Journal of Engineering Research xxx (xxxx) xxx–xxx

Contents lists available at ScienceDirect

Journal of Engineering Research

journal homepage: www.journals.elsevier.com/journal-of-engineering-research

Full length article

Variable neighborhood search-based algorithms for the parallel machine capacitated lot-sizing and scheduling problem

Seyda Topaloglu Yildiz[a,*,1], Sel Ozcan[b,2], Neslihan Cevik[b,c,3]

[a] *Department of Industrial Engineering, Dokuz Eylül University, İzmir, Türkiye*
[b] *Graduate School of Natural and Applied Sciences, Dokuz Eylül University, İzmir, Türkiye*
[c] *Department of Industrial Engineering, İzmir University of Economics, İzmir, Türkiye*

A R T I C L E I N F O

Keywords:
Capacitated lot-sizing and scheduling problem
Parallel machines
Heuristics
Variable neighborhood search
Variable neighborhood descent
Constraint handling techniques

A B S T R A C T

This paper addresses the capacitated lot-sizing and scheduling problem on parallel machines with eligibility constraints, sequence-dependent setup times, and costs. The objective is to find a production plan that minimizes production, setup, and inventory holding costs while meeting the demands of products for each period without delay for a given planning horizon. Since the studied problem is NP-hard, we proposed metaheuristic approaches, Variable Neighborhood Search, Variable Neighborhood Descent, and Reduced Variable Neighborhood Search algorithms to analyze their performance on the problem. Initially, we presented an initial solution generation method to satisfy each period's demand. Then, we defined insert, swap, and fractional insert moves for generating neighborhood solutions. We employed an adaptive constraint handling technique to enlarge the search space by accepting infeasible solutions during the search. Lastly, we performed computational experiments over the benchmark instances. The computational results show the effectiveness of the proposed solution approaches, compared to existing solution techniques in the literature, and the improvements in various problem instances compared to the best-known results.

Introduction

Lot-sizing problems are usually classified as small and large bucket problems. While small bucket problems consist of small-time periods that usually allow one product or setup per period and machine, big bucket problems contain fewer and longer periods and typically have no restriction on the number of products or setups per period and machine. A production schedule cannot be directly concluded from a solution with large bucket models since the sequence of products is not determined within each period. The Capacitated Lot-Sizing Problem (CLSP) provides a standard mathematical formulation for large bucket lot-sizing problems with a single machine for a specified number of periods and the known demand for each period. In this study, the CLSP is integrated into scheduling decisions. Moreover, the lot-sizing and scheduling decisions are given in the presence of parallel machines and sequence-dependent setups. Many practical cases of the Parallel Machine Capacitated Lot-Sizing and Scheduling Problem (PM_CLSP) exist in the real world [24].

As Hansen et al. [21] reported, Variable Neighborhood Search (VNS)-based heuristics have proven to be the leading heuristics for many NP-hard optimization problems. In recent years, many variants of this metaheuristic have been successfully applied to a wide variety of optimization problems. These performances indicate that developing VNS heuristics to address other NP-hard optimization issues will lead to a promising avenue of research. These algorithms' simplicity, ease of use, and high performance on various scheduling problems are also stated in [16,21,42]. Therefore, in this study, we employed the VNS and its variants, Variable Neighborhood Descent (VND), and Reduced Variable Neighborhood Search (RVNS) algorithms to solve the PM_CLSP, which have not been employed before in the solution of this problem. Other contributions of our study include the use of the adaptive constraint-handling technique, the adaptation of neighborhood solution generation schemes previously defined for the single-machine case in Almada-Lobo et al. [2] for the parallel-machine case, and the proposal of an additional new neighborhood scheme.

* Corresponding author.
E-mail address: seyda.topaloglu@deu.edu.tr (S.T. Yildiz).
1 ORCID: 0000–0001-6827–126X
2 ORCID: 0000–0002-4711–6663
3 ORCID: 0000–0003-4853–0548

https://doi.org/10.1016/j.jer.2023.100145
Received 7 May 2023; Received in revised form 22 June 2023; Accepted 3 July 2023

The remainder of the paper is organized as follows. Section 2 presents the literature review on the PM_CLSP. The mathematical formulation of the PM_CLSP is given in Section 3. Section 4 describes the initial solution generation procedure, proposed VNS, VND, and RVNS solution approaches, and neighborhood schemes used. Finally, the computational results and concluding remarks are presented in Sections 5 and 6, respectively.

Literature review

The standard CLSP is defined in Quadt and Kuhn [37] as follows. In each period, more than one product with deterministic and discrete demand quantities should be produced. Each product occupies a portion of the machine's capacity, and the setup cost occurs with each change in the set of products occupying the machines. Additionally, unique inventory holding costs are incurred when a product unit has been produced in the previous periods. The objective is to find an optimal production plan to minimize setup and inventory costs and find optimal lot sizes for each period and each product to satisfy each period's demand.

An extension of the CLSP discussed in this study is the one with parallel machines. The parallel machine CLSP (PM_CLSP) can be seen in various industries, such as chemicals, electronics, food, and textiles [37,34,39,48]. The complexity of the problem increases with this extension since a decision must be made on which machine to produce a product and how many machines to use in parallel for each product in each period [37].

Setup carryover is another extension of the CLSP. Carrying over a setup between periods is observed in many industries; for instance, in the semiconductor industry, production runs 24 h a day and seven days a week [36]. Setup carryover means that a machine's setup state can be preserved between two consecutive periods, requiring no extra setup. As Quadt and Kuhn [37] stated, in the standard CLSP, a setup is made for each product produced per period (and machine), whereas, with the setup carryover, the final product per period can be produced without any additional setup incurred in the following period. Haase's study (1998) has denoted that the solutions become significantly different when setup carryover is considered. Quadt and Kuhn [37] claim that if setup carryover is accounted for with parallel machines, a lot-for-lot policy can substantially reduce the number of setup operations. Carrying over a setup state makes the problem more complicated because a decision must be made for each machine as to which product should be the first and the last in a period. Gopalakrishnan et al. [17] developed a model for the CLSP problem with constant setup times and setup carryovers and noted the complexity of the proposed model. Suerie and Stadtler [45] presented a different mathematical model with sequence-independent setup costs and times. They used the idea of the standard facility location formulation and proposed new sets of variables and constraints to model the setup carryover.

A complete sequence of all products on each machine must be conducted when sequence-dependent setup times and/or costs are present. Bitran and Yanasse [6] show that the CLSP is NP-hard even without setup times. When sequence-dependent setups are also considered, the CLSP becomes NP-complete implying that it is hard to find a feasible solution Maes and Van Wassenhove [29]. The CLSP with sequence-dependent setup times is similar to the TSP and the vehicle routing problem (VRP) [27,26]. More specifically, the setup cost matrix in the CLSP is similar to the distance matrix in the TSP or VRP. However, solving the multi-period CLSP is equivalent to solving multiple-dependent TSPs.

Haase and Kimms [20] studied the CLSP with sequence-dependent setups. They introduced a model where the efficient product sequences are predetermined. As a solution approach, they used a tailor-made branch-and-bound method. In the study of Gupta and Magnusson [19], they initially developed an exact formulation of the single machine CLSP with sequence-dependent setup costs, non-zero setup times, and setup carryover as a Mixed Integer Programming (MIP) model. For large problem instances, they proposed a heuristic for a solution. The heuristic generates an initial feasible solution, assuming setup carryovers are used in each period. Then, setups are sequenced in each period in a greedy way. The final step starts with the last period, and unused capacity is searched in preceding periods. Then, setup and holding costs are checked to determine whether production may be moved to an earlier period with sufficient slack capacity and the overall cost is reduced. This step proceeds until all opportunities for moving production have been exhausted. Zhu and Wilhelm [52] conducted a literature review on the CLSP with sequence-dependent setup times on single and parallel machines.

Almada-Lobo et al. [2] presented two new linear MIP models for the single machine CLSP with sequence-dependent setup times and costs and setup carryover. No backlogging is allowed, i.e., each product's demand should be fully satisfied for each period. To keep track of schedules, they introduced a TSP-influenced constraint to simplify their formulations concerning others in the literature. Furthermore, they proposed a five-step heuristic for finding feasible solutions. Similarly, Sarin et al. [41] presented a high-multiplicity TSP in which the traditional TSP is enhanced by allowing multiple visits to the nodes, and a polynomial length formulation of the problem has been achieved with flow-based sub-tour elimination constraints. de Armas and Laguna, [4] proposed an MIP model for a pipe insulation company to obtain a lower bound for the total production maximization objective with a sequence-independent version of the problem. The solution obtained from the MIP model is used in a post-processing sequencing heuristic to create the best-performing sequence of products with sequence dependency.

In this paper, we conducted literature research on the PM_CLSP, considering several operational characteristics and the deployed solution methods, and showed them in Table 1. MIP-based heuristics are commonly selected solution approaches for the PM_CLSP. James and Almada-Lobo [23] studied the PM_CLSP with sequence-dependent setups, setup carryover, and machine eligibility restrictions among products. They proposed an MIP-based iterative neighborhood search heuristic, divided the problem into many sub-MIPs, and solved them randomly. The stochasticity of the solution algorithm arises from the idea of the selection of the sub-MIPs.

In Xiao et al.'s (2013) study, the PM_CLSP is studied with sequence-dependent setup times and costs, setup carryover, and backlogging. Similar to the study of James and Almada-Lobo [23], they also considered machine eligibility restrictions. Moreover, they have included soft machine preference constraints, the violation of which is penalized in the objective function. To find high-quality solutions, they proposed MIP-based Relax-and-Fix and Fix-and-Optimize heuristics, where the binary decision variables related to the assignment of machines are first fixed using the randomized least flexible machine rule, and the rest of the decision variables are settled by a MIP solver. Similarly, Beraldi et al. [5] proposed rolling-horizon and Fix-and-Relax heuristics, while Carvalho and Nascimento [7] used the Relax-and-Fix and Fix-and-Optimize heuristics with different search strategies, such as path relinking and kernel search. Larroche et al. [28] proposed a hybrid heuristic method to obtain feasible solutions while considering lost sales, overtime, safety stock, and sequence-dependent setups on the unrelated PM_CLSP. The proposed method uses the clustering technique to approximate the sequence-dependent setup times while using the setup times within the relax-and-fix and fix-and-optimize heuristics. In a study by Mateus et al. [32], a hybrid strategy was proposed where the lot-sizing problem was solved with integer programming, and the scheduling problem was solved with the GRASP heuristic.

Güngör et al. [18] proposed an MIP-based heuristic approach for the identical machine PM_CLSP while minimizing the required setups and teardowns. Marinelli et al. [31] proposed a heuristic approach to relax the lower bounds of the problem and suggested a decomposition method for their MIP model.

Table 1
Literature Research on the PM_CLSP.

PM_CLSP	Machine characteristics		Setup features		Operational constraints					Method	
References	**Unrelated**	**Eligibility**	**Sequence-Dependent**	**Setup Carryover**	**Backorder**	**Backlog**	**Safety Stock**	**Lost Sales**	**Objective (Min)**	**Exact**	**Heuristic**
Radhakrishnan and Ventura[38]	uniform		√						Delay		Metaheuristic-SA
Quadt and Kuhn[36]	uniform			√	√				Cost & Time		MIP-based H.
Sambasivan and Yahya [40]	uniform								Cost		MIP-based H.- Lagrangean
Silva and Magalhaes[43]	√	√			√				Setup		Heuristic-DD-based
Marinelli et al.[31]	√								Cost		MIP-based H. Relaxation-based
Beraldi et al.[5]	√		√						Cost		MIP-based H.- Fix-and-relax
Dolgui et al.[11]	√	√	√						Makespan		Heuristic-Greedy
Mateus et al.[32]	√		√		√				Cost & Makespan		Matheuristic-Hybrid-MIP and GRASP
James and Almada-Lobo [23]	√	√	√	√					Cost		Matheuristic
Sarin et al.[41]	√	√	√	√					Cost	MIP-HMATSP	
Kaczmarczyk[25]	uniform		√						Cost	MIP	
Ibarra-Rojas et al.[22]	√	√							Cost	MIP	
Almeder and Almada-Lobo[3]	√		√	√	√	√			Cost	MIP	
Xiao et al.[49]	√	√	√	√		√			Cost		MIP- based H.- RFO
Seeanner et al.[42]	√		√						Cost		Metaheuristic-VNDS and FO
Fiorotto and de Araujo [14]	√								Cost		MIP- based H.- Lagrangean
Fiorotto et al.[15]	√								Cost		MIP- based H.- LR, and DW
Xiao et al.[50]	√	√	√	√		√			Cost		Metaheuristic-SA
Fachini et al.[12]	√		√	√		√		√	Loss		
Mahdieh et al.[30]	uniform		√	√	√	√			Cost	MIP	
Güngör et al.[18]	uniform								Setup		MIP- based H.
Ghirardi and Amerio[16]	√			√	√			√	Cost		Matheuristic-VNS, LB, FP
Qin et al.[35]	√		√						Makespan		Metaheuristic-Two-stage ACO
Vincent et al.[47]	√								Cost		Metaheuristic-PR
de Armas and Laguna,[4]	√		both	√			√		Quantity (Max)		Heuristic
Zhang et al.[51]	√	√	√	√		√			Cost		Metaheuristic-SODBS
Carvalho and Nascimento [7]	√		√	√					Cost		MIP-based Matheuristic-RFO-PL and RFO-KS
Larroche et al.[28]	√		√				√	√	Cost		MIP-based H.-RFO with Clustering
This paper	√	√	√	√					Cost		VNS, VND and RVNS

Set-oriented Data-driven Branching and Selection (SODBS), High Multiplicity Asymmetric Traveling Salesman Problem (HMATSP), Local Branching (LB), Feasibility Pump (FP), Dantzig-Wolfe (DW) decomposition, Variable Neighborhood Decomposition Search (VNDS), Fix-and-Optimize (FO), Relax-Fix and Fix-Optimize (RFO) with Path-Relinking and Kernel Search (RFO-PL & RFO-KS), Asymmetric Traveling Salesman Problem, (ATSP), Variable Neighborhood Search (VNS), Simulated Annealing (SA), Due Date (DD), Greedy Randomized Adaptive Search Procedure (GRASP), Ant Colony Optimization (ACO), Path-Relinking (PR), Variable Neighborhood Descent (VND), Reduced Variable Neighborhood Search (RVNS)

Several studies have included the non-triangular setup times to the PM_CLSP, where the production of a certain product may cause contamination for another product, which is commonly observed in the food and beverage, oil, and feed industries Clark et al. [8]. The potential time to clean the contamination during the setup times is eliminated by producing item l right after item i instead of item j since item l can clean the contamination during the production period, particularly in the abovementioned industries. Carvalho and Nascimento [7] proposed a matheuristic approach to solving the unrelated parallel machine CLSP with sequence-dependent and non-triangular setup times to minimize the operational costs in the food industry. The proposed model enables a product to be produced multiple times with various sequences during a period. In a different study by Mahdeih et al. (2018), non-triangular setup times were considered while analyzing the identical PM_CLSP with sequence-dependent setups and setup carryover characteristics. This study has proposed a flexible MIP approach to tackle this problem by incorporating the setup carryover and setup overlapping features into their model.

Recent studies have included the setup carryovers and sequence-dependent setups in the PM_CLSP and proposed heuristic methods to overcome the complexity of the problem with these extensions Carvalho and Nascimento [7,16]. In a study by Silva and Magalhaes [43], the unrelated parallel machine discrete lot-sizing problem was solved with a due date-based heuristic method to minimize the total number of tool changeovers.

Sambasivan and Yahya [40] and Fiorotto and de Araujo [14] used the Lagrangian Relaxation to approximate the optimal solution. Similarly, Fiorotto et al. (2014) proposed a Lagrangian Heuristic that applied to the demand constraints, and the relaxed problem was decomposed per period and machine decomposition. A primal heuristic, based on production transfers, was designed to generate feasible solutions. Dolgui et al. [11] proposed a Greedy Heuristic and compared its performance with the Genetic Algorithm (GA) and exact solution methods, stating that the proposed algorithm performed better.

The exact solution strategies proposed in the PM_CLSP are MIP-based exact solutions conducted with various problem characteristics and objectives. Mahdieh et al. [30] proposed a novel flexible MIP model to minimize the operational costs on identical PM_CLSP with the non-triangular sequence-dependent setup costs and times with setup carryovers, backlog, and backorder allowances and branch and cut search with period overlapping property to relax the limitations of physical separation between the periods. In a different study, both the general lot-sizing and scheduling problem and the capacitated lot-sizing and scheduling problem have been studied, and different novel MIP solutions have been proposed to minimize operational costs [3]. In 2011, Ibarra-Rojas et al. proposed a decomposition-based approach where they constructed two MILPs, one of which determines the lot size, and the other searches for a feasible schedule for the machines. Ferreira et al. [13] proposed a TSP solution to the PM_CLSP with sequence-dependent characteristics, backorder, and backlog allowances possessing a remarkable performance. They compared the results obtained from their TSP-based method with the branch and cut method with limited execution times. Another study was conducted on the Proportional Lot-sizing and Scheduling Problem (PLSP) with identical machines using an MIP model to minimize the operational costs where the binary variables are replaced with integer variables that describe the number of machines enabling better performance [25].

As observed in Table 1, these exact solution strategies lack some of the mentioned characteristics due to a possible increase in the complexity and computation time ([3]; Mahdeih et al., 2018). This situation led the researchers to develop heuristics to tackle the complexity of the problem.

Besides the exact solution strategies, commonly used metaheuristics are used to solve the PM_CLSP in the literature. Qin et al. [35] used Ant Colony Optimization; Xiao et al. [50] and Radhakrishnan and Ventura [38] both used Simulated Annealing for their problems. Besides the mentioned metaheuristic methods, metaheuristics with different search strategies are introduced. Vincent et al. [47] constructed a population-based metaheuristic and used a path-linking strategy to strengthen their diversification procedure. Seeanner et al. [42] combined the variable neighborhood search metaheuristic with the Fix-Optimize heuristic to solve the unrelated PM_CLSP to minimize operational costs. Zhang et al. [51] proposed a metaheuristic method to improve the initial solution generated with a branching and selection procedure.

Matheuristics are other commonly used solution approaches in the literature for both the single and the PM_CLSP. The goal of matheuristics is to combine metaheuristics with mathematical programming techniques. The studies that deployed matheuristics as solution strategies are listed in Table 1.

Regarding the related literature, no study employs the VNS, VND, and RVND algorithms to solve the PM_CLSP. Given the simplicity, ease of use, and high performance of these algorithms on various scheduling problems [16,21,42], this paper proposes solution approaches based on the VNS, VND, and RVND algorithms for solving the PM_CLSP with unrelated parallel machines, sequence-dependent setups, machine eligibility restrictions, and setup carryovers to minimize the overall inventory and setup costs.

Mathematical formulation of the PM_CLSP

We consider a set of products $i, j \in N$ processed on $m \in M$ unrelated machines with eligibility and capacity constraints over a discrete planning horizon with $t \in T$ periods. Due to the sequence-dependency of setups in a product changeover, lot-sizing and sequencing decisions for products are simultaneously tackled. The objective is to find a strategy that satisfies demands and minimizes both setup and holding costs. James and Almada-Lobo previously modeled the PM_CLSP with sequence-dependent setups and setup carryover (2011).

The parameters and decision variables necessary for the mathematical formulation of the problem can be listed as follows:

Parameters

d_{it} = demand of product i in period t.

s_{mij} = setup time incurred when a setup occurs from product i to j on machine m.

c_{mij} = setup cost incurred when a setup occurs from product i to j on machine m.

h_i = unit inventory holding cost for product i from one period to the next.

p_{mi} = processing time of one unit of product i on machine m.

C_{mt} = capacity of machine m available in period t.

G_{mit} = upper bound on the production quantity of product i in period t on machine m.

A_{mi} = product i's capability of machine m.

Decision Variables

X_{mit} = quantity of product i produced in period t on machine m.

I_{it} = inventory level of product i at the end of period t.

V_{mit} = an auxiliary variable that assigns product i to machine m in period t.

T_{mijt} = 1 if a setup occurs from product i to j on machine m in period t.

Y_{mit} = 1 if machine m is set up for product i at the beginning of period t.

Mathematical Model

$$\min \sum_m \sum_i \sum_j \sum_t c_{mij}.\ T_{mijt} + \sum_i \sum_t h_i.\ I_{it} \quad 1$$

$$I_{i(t-1)} + \sum_m X_{mit} - d_{it} = I_{it},\ i \in N,\ t \in T \quad 2$$

$$I_{i0} = 0, \quad i \in N \quad 3$$

$$\sum_i p_{mi}.\ X_{mit} + \sum_i \sum_j s_{mij}.\ T_{mijt} \le C_{mt},\ m \in M,\ t \in T \quad 4$$

$$X_{mit} \le G_{mit}.\left(\sum_j T_{mjit} + Y_{mit}\right),\ m \in M,\ i \in N,\ t \in T \quad 5$$

$$Y_{mi(t+1)} + \sum_j T_{mijt} = Y_{mit} + \sum_j T_{mjit}\, m \in M,\ i \in N,\ t \in T \quad 6$$

$$\sum_i Y_{mit} = 1,\ m \in M,\ t \in T \quad 7$$

$$V_{mit} + N.\ T_{mijt} - (N-1) - N.\ Y_{mjt} \le V_{mjt},\ m \in M,\ i \in N,\ j \in N \setminus \{i\},\ t \in T \quad 8$$

$$\sum_t X_{mit} \le G_{mit} A_{mi},\ m \in M,\ i \in N \quad 9$$

$$X_{mit},\ I_{it} \ge 0,\ T_{mijt},\ Y_{mit} \in \{0, 1\},\ X_{mit} \in Z,\ V_{mit} \in R \quad 10$$

The objective function (1) minimizes the overall inventory and setup costs. Constraint (2) enforces production and inventory balance, whereas Constraint (3) sets the initial inventory level of each product to zero. The available capacity of the machines limiting the total production and setup time of each machine is set by Constraint (4). Constraint (5) imposes that a setup is made each time a different product is scheduled on a machine, while Constraint (6) ensures the setup carryover for two consecutive periods. Constraint (7) states that each machine should be set up for one product at the beginning of each period. Constraint (8) eliminates disconnected subtours. In other words, this Constraint works whenever a subtour occurs in a period, forcing the respective machine to be set up at the beginning of that period to a product that is part of the subtour. Constraint (9) indicates each machine's eligibility to produce a product, and Constraint (10) is for the nonnegativity and integrality of the decision variables.

Proposed solution approaches

This section describes the basic VNS and VND, the initial solution generation procedure, the neighborhood generation schemes, and the VNS solution methods developed.

Basic variable neighborhood search

VNS is a common approach to enhance solution quality with systematic neighborhood changes within a local search [33]. The algorithm involves iterative exploration of larger and larger neighborhoods for a given local optimum unless there is an improvement, and then the search is repeated. In VNS, this systematic neighborhood change is both deterministic and stochastic, whereas, in VND, only the deterministic part of VNS is used.

The basic procedure of VNS is as follows: start with an initial solution π_i and set π as the incumbent solution. Following, select a random solution π_1 from the k^{th} neighborhood of π ($\pi_1 \in N_k(\pi)$) in the *Shaking Phase* (*SP*), and find an improving solution π_2 (if any) in the k^{th} neighborhood of π_1 ($\pi_2 \in N_k(\pi_1)$), using the *Local Search* (*LS*). If there is an improvement, i.e., ($f(\pi_2) < f(\pi)$), set π_2 as the new incumbent solution, continue the search with the 1st neighborhood; otherwise, move to the next neighborhood. The search continues as long as $k \le k_{\max}$; otherwise, an iteration of the algorithm is completed, and the neighborhood structure is reset to the 1st neighborhood structure. The procedure stops when a stopping condition is fulfilled. The VNS procedure is presented in Algorithm 1.

Algorithm 1 Basic VNS Procedure

```
1: B_VNS($\pi_i$, $k_{max}$)
2: $\pi = \pi_i$
3: repeat
4:     $k = 1$
5:     while $k \le k_{max}$ do
6:         $\pi_1 = SP(\pi, N_k)$
7:         $\pi_2 = LS(\pi_1, N_k)$
8:         if $f(\pi_2) < f(\pi)$ then
9:             $\pi = \pi_2$
10:            $k = 1$
11:        else
12:            $k = k + 1$
13:        end if
14:    end while
15: until the stopping condition is fulfilled
16: return $\pi$
```

Basic variable neighborhood descent

In basic VND, the change in neighborhoods is performed in a deterministic way. Starting with an initial solution π_i and setting π as the incumbent solution, an improving solution (if any) is searched in $N_k(\pi)$ using *LS*. If there is an improvement, π_1 is set as the new incumbent solution, and the search continues with the 1st neighborhood; otherwise, it is moved to the next neighborhood. The search stops when all neighborhoods are tried, and no improvement is found. The basic VND procedure is given in Algorithm 2.

Algorithm 2 Basic VND Procedure

```
 1: Function B_VND ($\pi_i$, $k_{max}$)
 2:   $\pi = \pi_i$
 3:   while $k \le k_{max}$ do
 4:       $\pi_1 = LS(\pi, N_k)$
 5:       if $f(\pi_1) < f(\pi)$ then
 6:           $\pi = \pi_1$
 7:           $k = 1$
 8:       else
 9:           $k = k + 1$
10:       end if
11:   end while
12:   return $\pi$
```

Initial solution generation

An initial solution should be provided before the proposed algorithms are applied. Starting from the last period, for each period, a product on demand is chosen randomly and allocated to one of the eligible machines with the largest capacity unused. If the demand for the chosen product cannot be produced due to the insufficient capacity of the machine, only a part of the demand corresponding to the remaining capacity is allocated. The rest is produced one period earlier on the same machine. Note that the advantage of the setup carryover between the periods is considered in that case; on the other hand, this allocation results in inventory holding costs for one period. If there is not enough capacity left on this machine, the capacity violation in the corresponding period is calculated for the machine. Therefore, the objective function consists of sequence-dependent setup costs allowing setup carryover allowance between consecutive periods, inventory holding costs, and capacity violation penalties. The initial solution algorithm stops when the demand for all products for each period is allocated to the machines. Obtaining a feasible initial solution is unnecessary since we have used constraint-handling techniques in the developed solution approaches.

Neighborhood schemes

Three different types of moves previously defined by Almada-Lobo and James [1] for the single machine case have been adapted to our parallel machine case. These moves can be explained as follows.

In the *Insert Move (IM)*, the product lot is chosen randomly and inserted before another product randomly. Unlike the single-machine case, this move can be made on the same machine or other eligible machines. However, the *IM* is always possible in this study since capacity violations of machines are allowed.

In the *Swap Move* (*SM*), two product lots chosen randomly from different machines, either in the same period or different periods, are swapped without considering any capacity violations of machines on the condition that the eligibility of these machines to produce the swapped product lots is satisfied. However, similar to the insert move, two product lots on the same machine, whether in the same period or not, can also be swapped. Due to constraint violation techniques, it would not be prohibited even if a swap move results in infeasible solutions in capacity, demand satisfaction, or upper-bound production quantity.

The Fractional Insert Move (*FIM*) is similar to the *IM;* however, it allows splitting a product lot into two lots, where the total quantity produced is the same as the original lot. One of these new lots is left in the same position as the original lot, while the second part is randomly inserted into a new location on a randomly selected eligible machine. This move must occur on the selected machine for the same production period as the selected product lot. Therefore, the quantity moved is dictated by the available capacity of the machine during the period it is moved. Moreover, the capacity violation is not permitted in the *FIM*. If there is enough capacity, the complete lot will be moved; if not, only the amount that can fit in the period will be moved. All locations within the period are tested as the capacity available will vary depending on the new lot's position, the lots surrounding it, and the sequence-dependent setup times.

Objective function calculations

The objective function calculation differs from the standard calculation, where only setup and inventory holding costs are incurred. Infeasible solutions involving constraint violations have been accepted as candidate solutions during the search because the optimal solution can be found at the boundaries of the feasible region. Deb [10] proposes the following three criteria while deciding on the *Superiority of Solutions*.

1. Any feasible solution is preferred over an infeasible solution.
2. Among two feasible solutions, the one with a better objective function value is preferred.
3. Among two infeasible solutions, one with a smaller constraint violation is preferred.

In the initial solution generation phase, since the assignment of product lots is done in the period demanded or one period earlier, without considering machine capacity constraints, the *total violation* of constraints for the initial solution is calculated as the sum of only the capacity violations for machines. However, this situation is different for solutions generated after infeasible moves. For example, when two products are swapped from different periods, there is no guarantee that the demand for one or both will be satisfied. Thus, in addition to the capacity violation amount, the unsatisfied demand and the excess of the upper bound production levels of each product on each machine for each period should also be included in the calculation of the *total violation*.

An adaptive constraint handling method, known as *Near-Feasible Threshold* (*NFT*) [44,46], has been adopted for the proposed VNS and VND algorithms during the search. It handles the violation of constraints considering the search duration and the distance from the feasibility. *NFT* is computed for each constraint in the penalized objective function. It is defined as the threshold distance from the feasible region. With the penalty function, the algorithm is encouraged to explore the feasible region and the *NFT* neighborhood of the feasible region. In other words, searches that exceed the threshold are discouraged. The penalized objective function $f_p(x, t)$ is given below, where $f(x)$ is the unpenalized objective function:

$$f_p(x,\ t)\ = f(x) + (f_{feas}(t) - f_{all}(t))\sum_{i=1}^{m} \underbrace{\left(\frac{d_i}{NFT_i}\right)^k}_{dynamic\ part\ of\,NFT} \qquad 11$$

As stated in Equation (11), $F_{all}(t)$ denotes the unpenalized value of the best solution yet found, and $F_{feas}(t)$ denotes the value of the best feasible solution yet found. The $F_{all}(t)$ and $F_{feas}(t)$ terms serve several purposes. However, d_i denotes the violation amount of constraint i, where there are m constraints in total. First, they provide adaptive scaling of the penalty based on the search results. Second, they combine it with the NFT_i term to provide a search-specific and constraint-specific penalty.

The general form of is as follows.

$$NFT_i = \frac{NFT_0}{1 + \Lambda} \qquad 12$$

According to the proposed 12(12), NFT_0 is an upper bound of NFT. Λ is a dynamic search parameter that updates NFT considering the entire search period. Λ has been defined as a function of the iteration number (t), i.e., $\Lambda = f(x) = \lambda t$ (Baeck et al., 1995). Moreover, Gen and Cheng (2000) noted that the adaptive term might lead to zero or over-penalty. For instance, if $F_{feas}(t)$ and $F_{all}(t)$ are identical, the penalty would be zero, resulting in unpenalized infeasible solutions. For this reason, only the dynamic part of the penalty function with the NFT threshold is used [9].

Proposed VNS for the CLSP_PM

In the proposed VNS and VND approaches, the *Best Improvement Local Search* (LS_BI) procedure, given in Algorithm 3, is employed, using either *IM* or *SM* based neighborhoods(N_k), since it entirely explores the search space. The *FIM* cannot be used in the LS_BSI because the transferred quantity of the product lot would change in each move, and thus, there will be infinite possible moves when the entire search is explored.

both the sequence and the lot sizes in the problem, and hence provides diversification to the search.

The notation in the following algorithms can be explained as follows; π_b, the incumbent solution, which is also the best solution found so far; f_b, the best objective function value found so far; and tv_b, the minimum total amount for constraint violations encountered so far. Initially, $\pi_b = \pi_i$, $f_b = f_i$ and $tv_b = tv_i$, where π_i, the initial solution; f_i, the objective function value of the initial solution, and tv_i, the total minimum amount for constraint violations of the initial solution (lines 3–5, Algorithm 4).

The VNS algorithm first selects a random solution from the neighborhood of π_b by the *FIM* for diversification, using the *Shaking* function *SP*($N_{fim}(\ \pi_b)$). Then, the local search phase starts, where an entire search is done within the k^{th} neighborhood, N_k, using the *LS_BI*(π, N_k) function (line 9, Algorithm 4). A local optimum solution π_m is found among all solutions of this neighborhood, and its f_m value is compared with the f_b value of the incumbent solution, π_b, using the *compare* function described in Algorithm 5. In the *compare* function, these two solutions π_b and π_m are compared using the *Superiority of Feasible Solutions* and *NFT* values (lines 2–24, Algorithm 5) described in Section 3.5. According to the *Superiority of Feasible Solutions,* if both solutions are feasible and $f_b < f_m$, π_b requires no updating (lines 3–5, Algorithm 5); otherwise, π_b is updated as π_m, and f_b and tv_b are updated accordingly (lines 6–9, Algorithm 5). However, if π_b is feasible and π_m is infeasible, π_b remains the same (lines 10–11, Algorithm 5). If π_b is infeasible and π_m is feasible, then π_b is updated as π_m, and f_b and tv_b are updated accordingly (lines 12–15, Algorithm 5). Lastly, if both solutions are infeasible, *NFT* values for π_b and π_m are calculated, and if $NFT_b > NFT_m$, then π_b is updated as π_m, and f_b and tv_b are updated accordingly (lines 16–20, Algorithm 5); otherwise, π_b remains the same (lines 21–23, Algorithm 5).

If an improvement in π_b ($imprv = 1$) has been detected with the *compare* function, and, f_b, π_b and tv_b (if necessary) have been updated as mentioned above, the neighborhood structure remains the same, and it

Algorithm 3 Best Improvement Local Search (LS_BI)

```
1: Function LS_BI ($\pi$, $N_k$)
2: Repeat
3:     $\pi_m \leftarrow \pi$
4:     $\pi \leftarrow argmin_{x \in N_k(\pi_m)} f(x)$
5: until $f(\pi_m) \le f(\pi)$
6: return $\pi_m$
```

is turned back to the *SP*($N_{fim}(\pi_b)$) procedure. This part of the search continues until $k = 2$, and no improved solution in the 2nd neighborhood is reached ($imprv = 0$), corresponding to one iteration of the algorithm (lines 8–18, Algorithm 4).

The pseudocode of the developed VNS algorithm is presented in Algorithm 4. We use the LS_BI procedure in the *Local Search Phase* using either *IM* or *SM* neighborhoods, whereas we use the *FIM* in the *Shaking Phase* (*SP*) since it changes the structure of the problem by changing

Algorithm 4 VNS Algorithm

```
1: Function $VNS(\pi_i, k_{max} = 2, CPU_{max} = 3600)$
2: $CPU = 0$
3: $\pi_b \leftarrow \pi_i$
4: $f_b \leftarrow f_i$
5: $tv_b \leftarrow tv_i$
6: while $CPU \le CPU_{max}$ do
7:     $k = 1$
8:     while $k \le k_{max}$ do
9:         $\pi_{fim}, f_{fim}, tv_{fim} \leftarrow SP(N_{fim}(\pi_b))$
10:        $\pi \leftarrow \pi_{fim}$
11:        $\pi_m, f_m, tv_m \leftarrow LS\_BI(\pi, N_k)$
12:        $compare\ (\pi_b, f_b, tv_b, \pi_m, tv_m, f_m)$
13:        if $imprv == 1$ then
14:            $k = 1$
15:        else if
16:            $k = k + 1$
17:        end if
18:    end while
19: end while
20: return $\pi_b$
```

Algorithm 5 Compare Function

```
1: Function $compare\ (\pi_b, f_b, tv_b, \pi_m, tv_m, f_m)$
2: $imprv = 1$
3: if $tv_b == 0$ & $tv_m == 0$ then
4:     if $f_b < f_m$ then
5:         $imprv = 0$
6:     else                                  // update $\pi_b$ and $f_b$
7:         $f_b \leftarrow f_m$
8:         $\pi_b \leftarrow \pi_m$
9:     end if
10: else if $tv_b == 0$ & $tv_m > 0$ then
11:     $imprv = 0$
12: else if $tv_b > 0$ & $tv_m == 0$ then    // update $\pi_b$, $f_b$, and $tv_b$
13:     $f_b \leftarrow f_m$
14:     $\pi_b \leftarrow \pi_m$
15:     $tv_b \leftarrow tv_m$
16: else if $tv_b > 0$ & $tv_m > 0$ then     // Calculate $NFT$
17:     if $NFT_b > NFT_m$ then              // update $\pi_b$, $f_b$, and $tv_b$
18:         $f_b \leftarrow f_m$
19:         $\pi_b \leftarrow \pi_m$
20:         $tv_b \leftarrow tv_m$
21:     else if then
22:         $imprv = 0$
23:     end if
24: end if
25: return $imprv, \pi_b, f_b, tv_b$
```

When an iteration is completed, the algorithm takes π_b as the incumbent solution and initializes $k = 1$ and continues with the next iteration. More specifically, the proposed VNS algorithm repeats the same steps until a predetermined maximum CPU time is reached (lines 6–19, Algorithm 4).

We applied two versions of the VNS algorithm, where the *FIM* is used in the *Shaking Phase* in both versions, while the LS_BI uses *IM* as the 1st neighborhood structure and *SM* as the 2nd one in the first version and vice versa in the second version.

Proposed VND for the CLSP_PM

The main difference between the proposed VNS and VND algorithms is that the VND algorithm does not have a *Shaking Phase*. Also, the algorithm stops when the *k* value reaches two, and there is no improvement in the incumbent solution. In this paper, two different VND algorithms have been tested. The first one adopts $N_1 = IM$, $N_2 = SM$, and the second one adopts $N_1 = SM$, $N_2 = IM$. The pseudocode of the VND algorithm is given in detail in Algorithm 6.

Proposed RVNS for the CLSP_PM

The pseudocode of the RVNS algorithm is presented in Algorithm 7. Different from VNS, RVNS consists of *Shaking Phase* only. Starting with the initial solution, let $\pi_b = \pi_i$, $f_b = f_i$, and $tv_b = tv_i$ be the best values so far. The VNS algorithm first selects a random solution from the neighborhood of π_b using the *Shaking* function $SP(N_1(\pi_b))$. The results were compared with the incumbent solution using *compare* function. If there is an improvement, the search continues within the 1st neighborhood; otherwise, a move to the 2nd neighborhood is done. If there is no improvement in the 2nd neighborhood, the algorithm starts over searching from the 1st neighborhood and repeats all these steps, taking the best-so-far solution π_b obtained from the last iteration as the input until the maximum number of iterations is achieved. Six different RVNS algorithms were tested in this study. Pairwise combinations of the moves, *SM*, *IM*, and *FIM*, are used as the 1st and 2nd neighborhood structures.

Algorithm 6 VND Algorithm

```
1:  Function $VND(\pi_i, N_k, k_{max} = 2)$
2:  $\pi_b \leftarrow \pi_i$
4:  $f_b \leftarrow f_i$
5:  $tv_b \leftarrow tv_i$
6:  $k = 1$
8:  while $k \leq k_{max}$ do
9:      $\pi_m, f_m, tv_m \leftarrow LS\_BI(\pi_b, N_k)$
10:     $compare(\pi_b, f_b, tv_b, \pi_m, tv_m, f_m)$
11:     if $imprv == 0$ then
12:         $k = k + 1$
13:     else if
14:         $k = 1$
15:     end if
16: end while
17: return $\pi_b$
```

Algorithm 7 RVNS Algorithm

```
1:  Function $RVNS(\pi_b, iteration_{max} = 3000, k_{max} = 2)$
2:   $\pi_b \leftarrow \pi_i$
3:   $f_b \leftarrow f_i$
4:   $tv_b \leftarrow tv_i$
5:   $iteration = 1$
6:  while $iteration \le iteration_{max}$ do
7:      $k = 1$
8:      while $k \le k_{max}$ do
9:          if $k == 1$ then
10:             $\pi_m, f_m, tv_m \leftarrow SP(N_1(\pi_b))$
11:         else
12:             $\pi_m, f_m, tv_m \leftarrow SP(N_2(\pi_b))$
13:         end if
14:         $imprv \leftarrow compare\ (\pi_b, tv_b, f_b, \pi_m, tv_m, f_m)$
15:         if $imprv$ ==0 then
16:             $k = k + 1$
17:         else
18:             $k = 1$
19:         end if
20:     end while
21:     $iteration = iteration+1$
22: end while
```

Computational study

To assess the performance of the proposed VNS, VND, and RVNS algorithms, we tested them on benchmark instances used in the literature and compared their solutions with those developed by James and Almada-Lobo [23].

Problem instances tested

The benchmark instances were generated by James and Almada-Lobo [23] based on different problem types. They have represented the problem types as follows:

$M - N - T - Cut - \theta - MProb - MBal$

Where M denotes the number of machines, N denotes the number of products, T denotes the number of periods, Cut denotes the capacity usage per period. Moreover, θ indicates the setup cost per unit of time, and $MProb$ represents the total number of possible product-machine allocations, i.e., when $MProb$ increases, the problem becomes harder to solve. Lastly, $MBal$ indicates the balance of products across the machines. For each problem type, there are ten benchmark instances. A total of 100 instances are solved with each proposed algorithm.

Comparison with the existing solution algorithms

The existing solution algorithms by James and Almada-Labo (2011) use the Relax-and-Fix heuristic for the initial solution generation in all but different search strategies in the local search phase. They are abbreviated as follows for the local search strategies:

XPHRF: XPH *(Quantity (X) Period (P) Heuristic (H))* in the local search.

INSRF: MIP-based iterative neighborhood search heuristic in the local search.

FOHRF9: Fix-and-Optimize improvement heuristic with an MIP solution tolerance of 10^{-9} in the local search.

For each developed algorithm, the average percentage deviations from the lower bounds for each of the ten different problem types were calculated over ten problem instances and compared with the results of the XPHRF, INSRF, and FOHRF9 solution methods. The average percentage deviation from the lower bound is calculated using the following formula,

$$\frac{\left[\sum_{i=1}^{10}\frac{(best\quad so\quad far - Lower\quad Bound)\times 100}{Lower\quad Bound}\right]}{10} \qquad 13$$

The lower bound values were taken from James and Almada-Labo's (2011) study, which is based on a plant-location reformulation of their model and is known to produce tighter bounds. They have run this model for a one-hour limit to obtain the lower bounds. All computational experiments were performed on Intel (R) Core (TM) i-5 2430 M CPU:2.40 GHz with 4 GB RAM, and the algorithms were coded in MATLAB R2010A. After the preliminary test results, the NFT parameters were identified as the following: NFT_0 0.001 and λ: 0.4.

As mentioned before, we applied two versions of the VNS and VND algorithms, while the LS_BI uses IM as the 1st neighborhood structure and SM as the 2nd one in the first version and vice versa in the second version. Both versions of the VNS algorithm (VNS(1), VNS(2)) were run under the 3600-second time limit. As shown in Table 2, they improved the results of six problem types, and VNS(2) achieved a better overall average deviation than the known best-performing XPHRF.

Table 3 indicates the average percentage deviations from the lower bound values for both versions of the VND algorithm (VND(1) and VND (2)). As can be seen, the average solution values were improved for three problem types, and both VND(1) and VND(2) achieved a better overall average deviation.

The findings for all RVNS versions are presented in Table 4. The average solution values for each problem type were improved with one of the RVNS versions. The overall average deviations, except RVNS(2) and RVNS(3), are the best among all proposed and existing solution approaches, with RVNS(6) resulting in the best value of 6.23 %.

Table 2
Comparison of average percentage deviations from the lower bound values for the VNS algorithm.

Problem Type $M - N - T - Cut - \theta - MProb - MBal$	VNS (1) k_1: IM, k_2: SM	VNS (2) k_1: SM, k_2: IM	XPHRF	INSRF	FOHRF9
2–15–5–0.8–50–80–20	**1.36**	1.52	1.49	1.47	2.03
2–15–10–0.8–50–80–20	**2.55**	2.59	2.60	2.92	3.06
2–15–10–0.8–100–80–20	8.11	**7.08**	7.10	8.24	7.84
2–15–10–0.8–100–80–20	6.27	**4.94**	5.90	5.06	5.19
2–15–10–0.8–100–60–20	6.17	6.59	6.02	**6.01**	7.13
2–20–10–0.8–100–80–10	7.60	**6.86**	6.98	8.03	7.68
3–15–5–0.8–50–80–20	9.33	6.74	10.78	6.93	6.63
3–15–10–0.6–100–80–20	4.51	**4.28**	4.37	5.13	4.64
3–15–10–0.8–50–60–20	13.02	12.97	12.86	14.29	14.31
3–15–10–0.8–100–60–20	8.02	9.40	8.00	9.31	9.78
Overall Average	6.69	**6.30**	6.61	6.74	6.83

Table 3
Comparison of average percentage deviations from the lower bound values for the VND algorithms.

Problem Type $M - N - T - Cut - \theta - MProb - MBal$	VND (1) k_1: IM, k_2: SM	VND (2) k_1: SM, k_2: IM	XPHRF	INSRF	FOHRF9
2–15–5–0.8–50–80–20	2.01	1.71	1.49	**1.47**	2.03
2–15–10–0.8–50–80–20	2.71	**2.52**	2.60	2.92	3.06
2–15–10–0.8–100–80–20	**6.96**	7.07	7.10	8.24	7.84
2–15–10–0.8–100–80–20	5.12	5.09	5.90	**5.06**	5.19
2–15–10–0.8–100–60–20	6.11	6.38	6.02	**6.01**	7.13
2–20–10–0.8–100–80–10	**6.74**	6.88	6.98	8.03	7.68
3–15–5–0.8–50–80–20	7.58	7.49	10.78	6.93	**6.63**
3–15–10–0.6–100–80–20	5.21	5.17	**4.37**	5.13	4.64
3–15–10–0.8–50–60–20	14.38	14.42	**12.86**	14.29	14.31
3–15–10–0.8–100–60–20	9.14	9.26	**8.00**	9.31	9.78
Overall Average	**6.60**	**6.60**	6.61	6.74	6.83

Table 4
Comparison of average percentage deviations from the lower bound values for the RVND algorithms.

Problem Type	RVNS (1)	RVNS (2)	RVNS (3)	RVNS (4)	RVNS (5)	RVNS (6)
$M - N - T - Cut - \theta - MProb - MBal$	k_1: IM, k_2: SM	k_1: SM, k_2: IM	k_1: IM, k_2: FIM	k_1: SM, k_2: FIM	k_1: FIM, k_2: SM	k_1: FIM, k_2: IM
2–15–5–0.8–50–80–20	1.62	1.66	1.57	1.51	**1.43**	1.45
2–15–10–0.8–50–80–20	**2.38**	2.63	2.64	2.45	2.51	2.56
2–15–10–0.8–100–80–20	7.26	6.93	**6.92**	7.08	7.12	7.04
2–15–10–0.8–100–80–20	5.14	6.15	5.10	**5.03**	5.08	5.12
2–15–10–0.8–100–60–20	7.18	6.33	**5.88**	6.34	5.86	6.01
2–20–10–0.8–100–80–10	6.89	**6.50**	6.90	6.91	6.84	6.85
3–15–5–0.8–50–80–20	**6.59**	8.94	8.42	7.23	8.26	7.14
3–15–10–0.6–100–80–20	**4.33**	5.44	9.69	4.36	4.81	5.07
3–15–10–0.8–50–60–20	**12.81**	12.91	17.24	13.14	13.40	12.91
3–15–10–0.8–100–60–20	10.14	11.75	11.49	9.20	**7.96**	8.14
Overall Average	6.33	6.92	6.76	6.33	6.33	6.23

Conclusion

This study considers the solution of the Parallel Machine Capacitated Lot-Sizing and Scheduling Problem (PM_CLSP) with machine eligibility restrictions, sequence-dependent setup times and costs, setup carryover, different production capacities for each machine, and upper bound production limitations for products. It is the first time here that solution approaches based on Variable Neighborhood Search (VNS), Variable Neighborhood Descent (VND), and Reduced Variable Neighborhood Search (RVNS) methods are developed for the PM_CLSP.

As neighborhood strategies, we adapted three different moves from the single-machine case proposed by Almada-Lobo & James (2011). We also employed the constraint-handling techniques to accept infeasible solutions during the search with constraint violations such as capacity violation, unsatisfied demand, and violation of upper bound production quantity are penalized by the Near-Feasible Threshold (NFT) approach [44,46]. Also, the *Superiority of Feasible Solutions* (SFS) has been used to select solutions [10].

According to the computational results over the 100 benchmark problems that consist of 10 instances for six problem types, both VNS algorithms worked well in almost every problem type and have improved the best-known solutions in six problem types. Another variant of VNS, VND, has also worked well since we improved over the best-known solutions in three problem types. The performance quality of VNS is better than VND when the overall average solution values are considered. Lastly, six RVNS variants with different neighborhood structures were also tested. Most RVNS variants have obtained the best results, and nearly all the best-known solutions for each problem type have been improved. Finally, since the solution approaches in the literature are MIP-based, the solution complexity will increase when the problem size increases, making it more difficult to solve large-size problems. In this respect, this study shows that simple heuristics such as VNS, VND, and RVNS, in which there is no need to optimize parameters, can provide satisfactory results. For future work, algorithms presented in this study can be applied to various lot-sizing and scheduling problems in the literature with optional features such as backordering, backlogging, safety stock, and lost sales with more challenging problem sizes. Implementing the proposed algorithms to the stochastic version of the PM_CLSP can be a promising future research avenue.

Ethical approval

Ethics committee approval is not required.

Consent to publish

Authors confirm that the final version of the manuscript has been reviewed, approved, and consented to publication by all authors.

Funding

No funds, grants, or other support were received.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] B. Almada-Lobo, R. James, Neighbourhood search meta-heuristics for capacitated lot-sizing and sequence-dependent setups, Int. J. Prod. Res. 48 (2010) 861–878, https://doi.org/10.1080/00207540802446787

[2] B. Almada-Lobo, D. Klabjan, M.A. Carravilla, J.F. Oliveira, Single machine multi-product capacitated lot sizing with sequence-dependent setups, Int. J. Prod. Res. 45 (20) (2007) 4873–4894, https://doi.org/10.1080/00207540601094465

[3] C. Almeder, B. Almada-Lobo, Synchronisation of scarce resources for a parallel machine lotsizing problem, Int. J. Prod. Res. 49 (22–24) (2011) 7315–7335.

[4] J. de Armas, M. Laguna, Parallel machine, capacitated lot-sizing and scheduling for the pipe-insulation industry, Int J. Prod. Res. 58 (3) (2020) 800–817, https://doi.org/10.1080/00207543.2019.1600763

[5] P. Beraldi, G. Ghiani, A. Grieco, E. Guerriero, Rolling-horizon and fix-and-relax heuristics for the parallel machine lot-sizing and scheduling problem with sequence-dependent setup costs, Comput. Oper. Res. 35 (11) (2008) 3644–3656, https://doi.org/10.1016/j.cor.2007.04.003

[6] G.R. Bitran, H.H. Yanasse, Computational complexity of the capacitated lot size problem, Manag. Sci. 28 (10) (1982) 1174–1186, https://doi.org/10.1287/mnsc.28.10.1174

[7] D.M. Carvalho, M.C.V. Nascimento, Hybrid matheuristics to solve the integrated lot sizing and scheduling problem on parallel machines with sequence-dependent and non-triangular setup, Eur. J. Oper. Res. 296 (1) (2022) 158–173, https://doi.org/10.1016/j.ejor.2021.03.050

[8] A. Clark, M. Mahdieh, S. Rangel, Production lot sizing and scheduling with non-triangular sequence-dependent setup times, Int J. Prod. Res 52 (8) (2014) 2490–2503, https://doi.org/10.1080/00207543.2014.885662

[9] C.C. Coello, Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art, Comput. Methods Appl. Mech. Eng. 191 (2002) 1245–1287, https://doi.org/10.1016/S0045-7825(01)00323-1

[10] K. Deb, An efficient constraint handling method for genetic algorithms, Comput. Methods Appl. Mech. Eng. 186 (2–4) (2000) 311–338, https://doi.org/10.1016/S0045-7825(99)00389-8

[11] A. Dolgui, A.V. Eremeev, M.Y. Kovalyov, P.M. Kuznetsov, Multi-product lot-sizing and scheduling on unrelated parallel machines to minimize makespan, IFAC Proc. 42 (4) (2009) 828–833, https://doi.org/10.3182/20090603-3-RU-2001.0553

[12] R.F. Fachini, K.F. Esposto, V.C.B. Camargo, Glass container production planning with warm-ups and furnace extraction variation losses, Int. J. Adv. Manuf. Technol. 90 (1–4) (2017) 527–543, https://doi.org/10.1007/s00170-016-9369-7

[13] D. Ferreira, A.R. Clark, B. Almada-Lobo, R. Morabito, Single-stage formulations for synchronised two-stage lot sizing and scheduling in soft drink production, Int J. Prod. Econ. 136 (2) (2012) 255–265, https://doi.org/10.1016/j.ijpe.2011.11.028

[14] D.J. Fiorotto, S.A. de Araujo, Reformulation and a Lagrangian heuristic for lot sizing problem on parallel machines, Ann. Oper. Res. 217 (1) (2014) 213–231, https://doi.org/10.1007/s10479-014-1570-1

[15] D.J. Fiorotto, S.A. De Araujo, R. Jans, Hybrid methods for lot sizing on parallel machines, Comput. Oper. Res. 63 (2015) 136–148, https://doi.org/10.1016/j.cor.2015.04.015

[16] M. Ghirardi, A. Amerio, Matheuristics for the lot sizing problem with back-ordering, setup carry-overs, and non-identical machines, Comput. Ind. Eng. 127 (2019) 822–831, https://doi.org/10.1016/j.cie.2018.11.023

[17] M. Gopalakrishnan, D.M. Miller, C.P. Schmidt, A framework for modeling setup carryover in the capacitated lot-sizing problem, Int. J. Prod. Res. 33 (7) (1995) 1973–1988, https://doi.org/10.1080/00207549508904793

[18] M. Güngör, A.T. Ünal, Z.C. Taşkın, A parallel machine lot-sizing and scheduling problem with a secondary resource and cumulative demand, Int. J. Prod. Res. 56 (9) (2018) 3344–3357, https://doi.org/10.1080/00207543.2017.1406675

[19] D. Gupta, T. Magnusson, The capacitated lot-sizing and scheduling problem with sequence-dependent setup costs and setup times, Comput. Oper. Res. 32 (4) (2005) 727–747, https://doi.org/10.1016/j.cor.2003.08.014

[20] K. Haase, A. Kimms, Lot sizing and scheduling with sequence dependentsetup costs and times and efficient rescheduling opportunities, Int. J. Prod. Econ. 66 (2) (2000) 159–169, https://doi.org/10.1016/S0925-5273(99)00119-X

[21] P. Hansen, N. Mladenović, Hanafi S. Todosijević, Variable neighborhood search: basics and variants, Eur. J. Comput. Optim. 5 (3) (2017) 423–454, https://doi.org/10.1007/s13675-016-0075-x

[22] O.J. Ibarra-Rojas, R.Z. Ríos-Mercado, Y.A. Rios-Solis, M.A. Saucedo-Espinosa, A decomposition approach for the piece mold machine manufacturing problem, Int. J. Prod. Econ. 134 (1) (2011) 255–261, https://doi.org/10.1016/j.ijpe.2011.07.006

[23] R.J. James, B. Almada-Lobo, Single and parallel machine capacitated lotsizing and scheduling: New iterative MIP-based neighborhood search heuristics, Comput. Oper. Res 12 (38) (2011) 1816–1825, https://doi.org/10.1016/j.cor.2011.02.005

[24] R. Jans, Solving lot-sizing problems on parallel identical machines using symmetry-breaking constraints, INFORMS J. Comput. 21 (1) (2009) 123–136, https://doi.org/10.1287/ijoc.1080.0283

[25] W. Kaczmarczyk, Proportional lot-sizing and scheduling problem with identical parallel machines, Int. J. Prod. Res. 49 (9) (2011) 2605–2623, https://doi.org/10.1080/00207543.2010.532929

[26] G. Laporte, The vehicle routing problem: An overview of exact and approximate algorithms, Eur. J. Oper. Res. 59 (3) (1992) 345–358, https://doi.org/10.1016/0377-2217(92)90192-C

[27] G. Laporte, The traveling salesman problem: an overview of exact and approximate algorithms, Eur. J. Oper. Res 59 (2) (1992) 231–247, https://doi.org/10.1016/0377-2217(92)90138-Y

[28] F. Larroche, O. Bellenguez, G. Massonnet, Clustering-based solution approach for a capacitated lot-sizing problem on parallel machines with sequence-dependent setups, Int. J. Prod. Res. (2021) 1–24, https://doi.org/10.1080/00207543.2021.1995792

[29] J. Maes, L.N. Van Wassenhove, Multilevel capacitated lotsizing complexity and LP-based heuristics, European Journal of Operational Research 53 (1991) 131–148.

[30] M. Mahdieh, A. Clark, M. Bijari, A novel flexible model for lot sizing and scheduling with non-triangular, period overlapping and carryover setups in different machine configurations, Flex. Serv. Manuf. J. 30 (4) (2018) 884–923, https://doi.org/10.1007/s10696-017-9279-5

[31] F. Marinelli, M.E. Nenni, A. Sforza, Capacitated lot sizing and scheduling with parallel machines and shared buffers: a case study in a packaging company, Ann. Oper. Res. 150 (1) (2007) 177–192, https://doi.org/10.1007/s10479-006-0157-x

[32] G.R. Mateus, M.G. Ravetti, M.C. De Souza, T.M. Valeriano, Capacitated lot sizing and sequence dependent setup scheduling: an iterative approach for integration, J. Sched. 13 (3) (2010) 245–259, https://doi.org/10.1007/s10951-009-0156-2

[33] N. Mladenović, P. Hansen, Variable neighborhood search, Comput. Oper. Res 24 (1997) 1097–1100, https://doi.org/10.1016/S0305-0548(97)00031-2

[34] O. Moursli, Y. Pochet, A branch-and-bound algorithm for the hybrid flowshop, International journal of production economics 64 (1–3) (2000) 113–125.

[35] W. Qin, Z. Zhuang, Y. Liu, O. Tang, A two-stage ant colony algorithm for hybrid flow shop scheduling with lot sizing and calendar constraints in printed circuit board assembly, Comput. Ind. Eng. 138 (2019) 106–115, https://doi.org/10.1016/j.cie.2019.106115

[36] D. Quadt, H. Kuhn, Conceptual framework for lot-sizing and scheduling of flexible flow lines, Int J. Prod. Res. 43 (11) (2005) 2291–2308, https://doi.org/10.1080/00207540500066762

[37] Quadt D., Kuhn H., 2008. Capacitated Lot-sizing with Extensions: a review. 4or 6:61–83. https://doi.org/10.1007/s10288–007-0057–1.

[38] S. Radhakrishnan, J.A. Ventura, Simulated annealing for parallel machine scheduling with earliness-tardiness penalties and sequence-dependent setup times, Int. J. Prod. Res. 38 (10) (2000) 2233–2252, https://doi.org/10.1080/00207540050028070

[39] F. Riane, Scheduling hybrid flowshops: algorithms and applications. *Sciences de Gestion*, Facultés Universitaires Catholiques de Mons, Mons, 1998.

[40] M. Sambasivan, S. Yahya, A Lagrangean-based heuristic for multi-plant, multi-item, multi-period capacitated lot-sizing problems with inter-plant transfers, Comput. Oper. Res. 32 (3) (2005) 537–555, https://doi.org/10.1016/j.cor.2003.08.002

[41] S.C. Sarin, H.D. Sherali, L. Yao, New formulation for the high multiplicity asymmetric traveling salesman problem with application to the Chesapeake problem, Optim. Lett. 5 (2) (2011) 259–272, https://doi.org/10.1007/s11590-010-0205-y

[42] F. Seeanner, B. Almada-Lobo, H. Meyr, Combining the principles of variable neighborhood decomposition search and the fix&optimize heuristic to solve multi-level lot-sizing and scheduling problems, Comput. Oper. Res. 40 (1) (2013) 303–317, https://doi.org/10.1016/j.cor.2012.07.002

[43] C. Silva, J.M. Magalhaes, Heuristic lot size scheduling on unrelated parallel machines with applications in the textile industry, Comput. Ind. Eng. 50 (1–2) (2006) 76–89, https://doi.org/10.1016/j.cie.2006.01.001

[44] A.E. Smith, Genetic optimization using a penalty function, in: Fifth Proc (Ed.), International Conference on Genetic Algorithms, 1993, pp. 499–505.

[45] C. Suerie, H. Stadtler, The capacitated lot-sizing problem with linked lot sizes, Management Science 49 (8) (2003) 1039–1054.

[46] D.M. Tate, A.E. Smith, Unequal area facility layout using genetic search, IIE Trans. 27 (1995) 465–472, https://doi.org/10.1080/07408179508936763

[47] B. Vincent, C. Duhamel, L. Ren, N. Tchernev, A population-based metaheuristic for the capacitated lot-sizing problem with unrelated parallel machines, Int. J. Prod.

Res. 58 (21) (2020) 6689–6706, https://doi.org/10.1080/00207543.2019.1685699

[48] R.J. Wittrock, An adaptable scheduling algorithm for flexible flow lines, Operations research 36 (3) (1988) 445–453.

[49] J. Xiao, C. Zhang, L. Zheng, J.N.D. Gupta, MIP-based fix-and-optimise algorithms for the parallel machine capacitated lot-sizing and scheduling problem, Int. J. Prod. Res. (2013) 5011–5028, https://doi.org/10.1080/00207543.2013.790570

[50] J. Xiao, H. Yang, C. Zhang, L. Zheng, J.N.D. Gupta, A hybrid Lagrangian-simulated annealing-based heuristic for the parallel-machine capacitated lot-sizing and scheduling problem with sequence-dependent setup times, Comput. Oper. Res. 63 (2015) 72–82, https://doi.org/10.1016/j.cor.2015.04.010

[51] C. Zhang, D. Zhang, T. Wu, Data-driven branching and selection for lot-sizing and scheduling problems with sequence-dependent setups and setup carryover, Comput. Oper. Res. 132 (2021) 105289, https://doi.org/10.1016/j.cor.2021.105289

[52] X. Zhu, W.E. Wilhelm, Scheduling and lot sizing with sequence-dependent setup: a literature review, IIE Trans. 38 (11) (2006) 987–1007, https://doi.org/10.1080/07408170600559706