Computational results of the multi-objective metrics for all the algorithms

Metric	Problem No.	MOVDO	MOHSA	NSGA-II	MOSA
NOS	1	12	7	11	3
	2	8	10	12	4
	3	10	11	8	6
	4	4	4	8	3
	5	5	3	7	5
	6	3	4	6	9
	7	11	8	7	3
	8	6	7	5	1
	9	5	12	7	5
	10	8	12	9	6
	11	6	10	12	3
	12	4	4	15	2
Spacing	1	142×10^6	8980×10^6	995×10^6	178×10^6
	2	454×10^6	773×10^6	6540×10^6	121×10^6
	3	423×10^6	243×10^6	252×10^6	1420×10^6
	4	895×10^6	142×10^6	8540×10^6	465×10^6
	5	653×10^6	957×10^6	348×10^6	8590×10^6
	6	137×10^6	6520×10^6	4130×10^6	123×10^6
	7	7450×10^6	8960×10^6	1180×10^6	1120×10^6
	8	543×10^6	535×10^6	5340×10^6	5550×10^6
	9	9760×10^6	1320×10^6	1420×10^6	2230×10^6
	10	$25{,}300\times10^6$	1520×10^6	2540×10^6	$27{,}900\times10^6$
	11	5430×10^6	$32{,}500\times10^6$	9870×10^6	9680×10^6
	12	$96{,}800\times10^6$	$59{,}700\times10^6$	$59{,}700\times10^6$	$79{,}700\times10^6$
MOCV	1	0.016	0.000	0.003	0.001
	2	0.000	0.013	0.022	0.029
	3	0.132	0.222	0.007	0.826
	4	0.010	0.012	0.200	0.025
	5	0.485	0.069	0.075	0.007
	6	0.177	0.281	0.336	0.005
	7	0.004	0.274	0.213	0.056
	8	0.011	0.345	0.100	0.168
	9	0.023	0.002	0.066	0.051
	10	1.030	0.729	0.632	0.473
	11	0.052	0.068	0.050	0.104
	12	0.283	0.073	0.415	0.746
CPU Time	1	26.124	27.302	29.185	26.968
	2	28.265	27.281	32.413	28.876
	3	30.543	30.235	36.568	29.368
	4	33.997	36.765	39.865	34.988
	5	28.815	28.815	27.084	28.815
	6	32.769	33.676	35.596	30.113
	7	29.382	29.382	34.987	30.768
	8	41.988	42.877	49.986	42.986
	9	52.235	55.910	65.097	56.908
	10	75.097	76.987	87.758	79.098
	11	110.173	114.834	138.971	111.831
	12	183.864	192.073	248.974	191.891

implemented on P_j to create a new population Q_j. The combination of P_j and Q_j creates R_j in order to maintain elitism in the algorithm. In this step, the vectors R_j are sorted in several fronts based on FNDS and CD. Using the proposed selection method, a population of the next iteration P_{j+1} is chosen to have a predetermined size. Figure 2 illustrates the evolution process of the proposed MOVDO, and Fig. 3 presents the pseudo-code for the MOVDO algorithm based on the basic operators of a VDO algorithm and the described multi-objective operators.

3.2 The MOHSA

In addition to the proposed MOVDO method, we also propose a multi-objective evolutionary algorithm for lot-sizing called MOHSA. The main difference between MOHSA and MOVDO is the evolution process of the algorithms from P_t to Q_t. The evolution for the MOVDO method is depicted in Fig. 2. The evolution process of a HSA is used in MOHSA. Accordingly, after generating or modifying populations using single-objective operators of the algorithms (HSA or VDO), the population is analyzed using a multi-objective approach in a similar fashion for all the algorithms.

The objective function in HSA is interpreted as harmony and the esthetic judgment of the player helps him/her to find a pleasing harmony. Using this algorithm, the qualitative improvisation process corresponds to a quantitative optimization process. Similarly, when a musician improvises with an instrument, he or she faces three possible options: (I) playing from his/her memory (with probability P_{HMCR}), (II) adjusting the pitches slightly (with probability P_{pa}), and (III) composing randomly (with probability P_{rand}). These options are formalized into three quantitative operators in HSA called harmony memory, pitch adjusting, and randomization [37]. Consequently, the improvising process of the HSA is the combination of these three operators. Accordingly, the main steps of the HSA are explained in the next sub-sections. To apply the improvising process in different iterations, a random solution is selected first and then one/two operator(s) of the HSA (based on their probabilities) is/are used to improvise the selected solution. After improvising a new solution, the HSM is updated by replacing the worse solution with the new solution. Interested readers should refer to Geem et al. [33], Geem [31], and Rahmati et al. [35] for additional information. Figure 4 presents a schematic view of the relationship between different HSA probabilities.

We should note that the operators are designed similarly to minimize the difference between the performance measures of the algorithms using the different operators. Thus, the pitch-adjusting operator of MOHSA is designed similar to the neighborhood operator of MOVDO as presented in Fig. 5 [35, 37]. We also present the pseudo-code version of the HSA in Fig. 6.

Next, two well-developed Pareto-based multi-objective evolutionary algorithms called NSGA-II and MOSA are applied to demonstrate the performance of the proposed MOVDO and MOHSA methods.

3.3 The NSGA-II

The main difference between the NSGA-II method and the MOVDO and the MOHSA methods is the evolutionary process of the algorithm. The NSGA-II method follows the main loop of the genetic algorithm. Moreover, in NSGA-II, the binary tournament selection strategy is applied; while, in MOVDO and MOHSA, we implement a roulette wheel selection operator. The mutation operator is also designed similar to the neighborhood structure of the MOHSA and MOVDO methods using the swap strategy. The pitch-adjusting operator in MOHSA is similar to the mutation operator in the genetic algorithm. In addition, one of the mutation strategies is the swap strategy where two genes are selected and then replace each other. The crossover operator is also considered as a continuous crossover operator [38].

Table 4 Analysis of variance results for the multi-objective metrics for all the algorithms

Matric	Source*	Algorithms	Error	Total
NOS	*df*	3	44	47
	SS	145.56	366.92	512.48
	MS	48.52	8.34	–
	F	5.82	–	–
	P	0.002	–	–
Spacing	*df*	3	44	47
	SS	104×10^{18}	$20{,}700\times10^{18}$	$20{,}800\times10^{18}$
	MS	34.7×10^{18}	471×10^{18}	–
	F	0.07	–	–
	P	0.974	–	–
MOCV	*df*	3	44	47
	SS	0.0083	2.9381	2.9464
	MS	0.0028	0.0668	–
	F	0.04	–	–
	P	0.989	–	–
CPU time	*df*	3	44	47
	SS	1236	12,5706	126,941
	MS	412	2857	–
	F	0.14	–	–
	P	0.933	–	–

df degree of freedom, SS sum of square error, *MS* mean of square error, *F* *F* test, *P* *P* value

The flowchart given in Fig. 7 presents a graphical view of the NSGA-II framework.

3.4 The MOSA

Simulated annealing (SA), introduced by Kirkpatrick et al. [39] is another popular search algorithm. SA utilizes the principles of statistical mechanics with regards to the behavior of a large number of atoms at low temperatures, for finding minimal cost solutions to large optimization problems by minimizing the associated energy. Interested authors should refer to Ulungu et al. [24] and Bandyopadhyay et al. [40] for more details on the MOSA method. The neighborhood structure of the MOSA method is designed similar to the MOVDO and the MOHSA methods. Figure 8 presents the Pseudo-code for the MOSA method.

In the next section, we generate a series of problems to evaluate and demonstrate the applicability and performance of the proposed models and algorithms. We then compare the results using standard multi-objective optimization metrics.

4 Computational results and comparisons

We used four different multi-objective Pareto-based metaheuristic algorithms to solve the proposed MICLSP. The algorithms were calibrated based on the common statistical approaches of response surface methodology (RSM) [41] and the Taguchi method [30, 35]. The algorithm parameters are presented in Table 1 along with their descriptions and optimum values.

We applied three standard metrics including the number of solutions (NOS), spacing, and computational time (CPU Time) to evaluate the performance of the proposed MOHSA and MOVDO methods. We also consider the multi-objective coefficient of variation (MOCV) method recently introduced by Rahmati et al. [35]. These metrics are given below:

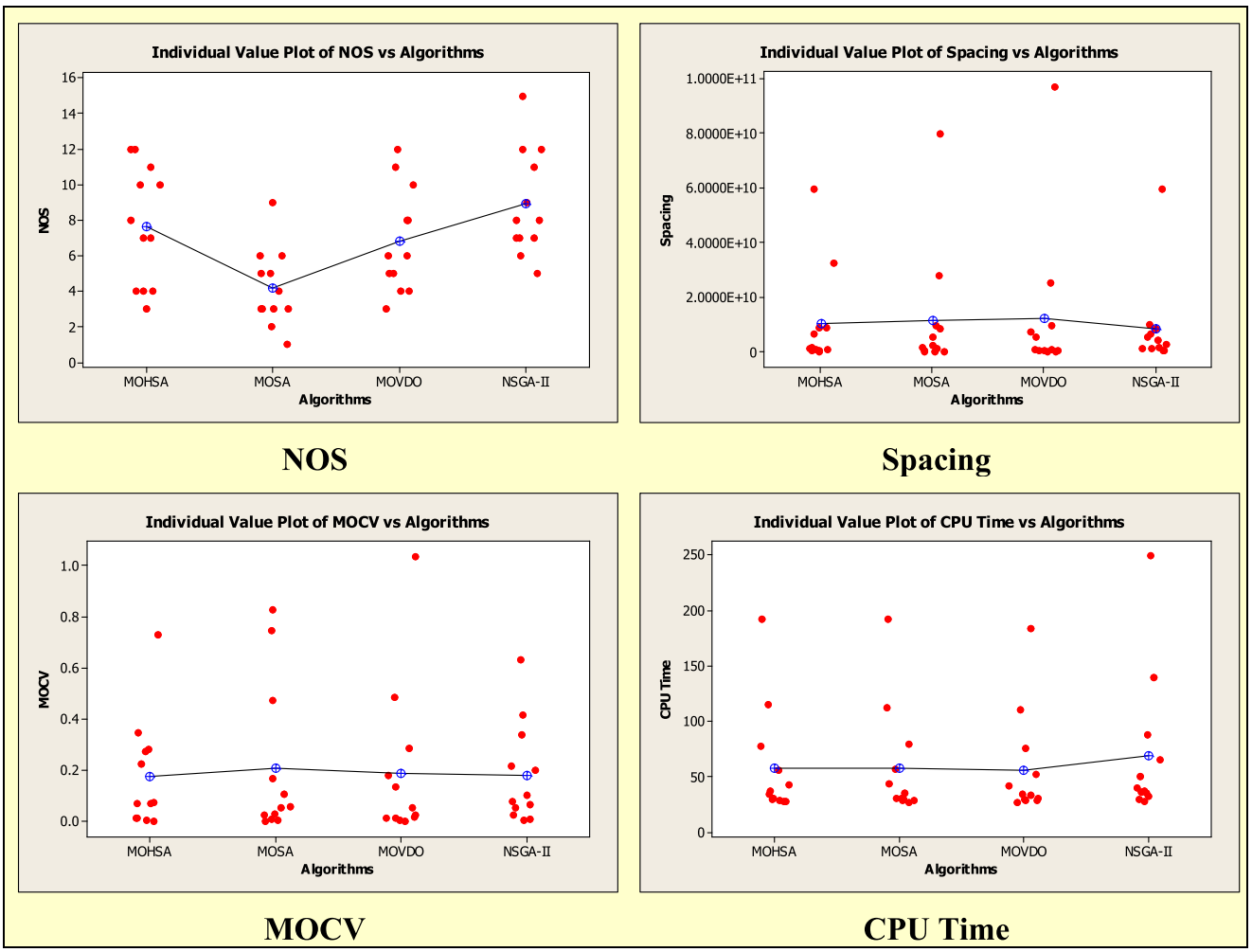

Fig. 9 Individual plot of the multi-objective metrics for all algorithms

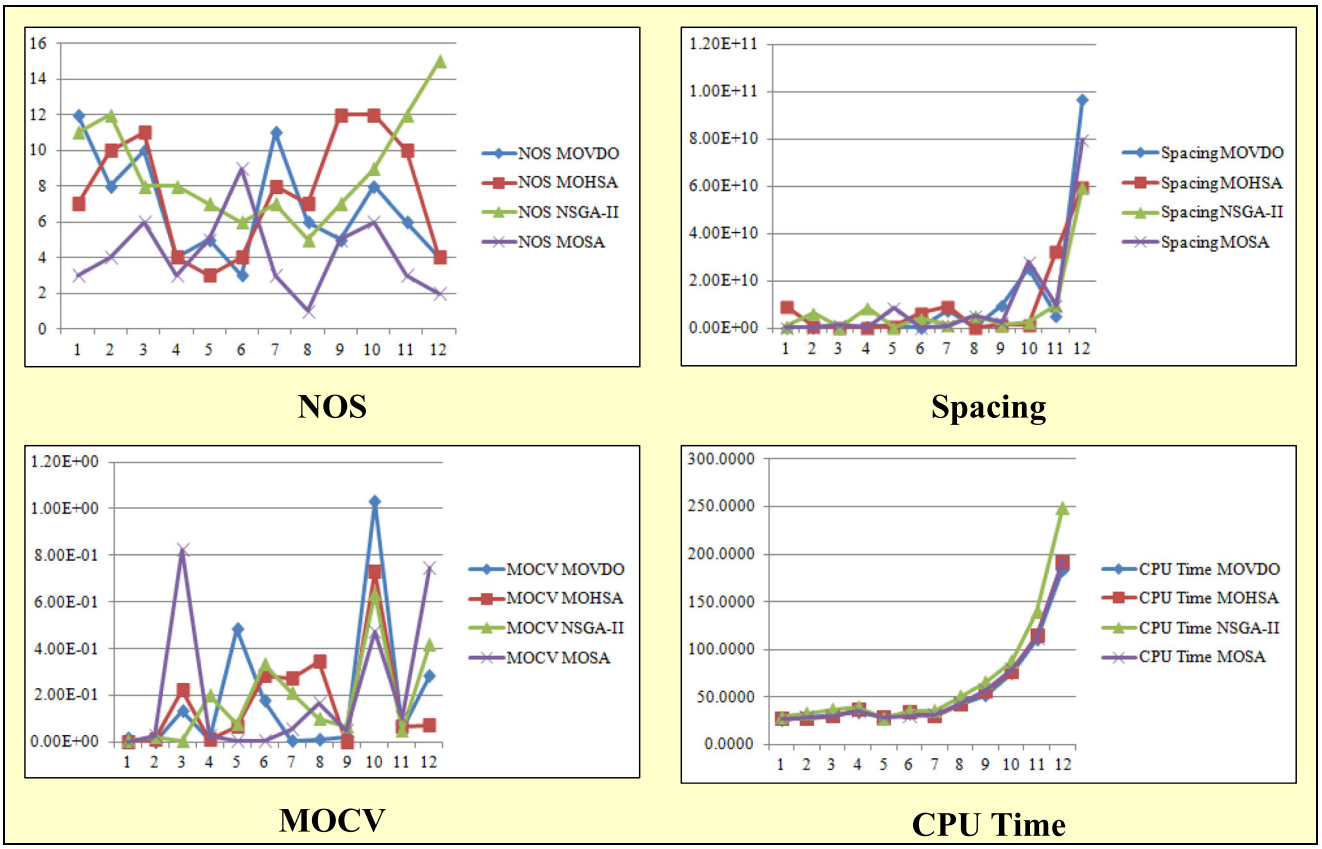

Fig. 10 Graphical comparison of the algorithms

1. Number of solutions (NOS): measures the number of Pareto solutions in the Pareto-optimal front in which the larger values are preferred to the smaller values [42].
2. Spacing**:** measures the standard deviation of the distances among the solutions in the Pareto front in which the smaller values are preferred to the larger values [42].
3. Computational time (CPU): measures the CPU time for running the algorithms to reach near optimum solutions (s).
4. Multi-objective coefficient of variation (MOCV): measures the convergence and diversity of the Pareto solutions [35].

The proposed algorithms were coded in a MATLAB software [43] environment, and the experiments were performed on a 2-GHz laptop with 4 Gb RAM, to estimate the response functions.

We conducted a series of experiments with 12 problems to evaluate and compare the performance of the four algorithms as shown in Table 2. As indicated in the parameter descriptions, N, J, and T are the number of items, number of production methods, and number of periods in the planning horizon, respectively.

We solved each problem three times under different random environments to eliminate any potential uncertainties. The mean value of the three trials was used as the final solution for each problem in Table 3. Larger mean values are more desirable for the NOS metric and smaller mean values are more desirable for the spacing, MOCV, and CPU time metrics.

Next, we used analysis of variance (ANOVA; with a 95 % confidence level) to compare the results of the four algorithms in Table 4. As shown in this table, $\alpha \geq p$ value only for the NOS metric in which case the null hypothesis is rejected. The results of the ANOVA test show that the algorithms have significant differences (at the 95 % confidence level) with regards to the NOS metric.

We then plot the individual values for the cases with a significant difference between the four blocks as shown in

Table 5 The overall rankings of the four algorithms according to the four comparison metrics

Metric	Algorithms			
	MOVDO	MOHSA	NSGA-II	MOSA
NOS	3	2	1	4
Spacing	4	2	1	3
MOCV	3	1	2	4
CPU time	1	3	4	2

Fig. 9. As shown in the three blocks of Spacing, MOCV and CPU Time, the four algorithms perform very similarly with no significant differences. However, there is a significant difference among the four algorithms with respect to the NOS metric. As shown in this figure, the NSGA-II method not only generates more Pareto (efficient) solutions, but also performs better than the other three algorithms in terms of the NOS metric.

Finally, we plot and compare all four metrics in Fig. 10. The NOS and Spacing graphs show a better performance for the NSGA-II algorithm while the MOCV and CPU Time graphs show a better performance for our proposed algorithms.

In summary, Table 5 presents the overall rankings of the four algorithms according to the four comparison metrics. As shown in this table, with regards to the NOS metric, the rankings of the algorithms are as follows: NSGA-II>MOHSA>MOVDO>MOSA. With regard to spacing metric, the rankings of the algorithms are as follows: NSGA-II>MOHSA>MOSA>MOVDO. With regards to the MOCV metric, the rankings of the algorithms are as follows: MOHSA>NSGA-II>MOVDO>MOSA. Finally, with regards to the CPU time metric, the rankings of the algorithms are as follows: MOVDO>MOSA>MOHSA>NSGA-II.

5 Conclusions and future research directions

In this paper, we proposed two models for solving multi-objective multi-item capacitated LSPs with setup times, safety stock shortage costs, and demand shortage costs. The objectives are to minimize the total cost, level the production volume in different production periods, and maintain the production level as close as possible to the just-in-time level. We proposed two Pareto-based multi-objective meta-heuristic algorithms (i.e., MOVDO and MOHSA). We used two well-known multi-objective evolutionary algorithms (NSGA-II and MOSA) to evaluate the performance of the proposed MOVDO and MOHSA. The statistical and graphical results showed the robustness and comparability of the proposed MOVDO and MOHSA in terms of the *MOCV*, *CPU time*, and *spacing* metrics. In addition, the NSGA-II was shown to work more efficiently in terms of the *NOS* metric. For future research, we propose implementing the Pareto-based multi-objective meta-heuristic algorithms developed in this study in a fuzzy environment as a fuzzy multi-objective mathematical programming model.

Acknowledgments The authors would like to thank the anonymous reviewers and the editor for their insightful comments and suggestions.

References

1. Trigeiro WW, Thomas JL, McCain JO (1989) Capacitated lot sizing with setups. Manag Sci 35(3):141–161
2. Chen WH, Thizy JM (1990) Analysis of relaxations for the multi-item capacitated lot-sizing problem. Ann Oper Res 26:29–72
3. Karimi B, Fatemi Ghomi SMT, Wilsonb JM (2003) The capacitated lot sizing problem: a review of models and algorithms. Omega 31: 365–378
4. Robinson P, Narayanan A, Sahin F (2009) Coordinated deterministic dynamic demand lot-sizing problem: a review of models and algorithms. Omega 37(1):3–15
5. Wagner HM, Whitin TM (1958) A dynamic version of the economic lot size model. Manag Sci 5(1):89–96
6. Manne AS (1958) Programming of economic lot-sizes. Manag Sci 4:115–135
7. Brahimi N, Dauzere-Peres S, Najid NM, Nordli A (2006) Single item lot sizing problems. Eur J Oper Res 168(1):1–16
8. Kazan O, Nagi R, Rump CM (2000) New lot-sizing formulations for less nervous production schedules. Comput Oper Res 27:1325–1345
9. Silver E, Mael H (1973) A heuristic selecting lot size requirements for the case of deterministic time varying demand rate and discrete opportunities for replenishment. Prod Invent Manag 14:64–74
10. Loparic M, Pochet Y, Wolsey LA (2001) The uncapacitated lot-sizing problem with sales and safety stocks. Math Program 89(487–504):2001
11. Aksen D, Altinkemer K, Chand S (2003) The single-item lot-sizing problem with immediate lost sales. Eur J Oper Res 147(558–566):2003
12. Absi N, Kedad-Sidhoum S (2007) MIP-based heuristics for multi item capacitated lot-sizing problem with setup times and shortage costs. RAIRO Oper Res 41(2):171–192
13. Absi N, Kedad-Sidhoum S (2008) The multi-item capacitated lot-sizing problem with setup times and shortage costs. Eur J Oper Res 185:1351–1374
14. Absi N, Kedad-Sidhoum S (2009) The multi-item capacitated lot-sizing problem with safety stocks and demand shortage costs. Comput Oper Res 36:2926–2936
15. Absi A, Detienne B, Dauzere-Peres S (2013) Heuristics for the multi-item capacitated lot-sizing problem with lost sales. Comput Oper Res 40(1):264–272
16. Tempelmeier H, Derstroff M (1996) A lagrangean-based heuristic for dynamic multilevel multi item constrained lot sizing with setup times. Manag Sci 42:738–757
17. Sural H, Denize M, VanWassenhove LN (2009) Lagrangean relaxation based heuristics for lot-sizing with setup times. Eur J Oper Res 194(1):51–60
18. Berretta R, Rodrigues LF (2004) A memetic algorithm for a multi-stage capacitated lot-sizing problem. Int J Prod Econ 87:67–81
19. Han Y, Tang J, Kaku I, Mu L (2009) Solving uncapacitated multilevel lot-sizing problems using a particle swarm optimization with flexible inertial weight. Comput Math Appl 57: 1748–1755
20. Tang O (2004) Simulated annealing in lot sizing problems. Int J Prod Econ 88:173–181
21. Xiao Y, Kaku I, Zhao Q, Zhang R (2011) A variable neighborhood search based approach for uncapacitated multilevel lot-sizing problems. Comput Ind Eng 60(2):218–227
22. Coello CA, Lamont GB, Van Veldhuizen DA (2007) Evolutionary algorithms for solving multiobjective problems, 2nd edn. Springer, Berlin
23. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6:182–197

24. Ulungu EL, Teghaem J, Fortemps P, Tuyttens D (1999) MOSA method: a tool for solving multiobjective combinatorial decision problems. J Multi-Criteria Decis Anal 8:221–236
25. Rezaei J, Davoodi M (2011) Multi-objective models for lot-sizing with supplier selection. Int J Prod Econ 130:77–86
26. Karimi-Nasab M, Aryanezhad MB (2011) A multi-objective production smoothing model with compressible operating times. Appl Math Model 35:3596–3610
27. Karimi-Nasab M, Konstantaras L (2012) A random search heuristic for a multi-objective production planning. Comput Ind Eng 62: 479–490
28. Mehdizadeh E, Tavakkoli-Moghaddam R (2008) Vibration damping optimization. Proceedings of the International Conference of Operations Research and Global Business, Germany, 3–5 September
29. Mehdizadeh E, Tavarroth MR, Hajipour V (2011) A new hybrid algorithm to optimize stochastic-fuzzy capacitated multi-facility location–allocation problem. J Optim Ind Eng 7:71–80
30. Mousavi SM, Akhavan Niaki ST, Mehdizadeh E, Tavarroth MR (2013) The capacitated multi-facility location–allocation problem with probabilistic customer location and demand: two hybrid meta-heuristic algorithms. Int J Syst Sci 44(10):1897–1912
31. Geem ZW (2007) Harmony search algorithm for solving Sudoku. In: Apolloni B, Howlett RJ, Jain L (eds) KES 2007, part I. LNCS (LNAI), vol 4692. Springer, Heidelberg, pp 371–378
32. Lee KS, Geem ZW (2004) A new structural optimization method based on the harmony search algorithm. Comput Struct 82:781–798
33. Geem ZW, Kim J-H, Loganathan GV (2002) Harmony search optimization: application to pipe network design. Int J Model Simul 22(125–133):2002
34. Taleizadeh AA, Niaki STA, Barzinpour F (2011) Multiple-buyer multiple-vendor multi-product multi-constraint supply chain problem with stochastic demand and variable lead-time: a harmony search algorithm. Appl Math Comput 217:9234–9253
35. Rahmati SHA, Hajipour V, Niaki STA (2013) A soft-computing Pareto-based meta-heuristic algorithm for a multi-objective multi-server facility location problem. Appl Soft Comput 13(4): 1728–1740
36. Yeniay O, Ankare B (2005) Penalty function methods for constrained optimization with genetic algorithms. Math Comput Appl 10:45–56
37. Geem ZW, Kim J-H, Loganathan GV (2001) A new heuristic optimization algorithm: harmony search. Simulation 76(2):60–68
38. Haupt RL, Haupt SE (2011) Practical genetic algorithms, 2nd edn. John Wiley & Sons
39. Kirkpatrick S, Gelatt C, Vecchi M (2003) Optimization by simulated annealing. Science 220:671–680
40. Bandyopadhyay S, Saha S, Maulik U, Deb K (2008) A simulated annealing-based multiobjective optimization algorithm: AMOSA. IEEE Trans Evol Comput 12(3):269–283
41. Pasandideh SHR, Niaki STA, Hajipour V (2013) A multi-objective facility location model with batch arrivals: two parameter-tuned meta-heuristic algorithms. J Intell Manuf 24(2):331–348
42. Zitzler E, Thiele L (1998) Multi-objective optimization using evolutionary algorithms a comparative case study. In: Eiben A E, Back T, Schoenauer M Schwefel H P (eds) Fifth International Conference on Parallel Problem Solving from Nature (PPSN-V), pp. 292–301. Berlin, Germany
43. MATLAB (2010) Version 7.10.0.499 (R2010a). The MathWorks, Inc. Protected by US and International patents