Applied Soft Computing 125 (2022) 109191

Contents lists available at ScienceDirect

Applied Soft Computing

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

# Variable neighborhood-based Cuckoo Search for production routing with time window and setup times

Gen-Han Wu [a], Chen-Yang Cheng [b], Pourya Pourhejazy [c,*], Bai-Lyn Fang [d]

[a] Department of Industrial Engineering and Management, Yuan Ze University, Taoyuan 32003, Taiwan
[b] Department of Industrial Engineering and Management, National Taipei University of Technology, Taipei 10608, Taiwan
[c] Department of Industrial Engineering, UiT - The Arctic University of Norway, Lodve Langsgate 2, Narvik 8514, Norway
[d] Graduate Institute of Logistics Management, National Dong Hwa University, Hualien 97401, Taiwan

ARTICLE INFO

*Article history:*
Received 22 November 2021
Received in revised form 1 June 2022
Accepted 13 June 2022
Available online 23 June 2022

*Keywords:*
Supply chain
Process integration
Vehicle routing
Distribution
Production scheduling
Optimization

ABSTRACT

Major corporations compete over the strengths of their supply chains. Integrating production and distribution operations helps improve supply chain connectedness and responsiveness beyond the standalone optimization norms. This study proposes an original Mixed-Integer Linear Programming (MILP) formulation for the Production scheduling-based Routing Problem with Time Window and Setup Times (PRP-TWST). For this purpose, the identical parallel machine scheduling is integrated with the vehicle routing problem. Considering the highly intractable solution spaces of the integrated problem, hybrid metaheuristics based on the Variable Neighborhood Search (VNS), Particle Swarm Optimization (PSO), and Cuckoo Search (CS) algorithms are developed to solve the PRP-TWST problem. Extensive numerical experiments are conducted to evaluate the effectiveness of the developed algorithms considering the total delay time as the objective function. The results are supportive of the VNS-based CS algorithm's effectiveness; the developed metaheuristics can be considered strong benchmarks for further developments in the field. This study is concluded by suggesting directions for modeling and managing integrated operations in the supply chain context.



## 1. Introduction

The supply chain concept revolves around the connectedness of business operations, from the acquisition of raw materials to production and the delivery of goods to the final consumer. In this process, no entity can operate at its best capacity until all other entities are performing well. Disruption propagation across the supply chain, i.e. the ripple effect [1], is prime evidence of the extent of interdependencies among supply chain operations and entities. Supply chain integration not only helps reduce operational inefficiencies [2] but also improves organizational performance, and environmental factors [3], and facilitates Industry 4.0 adoption [4]. Optimization problems are of high relevance for establishing supply chain integration [5].

With many application areas from the movement of raw material, work-in-progress, and final goods to courier services and reverse logistics, vehicle routing constitutes a major aspect of supply chain optimization [6]. The scope of routing decisions has been extended in various ways to facilitate supply chain integration. Acknowledging the reciprocal influence between some of the supply chain operational, tactical, and strategic decisions [7,8], routing decision has been integrated with inventory management and location–allocation decisions (see [9] for inventory routing and [10] for location routing). It is widely recognized that the simultaneous optimization of these operations results in a significant performance improvement [11,12]. Overall, these routing extensions are suitable for optimizing service supply chains or when the goods are produced by a third party and sold on e-commerce platforms.

The Production-scheduling-based Routing Problem, hereafter denoted by PRP, is another class of integration with the manufacturing supply chain being its main application area (i.e., when production and distribution activities are managed by the same supply chain entity). PRPs are particularly important in the supply chain optimization of time-sensitive and perishable products [13] where timeliness and connectedness are of high relevance. In an uncoordinated approach, production scheduling takes place first and the outcomes will be a basis for planning the distribution operations. Isolated and sequential optimization of the production and distribution operations may result in the following situations. First, the distribution operations may be planned based on wrong or infeasible input data, e.g., the delivery may be scheduled for an order that is yet to be produced. This situation results in infeasible solutions. The second situation arises when a lack of coordination

* Corresponding author.
*E-mail addresses:* genhanwu@saturn.yzu.edu.tw (G.-H. Wu), cycheng@ntut.edu.tw (C.-Y. Cheng), pourya.pourhejazy@uit.no (P. Pourhejazy), 610437011@ems.ndhu.edu.tw (B.-L. Fang).

https://doi.org/10.1016/j.asoc.2022.109191

between the two major phases of the supply chain process results in sub-optimal solutions. For example, a customer order with less urgency may be prioritized in the production stage earlier than a more urgent order; this situation results in poor responsiveness, operational burden, and, in some cases, avoidable costs.

PRP is a relatively new extension to the renowned vehicle routing problems; the authors of Ref. [14] have recently conducted an exhaustive review and taxonomy analysis of the published works. They found that most of the existing problems considered a single-machine production environment, which, often, cannot reflect the real situation. Given that routing decisions require optimization tools when the number of delivery tasks is relatively large, it would be simplistic to assume that a large quantity of deliverables is produced in a single-machine production setting. Besides, considering setup times, i.e., the preparation tasks required before the production process takes place, is another practical scheduling setting that should be considered; only a slight minority of the problems accounted for the setup time feature despite its prevalence in production environments with general-purpose machinery [15]. From the existing limited studies, Ref. [16] is the first to develop a PRP with setup times in a parallel machine production environment; they considered total cost minimization as the objective of the problem to investigate a trade-off between setup cost, transportation cost, and the quality of the perishable food products. Later, the authors of Ref. [17] studied the PRP with setup times for minimizing total cost to investigate lot-sizing versus batching decisions in the supply chain of perishable goods. Developing a general heuristic, the total cost was considered by Ref. [18] as the objective function for optimizing PRPs with setup times. More recently, Ref. [19] developed an Iterated Greedy algorithm to solve the PRP problem with setup times considering the maximum completion time, and makespan measure. None of these studies accounted for the time window constraints at the distribution end of the operations while improving the timeliness of the operations is the main purpose of the integrated planning of production and distribution operations.

The only studies that simultaneously account for the time window and setup time features are Refs. [20,21] where the total cost is minimized in single-machine and unrelated parallel machine environments, respectively. The former study developed a sequential solution approach with no feedback between the consecutive optimization steps, which reduces the effectiveness of the integrated problem. The latter study proposed an exact algorithm that can only solve small-scale problems. Besides, they developed two separate formulations for the production scheduling and vehicle routing decisions. To the best of the authors' knowledge, the PRP with identical parallel machines considering the time window and setup time constraints has not been studied. Besides, timeliness has not been considered as the optimization criterion despite being the main cause for integrating production and distribution planning. A twofold contribution inspired by the above research gaps is put forward. As its first contribution, this study proposes an original Mixed-Integer Linear Programming (MILP) formulation to the Production scheduling-based Routing Problem with Time Window and Setup Times (PRP-TWST) with total delay time being the objective function in an identical parallel machines production setting. The mathematical model is validated using an exact solver for solving small-size test instances. Second, two improved metaheuristics are developed to solve the proposed optimization problem for large-scale instances. In the first solution algorithm, the Variable Neighborhood Search (VNS) is integrated into the Cuckoo Search (CS) algorithm, hereafter named VNS-CS. The Particle Swarm Optimization (PSO) algorithm is also enhanced through VNS integration (VNS-PSO).

The remainder of this manuscript begins with a review of the relevant literature in Section 2. Section 3 presents a new mathematical formulation of the problem. An elaboration on the proposed solution algorithms can be found in Section 4. Numerical results are presented next in Section 5 to evaluate the applicability of the model and the effectiveness of the solution algorithms. Finally, Section 6 provides the concluding remarks and directions for future research on the integrated scheduling–routing problems.

## 2. Relevant literature

This section reviews the most relevant journal articles on PRP. For this purpose, the production settings, mathematical extension, objective function, and the solution algorithm are considered. For a comprehensive review of the PRP literature, the interested readers are encouraged to follow the recent review articles by [14,22].

Ref. [23] proposed a non-linear formulation of the PRP in a single-machine production environment and minimized the maximum order delivery time using a Genetic Algorithm. The authors of Ref. [24] proposed a multi-factory variant of the PRP with each factory operating in a single-machine production setting; the authors developed an Imperialist Competitive Algorithm to solve the problem of minimizing total cost. Ref. [16] proposed a PRP with setup times considering a parallel machine production setting to minimize total cost. They used a math-heuristic algorithm that first solves the production scheduling part of PRP using an exact method, and then, uses the generated solution as input parameters in the metaheuristic algorithm to optimize the routing part of the problem. Ref. [17] studied the PRP with setup times for minimizing total cost and used a commercial optimizer to solve the problem considering small instances. Ref. [25] applied the PRP with identical parallel machines for a new application area, i.e. automated guided vehicles; they developed a Lagrangian Relaxation-based solution algorithm to minimize total weighted tardiness. The authors of Ref. [18] developed a general heuristic to minimize the total cost when solving PRP with setup times considering a production system with an identical parallel machine. Ref. [19] solved the PRP problem with setup times using an Iterated Greedy algorithm considering the maximum completion time, makespan as the optimization measure. Ref. [26] developed a metaheuristic solution algorithm to minimize total cost in PRP with flexible departure times and identical parallel machines.

More recent studies considered multi-objective optimization approaches for solving PRP with conflicting objectives in a single-machine production environment. Ref. [27] integrated inventory management decisions into PRP and developed a level-based multi-objective particle swarm optimization (PSO) for minimizing total cost and tardiness. Ref. [28] adapted the Non-Dominated Sorting Genetic Algorithm II to minimize carbon emissions along with total cost in the PRP within the time window. Ref. [29] developed a Monte Carlo-based hyper-heuristic algorithm for minimizing total cost and maximizing customers' purchasing probability in the basic PRP. None of these studies simultaneously accounted for setup time in the production stage and time window in the distribution stage.

In the most relevant works, the authors of [20] developed a Branch-Price-and-Cut algorithm for total cost minimization in a single-machine production setting considering time window and setup time. Ref. [30] developed a decomposition-based Fix-and-Optimize approach to solve the PRP with time-window and family setup times considering a similar production setting. Ref. [21] developed a two-phase iterative heuristic algorithm for the sequential optimization of the production and distribution operations; the scheduling formulation accounts for job-splitting while considering time window and setup time features and total cost

minimization; this setting is prevalent in unrelated parallel machines. The proposed method in the next section is different from the most relevant works in the following points. (1) Production scheduling and vehicle routing decisions are included in one integrated formulation. (2) Production environment with identical parallel machines is investigated; scheduling jobs on parallel machines while considering delivery time window is of practical relevance in consumer goods industries, like beverage production where jobs are scheduled on parallel bottling machines and on-time delivery is important for the restaurants and pubs to avoid possible shortage. The printing and pharmaceutical industries are other examples of this type of application area of an identical parallel machine scheduling problem [31]. (3) Total delay time is considered as the objective function; this emphasizes responsiveness, which is in contrast with the cost-effective nature of the above studies. Taking the book publishing industry as a possible application of PRP-TWST; printing tasks from different publishers must be completed on parallel printers in the production stage and on-time delivery is critical for the publisher to ensure timely shelf availability.

## 3. Mathematical formulation

This section presents a new MILP formulation to PRP-TWST. For a formal definition of the problem, let assume a manufacturing supply chain that accommodates products for fulfilling orders of size $u_i$, where $i \in \{1, \ldots, n\}$, and every order is associated with a time window, $[a_i, b_i]$. Assuming that the production and distribution operations are both managed internally, the former operations should be scheduled on identical parallel machines, $k \in \{1, \ldots, m\}$, and the latter operations should be planned considering a fleet of heterogeneous trucks, $v \in \{1, \ldots, f\}$.

Assuming that the raw material is available at time zero, the production process can begin immediately after the commencement of the scheduling period. In the production stage, each order can be processed on one machine at a time with the processing time ($p_i$) being job-dependent; an ongoing process on a machine cannot be interrupted until the operation is complete. A setup time, $S_{ij}$, is defined to account for the sequence-dependent preparation time required before processing job $j$ after job $i$. It is assumed that the machines' required maintenance can be completed within the defined setup time. In the distribution stage, a deterministic travel time is associated with every path connecting two consecutive nodes, $t_{ij}$. It is worthwhile mentioning that the time gap between the completion of the jobs at the production stage and the start of delivery is assumed to be negligible. That is, the products can be assigned to delivery trucks immediately after the completion of their production. The indices, parameters, and decision variables for this problem are defined in Table 1.

Given these notations, the MILP formulation to the PRP-TWST problem is as follows.

$$\textit{Minimize } z = \sum_{i=1}^{n} T_i \tag{1}$$

*Subject to* :

$$\sum_{k=1}^{m} O_i^k = 1, \forall i \in \{1, \ldots, n\} \tag{2}$$

$$x_{ij}^k \le O_i^k; \forall i, j, k; i \ne j \tag{3}$$

$$x_{ij}^k \le O_j^k; \forall i, j, k; j \ne i \tag{4}$$

$$C_i \ge S_{0i} \cdot O_i^k + p_i \cdot O_i^k; \forall i \in \{1, \ldots, n\} \tag{5}$$

$$C_j - C_i + M \cdot \left(3 - x_{ij}^k - O_i^k - O_j^k\right) \ge S_{ij} + p_j; \forall i, j \in \{1, \ldots, n\}, \quad k \in \{1, \ldots, m\}, i \ne j \tag{6}$$

$$C_i - C_j + M \cdot \left(2 + x_{ij}^k - O_i^k - O_j^k\right) \ge S_{ji} + p_i; \forall i, j \in \{1, \ldots, n\}, \quad k \in \{1, \ldots, m\}, i \ne j \tag{7}$$

$$\sum_{v=1}^{f} V_i^v = 1; \forall i \in \{1, \ldots, n\} \tag{8}$$

$$y_{ij}^v \le V_i^v; \forall i, j \in \{1, \ldots, n\}, v \in \{1, \ldots, f\}, i \ne j \tag{9}$$

$$y_{ij}^v \le V_j^v; \forall i, j \in \{1, \ldots, n\}, v \in \{1, \ldots, f\}, i \ne j \tag{10}$$

$$R_{0i}^v - C_j + M \cdot \left(2 - V_i^v - V_j^v\right) \ge 0; \forall i, j \in \{1, \ldots, n\}, \quad v \in \{1, \ldots, f\} \tag{11}$$

$$R_{0i}^v \le M \cdot V_i^v; \forall i \in \{1, \ldots, n\}, v \in \{1, \ldots, f\} \tag{12}$$

$$R_{0i}^v - R_{0j}^v \le M \cdot \left(2 - V_i^v - V_j^v\right); \forall i, j \in \{1, \ldots, n\}, v \in \{1, \ldots, f\} \tag{13}$$

$$D_i \ge R_{0i}^v + t_{0i} \cdot V_i^v; \forall i \in \{1, \ldots, n\}, v \in \{1, \ldots, f\} \tag{14}$$

$$D_j - D_i + M \cdot \left(3 - y_{ij}^v - V_i^v - V_j^v\right) \ge t_{ij}; \forall i, j \in \{1, \ldots, n\}, \quad v \in \{1, \ldots, f\}, i \ne j \tag{15}$$

$$D_i - D_j + M \cdot \left(3 - y_{ij}^v - V_i^v - V_j^v\right) \ge t_{ji}; \forall i, j \in \{1, \ldots, n\}, \quad v \in \{1, \ldots, f\}, j \ne i \tag{16}$$

$$\sum_{i=1}^{n} u_i V_i^v \le cap_v; v \in \{1, \ldots, f\} \tag{17}$$

$$D_i \ge a_i; \forall i \in \{1, \ldots, n\} \tag{18}$$

$$T_i \ge D_i - b_i; \forall i \in \{1, \ldots, n\} \tag{19}$$

$$T_i, D_i, C_i \ge 0; \forall i \in \{1, \ldots, n\}$$

$$R_{0i}^v \ge 0; \forall i \in \{1, \ldots, n\}, v \in \{1, \ldots, f\}$$

$$x_{ij}^k \in \{0, 1\}; \forall i, j \in \{1, \ldots, n\}, \forall k \in \{1, \ldots, m\} \tag{20}$$

$$y_{ij}^v \in \{0, 1\}; \forall i, j \in \{1, \ldots, n\}, \forall v \in \{1, \ldots, f\}$$

$$O_i^k \in \{0, 1\}; \forall i \in \{1, \ldots, n\}, \forall k \in \{1, \ldots, m\}$$

$$V_i^v \in \{0, 1\}; \forall i \in \{1, \ldots, n\}, \forall v \in \{1, \ldots, f\}$$

The objective function in Eq. (1) aims to minimize the total delay time for the delivery of the orders to the final customer. This optimization objective is subject to the following production- and distribution-related constraints.

Eq. (2) restricts order $i$ from being assigned to more than one machine for processing in the production stage. Constraints (3)–(4) associate the sequence planning and assignment decision variables, making sure that orders $i, j$ are consecutively processed on production machine $k$. According to Constraints (5), the completion time of order $i$ on production machine $k$ should be larger than or equal to the summation of the setup and processing times. Given '$M$' as a very large positive number, Constraints (6)–(7) are used to determine the completion time of the production process of an order considering that of the earlier order on the same machine. These constraints also associate the binary assignment variables with the respective completion time variables.

Eq. (8) ensures that every order is assigned to one and only one vehicle. Constraints (9)–(10) determine the sequence of orders and associate the time and assignment decision variables. Establishing a link between the production and distribution tasks, Constraints (11) ensures that the delivery process of a certain order begins only after the production of the same item is completed. Besides, it ensures that orders $i$ and $j$ are assigned to the same vehicle. Constraints (12) associate two binary variables, where there cannot be a departure time for order $i$ on vehicle $v$ if it is not assigned to the vehicle. Constraints (13) determine that consecutive orders are delivered using the same vehicle. Given orders assigned to vehicle $k$ on a tour, Constraints (14) calculates the delivery time for every delivery. Constraints (15)–(16) are

**Table 1**
Mathematical notations.

| Type | Symbol | Description |
|---|---|---|
| Index | $i, j$ | Order indices where $i, j \in \{1, \ldots, n\}$ |
| | $k$ | Machine tag where $k \in \{1, \ldots, m\}$ |
| | $v$ | Vehicle number where $v \in \{1, \ldots, f\}$ |
| Parameter | $p_i$ | Production (processing) time of order $i \in \{1, \ldots, n\}$ |
| | $S_{ij}$ | Setup time of order $j \in \{1, \ldots, n\}$ when it is processed after $i \in \{1, \ldots, n\}$;for index '0' indicates the dummy job that is assigned to every machine before processing the first order. The dummy job has a processing time and latest time equal to zero |
| | $t_{ij}$ | Transportation time from the location at which order $i$ is delivered to the designated delivery location of order $j$; index '0' indicates the factory location |
| | $u_i$ | Demand size of order $i \in \{1, \ldots, n\}$ |
| | $a_i$ | Earliest time for delivering order $i \in \{1, \ldots, n\}$ |
| | $b_i$ | Latest time for delivering order $i \in \{1, \ldots, n\}$ |
| Decision variable | $C_i$ | Completion time of order $i$ in the production stage. |
| | $D_i$ | Delivery time of order $i$ in the distribution stage. |
| | $x_{ij}^k$ | The sequence of processing orders in the production stage; = 1 if order $j$ is processed after order $i$ on machine $k$; = 0, otherwise |
| | $y_{ij}^v$ | The sequence of delivering orders at the distribution stage; = 1 if order $j$ is delivered after order $i$ using vehicle $v$; = 0, otherwise |
| | $O_i^k$ | Machine assignment variable: $O_i^k = 1$ if order $i$ is assigned to machine $k$; $O_i^k = 0$, otherwise |
| | $V_i^v$ | Vehicle assignment variable: $V_i^v = 1$ if order $i$ is assigned to vehicle $v$; $V_i^v = 0$, otherwise |
| | $R_{0i}^v$ | Departure time from the factory for delivering order $i$ using vehicle $v$ |
| | $T_i$ | Delaying time of delivering order $i$ |

defined to enforce the sequence of deliveries and associate it with the respective delivery times. Constraints (17) ensured that the capacity of the vehicle is not violated. Constraints (18)–(19) ensure the timeliness of the deliveries. On this basis, Constraints (18) restrict the driver from early visits while Constraints (19) determine the total delay time, which is sought to be minimized in the objective function. Finally, the last constraints determine whether a decision variable accepts integer or binary values.

## 4. Solution algorithms

Unlike the exact solution methods that solve the optimization problem through exploring its gradient, metaheuristic algorithms do not require the problem to be differentiable. This approach is particularly useful when integrated problems, like the one studied in this study are investigated. This study proposes two hybrid metaheuristics based on the VNS, CS, and PSO algorithms, which are widely used in the production scheduling and vehicle routing literature. This section begins with briefing the CS and PSO algorithms to present the structure of the main loop. We then delve deeper into the initialization, encoding, and neighborhood search methods. Finally, the proposed hybridization schemes are presented.

### 4.1. Cuckoo search

The CS algorithm [32] is a population-based metaheuristic inspired by the brood parasite of cuckoo species and the Levy flight of birds. The algorithm has gained recent popularity due to its simplicity and ease of implementation, requiring few parameter settings, as well as its effectiveness in solving a variety of engineering optimization problems, like vehicle routing and production scheduling problems [33]. In this method, a set of $N$ nests (solutions) are randomly distributed in the search space. The cuckoo bird (new solution) is substantiated using Levi flight. The cuckoo bird randomly selects a nest and locates the egg. The host may or may not recognize an alien egg. This procedure relates to the new solution evaluation in the optimization algorithm. If the host recognizes the alien egg, two scenarios will take place; (1) the host will throw away the egg, or (2) the host will abandon the nest. The former situation is equivalent to comparing a new solution against the current solution and discarding the new solution if it is not strictly better. Discarding a fraction ($Fr_a \in (0, 1)$) of the worst nests at the end of every iteration is in direct association with case (2). The computational steps of the CS algorithm are summarized below.

**Step 1.** Initialize (locate) a population of $N$ solutions (host nests) in random locations and calculate their fitness values, $f_i$. Solution encoding is detailed in Section 4.4.

**Step 2.** Generate a new solution (cuckoo; $i$) using Levi flights and calculate its fitness, $f_i$.

**Step 3.** Select a random nest, $j$, from the nest population for locating the cuckoo egg; let $j$ be the solution if $f_j$ is better than $f_i$, otherwise, replace it with the new solution, $i$.

**Step 4.** Save the best solutions (nests) and replace a fraction $Fr_a$ of the worst solutions with new ones in new locations using Levi flight.

**Step 5.** Return to Step 2 if the maximum computation time is not reached.

Levi flight consists of perturbing the current solution, $x_{current}$, to generate a new solution, $x_{new}$. This procedure is done using Eq. (21).

$$x_{new} = x_{current} + rnd \cdot c \tag{21}$$

In this equation, $rnd$ is a random number generated by a normal distribution and $c$ indicates the extent of change/perturbation in the current solution, which is calculated by $c = 0.01 \cdot s \cdot (x_{current} - x_{nbest})$, where $x_{nbest}$ represents the best solution in the nest and $s$ indicates the step size generated using symmetric Levy Distribution.

### 4.2. Particle swarm optimization

The PSO algorithm [34] is also a population-based metaheuristic inspired by the social interaction and experience-sharing behavior of the members of a society, the so-called swarm intelligence. PSO has a successful track record in solving logistics and workflow scheduling problems [35]. Considering a set of $N$ particles (solutions) that are randomly distributed in the search space, each of the particles moves with a certain velocity searching for the global best. Therefore, their starting position and

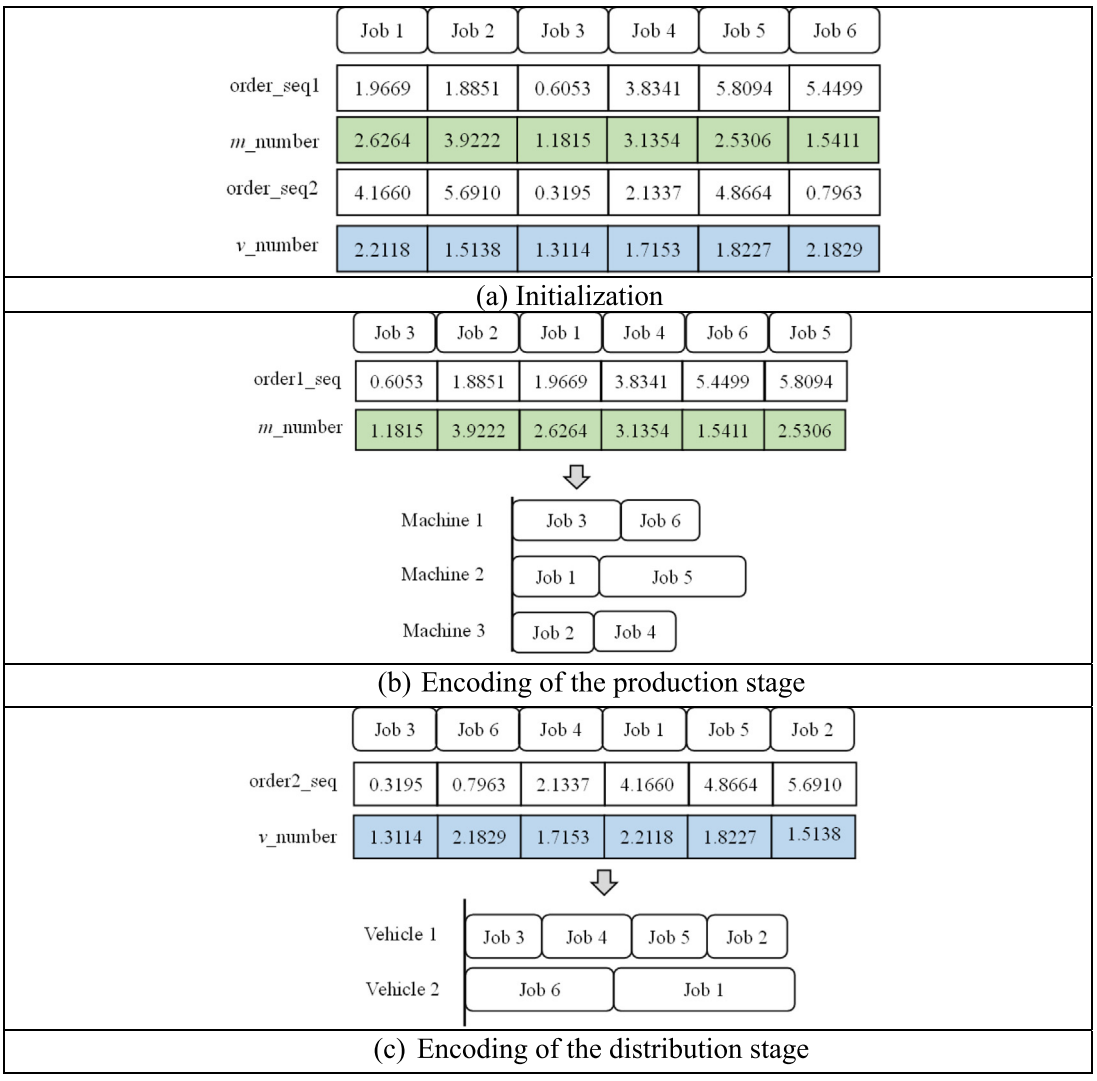


**Fig. 1.** Illustration of the solution initialization/encoding procedure.

velocity should be initialized in the first iteration and get updated over the next iterations. To update the velocity and direction of movement, it is assumed that a particle accelerates its movement considering its personal best positional experience (cognitive; individual best), and that of the group (social; global best). This concept can be expressed by Eqs. (22)–(23), based on which the position and velocity of the particles, respectively, are updated.

$$Position_i^{t+1} = Position_i^t + Velocity_i^{t+1} \tag{22}$$

$$Velocity_i^{t+1} = \omega Velocity_i^t + c_1 r_1 \left(Position_{pbest(i)}^t - Position_i^t\right) + c_2 r_2 \left(Position_{gbest*}^t - Position_i^t\right) \tag{23}$$

The first component of Eq. (23) refers to the inertia of the particle; $\omega$ presents the inertia weight that governs the particles' ability to change the movement direction. A small value of $\omega$ helps exploit the best solution while a large $\omega$ improves the exploration around the best solutions. The second and third parts of Eq. (23) specify the personal and group experience with $c_1, c_2$ being the cognitive and social coefficients, respectively, and $r_1, r_2$ are random values between zero and two, which are unique to particle/iteration. Overall, the best solution by particle $i$, denoted by $pbest\ (i)$, and the global best found by the particles swarm up to a certain iteration ($gbest*$) and the velocity in the current iteration ($t$) determine the new velocity in the new iteration, $t+1$. The computational steps of the PSO algorithm are summarized below. It is worthwhile noting that the control parameters of the PSO are $N$, $\omega$, $c_1$, $c_2$, which, together, regulate the exploration and exploitation levels of the search procedure. These parameters should be initialized before starting the algorithm procedure.

**Step 1.** Initialize the population and calculate the fitness value of every particle. The encoding procedure is explained in Section 4.4.

**Step 2.** Find the personal best for every particle.

**Step 3.** Find the global best for the population.

**Step 4.** Update the velocity and position of every particle.

**Step 5.** Check for the stopping criterion, i.e., the maximum computational time; return to Step 2 if the condition is not met.

### 4.3. Initialization and encoding methods

Real, cycle, and float methods are used for the solution encoding in the developed algorithms. For initializing solutions, random solutions are generated using Uniform distribution in four vectors: the production sequence (*order_seq1*) with $U[0, j)$, the corresponding processing machine (*m_number*) with $U[1, m+1)$, the distribution sequence (*order_seq2*) with $U[0, j)$, and the corresponding delivery vehicle (*v_number*) with $U[1, v+1)$. Fig. 1(a) shows the encoded initial solution for an example with 6 jobs that must be processed on 3 machines and delivered using 2 vehicles.

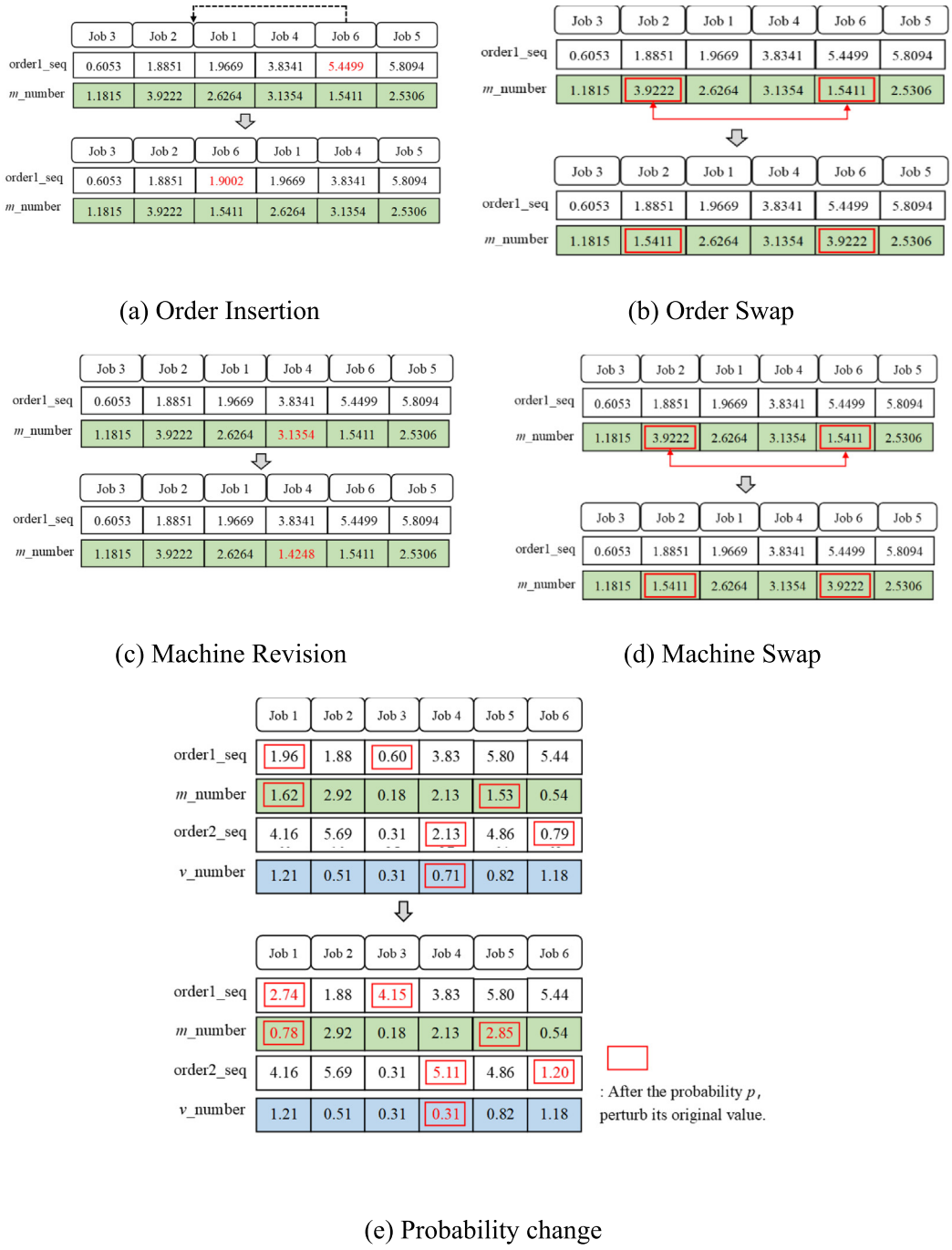


**Fig. 2.** Visual representation of the neighborhood search and perturbation mechanisms.

Once the random numbers are generated, the *order_seq1* and *order_seq2* should be arranged in descending order based on which, the *m_number* and *v_number* vectors should be reorganized. In this example, Jobs 3, and 6 are arranged on machine 1, Jobs 1, and 5 are arranged on machine 2, and Jobs 2, and 4 are arranged on machine 3, as shown in Fig. 1(b). The same procedure applies to the distribution vector, as shown in Fig. 1(c).

In the above encoding procedure, a real number variant is used for generating random numbers, where a new value should be generated if the initial random value exceeds the allowed range. As an alternative, the cycle value encoding can be used for dealing with the mentioned situation. Let us assume that the number of orders is 20, that is, the allowed range for *order1_seq* is between 0 and 20. In the real encoding method, if the initial value is 25.17 (falls outside of the allowed range), a new random value is generated using $U[0, 20)$. In the cycle encoding method, the new value is computed by (*initial value*−*max value* + *min value*), if the code value is greater than the max value of the range. Otherwise, the new value is computed by (*initial value* + *max value*−*min value*). That is, 25.17 is over the range, hence, the new value is $25.17 - 20 + 0 = 5.17$.

In the real and cycle methods, four numbers were used to represent each solution. In the float encoding method, however, only two numbers are required. That is, a single numerical code includes both the task and the assignment values; the integer part shows the machine/vehicle number while the decimal point value corresponds to the sequencing value (production/distribution). For example, the float value of 1.74 shows that the job processing

**VNS** ($N_k, k_{max}$)
**Begin:**
  $X = X_0$ ; /* Initial solution*/
  $k = 1$
  ***while*** (stop criteria) ***do*** /* Maximum iteration */
    Search for new solution, $X_1$ within $N_k$ ;
    ***if*** $(obj(X_1) < obj(X_0))$ ***do*** /* $X_0$ is *improved* */
      $X = X_1$
    ***else***
      $k = k + 1$;
      ***if*** $k = k_{max}$ ***do***
        $k = 1$;
      ***end if***
    ***end else***
    ***end if***
    Generate a new solution using the perturbation mechanism
  ***end while***
**Output:** $X$ current best solution, $obj\ (X)$

**Fig. 3.** Pseudocode of the variable neighborhood search algorithm.

sequence value is 0.74 and the corresponding machine is machine 1. In dealing with the random values that fall outside of the allowed interval, a similar method to the cycle value coding is applied. Assuming that the value becomes 8.64 after applying the Levy flight in the CS or the Position Movement in the PSO algorithm, the integer value 8 is over the number of machines 5. Hence, the value should be adjusted to $8 - 5 + 1 = 4$.

### 4.4. Neighborhood search and perturbation methods

The VNS algorithm [36] operates based on the systematic change of neighborhood applying local search moves bearing in mind the following facts [37]. First, a local optimum, which is found within one neighborhood structure, is not essentially optimal for another neighborhood structure. Second, a global optimum can be considered a local optimum for all conceivable neighborhood structures. Third, local optima of one or several neighborhood structures are usually close to each other. These characteristics together have made VNS a popular local search in combinatorial and global optimization [37], more particularly for solving PRPs [38].

The VNS method works with a given set of neighborhood structures hereafter denoted by $N_k$ ($x$). In this definition, $N_1$ ($x$) is the set of solutions in the first neighborhood of $x$. It is also known that every solution can have a maximum number of $k_{\max}$ neighborhood solutions. Given this information, the VNS procedure applies *Swap* and *Insertion* moves to the neighborhoods of every input solution to find better alternatives. The *Swap* method selects two random positions in the solution vector and exchanges their corresponding numbers. The *Insertion* method selects two random positions in the solution vector, removes the number in the first position, and inserts it next to the second position. The *Swap* and *Insertion* moves can be applied randomly either on the job and/or delivery orders in the production and distribution stages, respectively. Besides, *Swap* and *Revision* moves are applied for modifying machine and vehicle vectors. The neighborhood search moves shown in Fig. 2(a–d) are applied separately for displacing orders in the production stage and the distribution stage. To avoid local optimality traps, a perturbation mechanism, named Probability Change (Fig. 2e) is further considered at the end of the VNS procedure. Every element of the vectors is associated with a probability, $p$; in the perturbation mechanism, the original value is randomly changed to another new value.

The features of VNS make it an effective local search algorithm, which can be coupled with a global search metaheuristic to improve its exploitation capability. The pseudocode of this procedure is provided in Fig. 3.

### 4.5. Proposed hybridizations

CS and PSO are well-known global-search algorithms with solid exploration power [39,40]. However, they are both prone to getting trapped in local optima. Incorporating a local search module within the global search procedure enhances the exploitation power of the solution algorithm and is expected to result in better optimization outcomes. Two hybridization schemes are proposed to alleviate the early convergence and local optima issues.

The first proposed scheme incorporates the VNS module into the CS algorithm. In the VNS-CS algorithm, every iteration begins with assigning the solutions (birds) to unique positions. The solutions are then fed into the neighborhood search module to seek better alternatives considering the maximum number of iterations, *Max_It*. The best alternative from the VNS module is then sent [back] to the main CS loop where the global search procedure continues using Levi's flying procedure. This procedure continues until the stopping condition is met. The maximum CPU time is considered as the stopping condition of the solution algorithms. Fig. 4 presents the pseudocode of the VNS-CS algorithm.

The second hybridization scheme integrates the VNS module into the PSO algorithm. In the VNS-PSO algorithm, a group of solutions (particles) are used as inputs to the VNS module for the neighborhood search procedure that applies for *Max_It* iterations. The speed and position of the particles are then updated after every iteration. Next, the solutions (particles) with updated speed and position are sent to the main loop where the PSO algorithm directs the search into more promising search spaces using Eqs. (22)–(23). This procedure will be completed when the maximum computation time is reached. The pseudocode of

**VNS-CS** ($N_k$, Max_It, $N, \alpha, \beta, P_a$)

**Begin:**
  $k = 1$ ;
  Initialize a population of $N$ host nests ($i = 1 \sim N$)
  Evaluate fitness value, $obj(X_i)$ of every nest $i$
  Record initial global best solution as $gbest^*$
  ***while*** (Current time T < Threshold_CS)
    Get a cuckoo ($i$) using L$\acute{e}vy\ flight$
    Evaluate its objective value $obj(X_i)$
    Choose a nest among 1~N ($l$) randomly
    ***if*** ( $obj(X_i) < obj(X_l)$ )
      Replace solution $l$ with solution $i$
    ***end if***
    Abandon a fraction $Fr_a$ of worse nests
    Build new ones at new locations using L$\acute{e}vy\ flight$
    Keep the relative best solution
    Rank the solutions and find the current global best (g$best$)
    ***if*** (g$best < gbest^*$)
      Replace $gbest^*$ by g$best$
    ***end if***
    ***for*** (nests $i = 1 \sim N$)
      ***while*** (iteration < Max_it) ***do***
        Search new solution $X_1$ by $N_k$ ;
        ***if*** ($obj(X_1) < obj(X_0)$) $X_0$ is *improved* ***do***
          $X = X_1$
        ***else***
          $k = k + 1$;
          ***if*** $k = k_{max}$ ***do***
            $k = 1$;
          ***end if***
        ***end else***
          Generate a new solution by shaking mechanism
        ***end if***
        iteration = iteration + 1
      ***end while***
      Output current best solution and $obj(X_i)$
    ***end for***
  ***end while***
**Output:** $gbest^*$
**End**

**Fig. 4.** Pseudocode of the VNS-CS algorithm.

the VNS-PSO algorithm is provided in Fig. 5. It is worthwhile noting that the VNS-PSO algorithm considers the best individual solution(s) over the past iterations in addition to the current population's best solution; this is in contrast with the VNS-CS algorithm which only considers the population's best solution in every iteration.

## 5. Numerical experiments

This section elaborates on the analysis of the experimental results. The test instances and performance measures are first explained. The parameters of the metaheuristic algorithms are then calibrated to ensure the best computational performance. Next, the Gurobi optimizer is used to solve and validate the PRP-TWST problem considering small test instances. The final analysis of the results then follows to conclude the experiments. VNS-CS and VNS-PSO algorithms are both coded and compiled on Microsoft Visual Studio C++ using a personal computer with the following specifications: Intel (R) Core (TM) i7-6700 (3.4 GHz) processor, 32 GB of RAM, and a Windows 10/64 bit operating system.

### 5.1. Test instances and performance measure

The dataset consists of small test instances, which are sought for model validation, and large test instances for the final experiments. The small test instances are configured by $n = \{5, 8, 10\}$ jobs with the production being processed on $m = \{2, 3\}$ machines

**VNS-PSO** ($N_t$, Max_It, $N$, $w$, $c_1$, $c_2$)

**Begin:**
Initial a population of $N$ particle
Evaluate the fitness value of each particle $i$
Record initial personal best solution $pbest^*$ of each particle $i$
Record initial global best solution $gbest^*$ from $pbest^*$ of particles
***while*** (Current time T < Threshold_PSO)
Update velocity $Velocity_i^t$ of each particle $i$ by Equation (2)
Update position $Position_i^t$ of each particle $i$ by Equation (1)
Evaluate the fitness value of each particle $i$
***for*** each particle $i$ ***do***
***if*** (current solution < $pbest^*$)
Replace $pbest^*$ by the current solution
***end if***
***if*** (current solution <$gbest^*$)
Replace $gbest^*$ by the current solution
***end if***
***end for***
Evaluate its objective value $obj(X_i)$
Record particle $i$ and $obj(X_i)$
***for*** ($i = 1{:}N$)
***while*** (iteration < Max_it) ***do***
Search new solution $X_1$ by $N_k$ ;
***if*** ($obj(X_1) < obj(X_0)$)$X_0$ is *improved* ***do***
$X = X_1$
***else***
$k = k + 1$;
***if*** $k = k_{max}$ ***do***
$k = 1$;
***end if***
***end else***
Generate a new solution applying the shaking mechanism
***end if***
iteration = iteration + 1
***end while***
Output current best solution and $obj(X_i)$
***end for***
***end while***
**Output:** $gbest^*$
**End**

**Fig. 5.** Pseudocode of the VNS-PSO algorithm.

and distribution being performed using $v = \{2, 3\}$ vehicles. The large-scale instances are generated considering $n = \{20, 40, 50\}$ jobs, $m = \{5, 10, 15\}$ machines, and $v = \{5, 10, 15\}$ vehicles. The vehicle capacity is set at 20 weight units. The remainder of the operational parameters is generated randomly using a uniform distribution with the following specifications; that is, setup and processing time $U$ (10, 50), delivery time $U$ (50, 100), buffer capacity $U$ (2, 4), and order (job) weight $U$ (1, 10). Different seed values are used to generate ten distinct instances for every configuration.

The solution algorithms are compared considering the Relative Performance Difference (RPD; Eq. (15)), where a smaller RPD indicates that the associated solution is of better quality.

$$RPD = \frac{Fitness(\pi') - Fitness(\pi)}{Fitness(\pi)} \times 100 \tag{24}$$

### 5.2. Results analysis

The performance of VNS-CS and VNS-PSO algorithms is significantly influenced by the computational parameters. For this purpose, random instances with the size of the largest problem are considered to determine the best value for the algorithm parameters. In the VNS-CS algorithm, the step size ($s$) is calibrated manually. Considering $s \in \{0.01, 0.1, 1.0\}$, $N_t \in \{0.2, 0.4, 0.6\}$, and

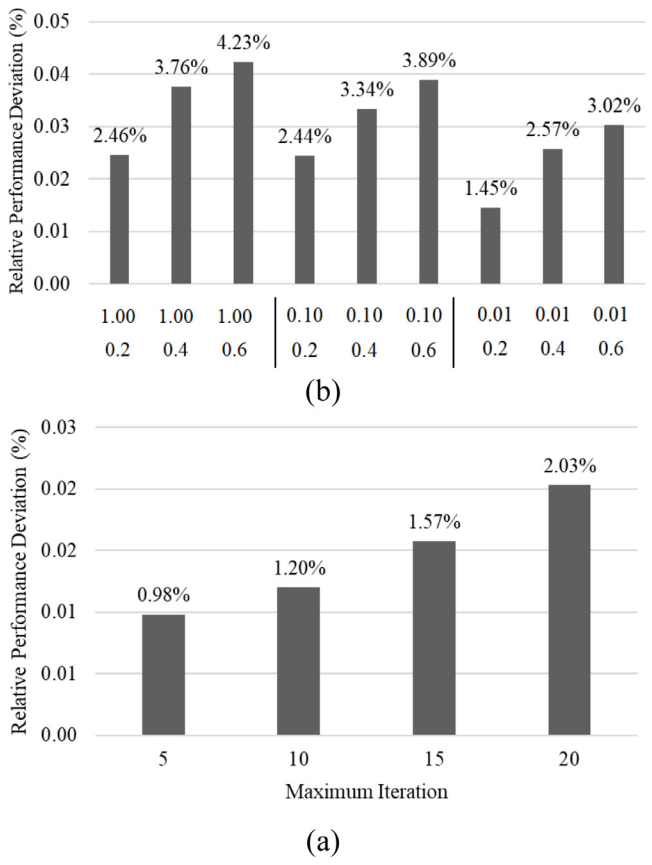


**Fig. 6.** Calibration test results of the VNS-CS algorithm considering (a) maximum iterations, (b) step size, and disturbance ratio.

$Max\_It \in \{5, 10, 15, 20\}$, the calibration results are summarized in Fig. 6.

Given the dimensionality of the VNS-PSO algorithm parameters, a dynamic parameter setting approach is used to find the optimum value of the disturbance ratio ($N_t$), the maximum iteration for the neighborhood search ($Max\_It$), inertia weight ($\omega$), the cognitive ($c_1$) and social coefficients ($c_2$). The pseudocode of the calibration algorithm is provided in Fig. 7.

On this basis, the final experiments will be conducted considering the following parameter values. (1) VNS-CS: $s = 0.01$, $N_t = 0.02$, and $Max\_It = 5$. (2) VNS-PSO: $\omega = 0.9$, $c_1 = 0.1$, $c_2 = 0.1$, $N_t = 0.2$, $Max\_It = 20$. Besides, the performance of every algorithm is benchmarked considering three encoding methods, i.e., real, cycle, and float, throughout the experiments. Finally, a CPU time of 300 s is considered to compare the performance of the algorithms fairly.

The small test instances are first solved using an exact method, MILP, to validate the formulation and compare the results with those of the VNS-CS and VNS-PSO algorithms. Table 2 summarizes the resulting fitness values and computational time. It is observed that there is no meaningful difference in the computational time of the metaheuristics, which are both outperforming the exact optimizer in terms of computational efficiency. Notably, the Gurobi optimizer could not find an exact solution to the problem of 10 jobs/3 machines/3 cars within 3600 s; the near-optimum value returned by the optimizer for experienced a 13.7 percent error. On the other hand, the solutions obtained by the VNS-CS algorithm present an equal fitness value but the results of the VNS-PSO algorithm are slightly larger while minimizing the fitness value is desirable.

Table 3 summarizes the best-found solutions to the large-size instances. The vast majority of the best solutions in the first two configurations, (20, 5, 5) and (40, 10, 10), are yielded by the VNS-CS algorithms while all the best solutions in the largest problem, (50, 15, 15) are found by the VNS-PSO algorithms. Notably, the real number encoding is best suited for solving the PRP-TWST problem. Considering the overall outcomes, however, VSN-CSC shows a better performance.

Figs. 8–10 visually compare the performance of the algorithm considering the RPD value. The first observation is that the VNS-CS algorithm performs better when real and cycle encoding approaches are applied while the float encoding approach seems to result in better outcomes in the VNS-PSO when the problem of 20 jobs and 5 machines/vehicles is considered. It is also observed that the difference between the performance of the VNS-CS and VNS-PSO becomes smaller with an increase in the problem size with VNS-CS performing slightly better in the problem of 40 jobs and 10 machines/vehicles. In the largest instance, however, VNS-PSO with real encoding outperforms with a 0.83 percent margin.

## 6. Conclusions

Production and distribution decisions at the operational level of supply chain management are interrelated and should be planned simultaneously to ensure the feasibility of the solutions and the effectiveness of the optimization approach. This study investigated a relatively new extension to the integrated production scheduling and vehicle routing problems. Given the impact of the setup operations and time window on the responsiveness of the operations, a mathematical formulation is proposed to account for these operational aspects in PRP, considering total delay time as the optimization objective. Two improved metaheuristics were developed to solve this highly dimensional optimization problem. Comparing these metaheuristics, the experiments showed that they can obtain the optimum solution to the small-scale problems in a slight fraction of the time required by the exact optimizer. Besides, the numerical results suggest that the VNS-CS algorithm is relatively more effective for solving large-size instances.

This study is limited in that it assumes a static and deterministic operational environment. Addressing these shortcomings will allow for a deeper analysis of the system behavior in particular circumstances. For example, one can include the possibility of rejecting or partially accepting an order taking into account the status of both production and distribution resources. As a second suggestion, the pickup and delivery considering dynamic capacity adjustment can be featured in a way that forward and reverse operations can be planned simultaneously. Third, the present study only considered outbound logistics operations. The next suggestion, therefore, comes from integrating inbound logistics decisions into PRP-TWST to further improve the connectedness of the supply chains, i.e., by considering the availability of supplies. Fourth, other variants of routing problems, like the General Routing Problems and Arc Routing Problems are not integrated with the production scheduling problems; disassembly operations could be coupled with reverse logistics operations using an integrated general routing-based disassembly line-balancing problem. As the final suggestion, other state-of-the-art optimization algorithms, like the chaos-enhanced simulated annealing, Meta-Lamarckian-based iterated greedy, and the hyperbolic gray wolf optimizer should be developed for the benchmark against our proposed methods. For this purpose, introducing a standard testbed for PRPs helps the development of this understudied optimization problem. On this basis, more constructive heuristics and metaheuristic algorithms should be developed to provide better solutions to the PRPs.

```
Calibration (parameters)
Begin:
    Initialization: global_best, particle_best, curr_particle_obj, curr_global_obj
    while (Current time T < stop criteria of PSO)
      VNS-PSO ();
      for (particle I = 1~N)
        if (curr_particle_obj< particle_best)
          w = w − 0.01;
          c1 = c1 + 0.03;
        end if
        else
          particle_best = curr_particle_obj;
        end else
      if (curr_global_obj< global_best)
        w = w − 0.01;
        c2 = c2 + 0.03;
      end if
      else
        global_best = curr_global_obj;
      end else
    end while
Output: optimum parameter values
End
```

**Fig. 7.** Pseudocode of the calibration algorithm.

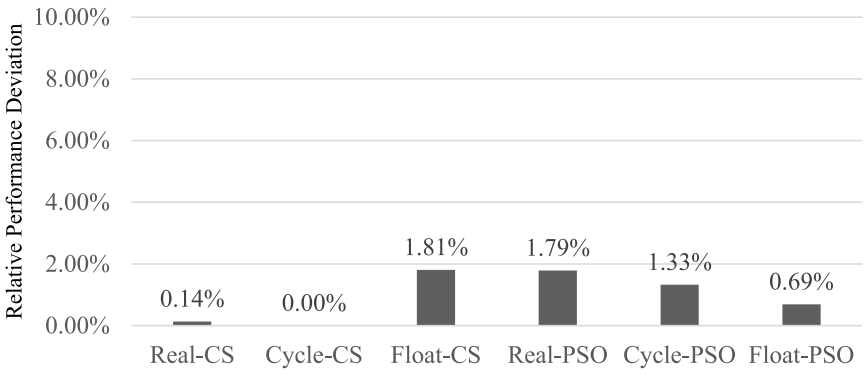


**Fig. 8.** Benchmark results for the problem of 20 jobs, 5 production machines, and 5 delivery vehicles.

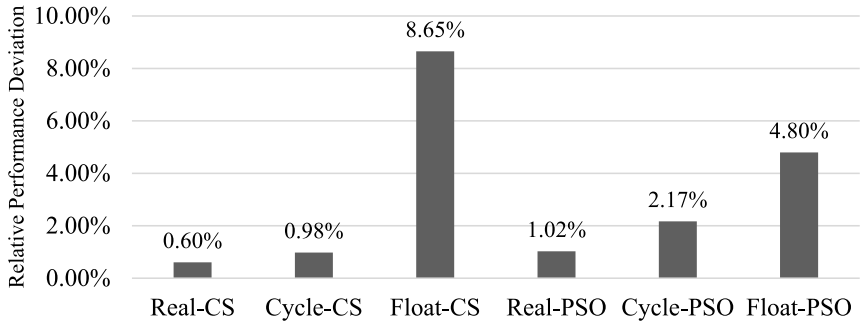


**Fig. 9.** Benchmark results for the problem of 40 jobs, 10 production machines, and 10 delivery vehicles.

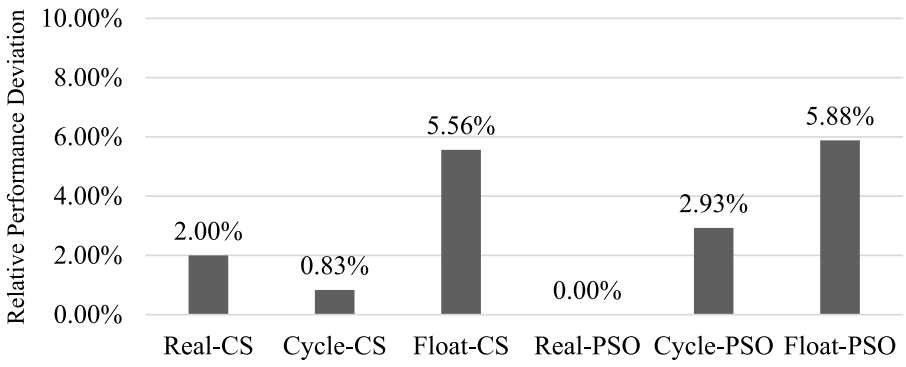


**Fig. 10.** Benchmark results for the problem of 50 jobs, 15 production machines, and 15 delivery vehicles.

**Table 2**
Solutions to the small-size instances obtained by the exact method and the metaheuristics (best in **bold**).

| $(m, n, v)$ _seed | MILP | VNS-CS | VNS-PSO |
|---|---|---|---|
| (5, 2, 2) _10 | **2448.50** | **2448.50** | **2448.50** |
| (5, 2, 2) _20 | **3336.20** | **3336.20** | **3336.20** |
| (5, 2, 2) _30 | **2053.63** | **2053.63** | **2053.63** |
| Average CPU time (s) | 0.437 | 0.014 | **0.008** |
| (8, 2, 2) _10 | **4876.30** | **4876.30** | **4876.30** |
| (8, 2, 2) _20 | **4725.48** | **4725.48** | **4725.48** |
| (8, 2, 2) _30 | **6571.26** | **6571.26** | **6571.26** |
| Average CPU time (s) | 191.420 | **0.015** | 0.017 |
| (10, 3, 3) _1 | **5303.18** | **5303.18** | 5723.98 |
| (10, 3, 3) _2 | **5141.90** | **5141.90** | 5662.81 |
| (10, 3, 3) _3 | **4745.82** | **4745.82** | 5841.33 |
| Average CPU time (s) | >3600 | 0.051 | **0.046** |

**Table 3**
Best-found solutions to the large-scale instances (best in **bold**).

| Configuration $(n, m, v)$ | Instance | VNS-CS-R[1] | VNS-CS-C[2] | VNS-CS-F[3] | VNS-PSO-R | VNS-PSO-C | VNS-PSO-F |
|---|---|---|---|---|---|---|---|
| | 1 | 12170.75 | **12122.43** | 12432.59 | 12303.47 | 12383.55 | 12385.4 |
| | 2 | 12057.82 | **12024.35** | 12057.82 | 12288.48 | 12057.82 | 12063.1 |
| (20, 5, 5) | 3 | **11887.82** | **11887.82** | 12284.08 | 11973.8 | **11887.82** | **11887.82** |
| | 4 | **9470.76** | **9470.76** | 9742.4 | 9577.64 | 9642.91 | 9562.27 |
| | 5 | **11949.03** | **11949.03** | **11949.03** | 12355.27 | 12234.1 | **11949.03** |
| | 1 | **23616.37** | **23641.81** | 25295.01 | 23854.72 | 23857.85 | 24143.41 |
| | 2 | 22871.96 | 22631.42 | 23865.73 | 22642.04 | **22202.21** | 23586.62 |
| (40, 10, 10) | 3 | **20307.24** | 20587.45 | 22971.63 | 20483.15 | 20329.2 | 21235.16 |
| | 4 | **21987.13** | 22255.3 | 24010.11 | 22258.84 | 23119.12 | 23204.12 |
| | 5 | **24986.39** | 25048.55 | 26569.59 | 24993.82 | 26123.52 | 26337.88 |
| | 1 | 24984.91 | 24492.53 | 25695.12 | **24382.24** | 24598.96 | 24812.68 |
| | 2 | 23532.23 | 23108.15 | 23900.95 | **23072.91** | 24572.02 | 25595.1 |
| (50, 15, 15) | 3 | 23196.75 | 22851.72 | 23945.67 | **22423.77** | 22533.04 | 23465.42 |
| | 4 | 22478.23 | 22249.74 | 23774.28 | **22201.71** | 23565.13 | 23844.94 |
| | 5 | 23944.93 | 24088.43 | 24928.13 | **23745.82** | 23895.49 | 24854.17 |
| Average | | 19296.15 | **19227.30** | 20228.14 | 19237.18 | 19533.52 | 19928.47 |

R: real, C: cycle, F: float.

## CRediT authorship contribution statement

**Gen-Han Wu:** Conceptualization, Methodology, Validation, Formal analysis. **Chen-Yang Cheng:** Software, Resources, Project administration. **Pourya Pourhejazy:** Writing – original draft, Writing – review & editing, Formal analysis, Investigation. **Bai-Lyn Fang:** Formal analysis, Data curation.

## 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.

## Data availability

Data will be made available on request.

## References

[1] A. Dolgui, D. Ivanov, B. Sokolov, Ripple effect in the supply chain: An analysis and recent literature, Int. J. Prod. Res. 56 (2018) 414–430.
[2] H.N. Geismar, G. Laporte, L. Lei, C. Sriskandarajah, The integrated production and transportation scheduling problem for a product with a short lifespan, INFORMS J. Comput. 20 (2008) 21–33, http://dx.doi.org/10.1287/ijoc.1060.0208.
[3] A. Khanuja, R.K. Jain, Supply chain integration: A review of enablers, dimensions and performance, Benchmarking Int. J. (2019).
[4] S. Tiwari, Supply chain integration and industry 4.0: A systematic literature review, Benchmarking Int. J. 28 (2020) 990–1030, http://dx.doi.org/10.1108/BIJ-08-2020-0428.

[5] C. Liu, J. Yao, Dynamic supply chain integration optimization in service mass customization, Comput. Ind. Eng. 120 (2018) 42–52, http://dx.doi.org/10.1016/j.cie.2018.04.018.
[6] K. Ganesh, A.S. Nallathambi, T.T. Narendran, Variants, Solution approaches and applications for vehicle routing problems in supply chain: Agile framework and comprehensive review, Int. J. Agil. Syst. Manag. 2 (2007) 50–75.
[7] X. Zheng, M. Yin, Y. Zhang, Integrated optimization of location, inventory and routing in supply chain network design, Transp. Res. Part B Methodol. 121 (2019) 1–20.
[8] P. Pourhejazy, O.K. Kwon, The new generation of operations research methods in supply chain optimization: A review, Sustainability 8 (2016) http://dx.doi.org/10.3390/su8101033.
[9] S. Sofianopoulou, I. Mitsopoulos, A review and classification of heuristic algorithms for the inventory routing problem, Int. J. Oper. Res. 41 (2021) 282–298.
[10] P. Pourhejazy, O.K. Kwon, H. Lim, Integrating sustainability into the optimization of fuel logistics networks, KSCE J. Civ. Eng. 23 (2019) 1369–1383, http://dx.doi.org/10.1007/s12205-019-1373-7.
[11] M. Darvish, L.C. Coelho, Sequential versus integrated optimization: Production, location, inventory control, and distribution, Eur. J. Oper. Res. 268 (2018) 203–214.
[12] J. Tarifa-Fernandez, J. De Burgos-Jiménez, Supply chain integration and performance relationship: A moderating effects review, Int. J. Logist. Manag. (2017).
[13] C.A. Ullrich, Integrated machine scheduling and vehicle routing with time windows, European J. Oper. Res. 227 (2013) 152–165, http://dx.doi.org/10.1016/j.ejor.2012.11.049.
[14] S. Moons, K. Ramaekers, A. Caris, Y. Arda, Integrating production scheduling and vehicle routing decisions at the operational decision level: A review and discussion, Comput. Ind. Eng. 104 (2017) 224–245.
[15] C.-Y. Cheng, P. Pourhejazy, K.-C. Ying, C.-F. Lin, Unsupervised learning-based artificial bee colony for minimizing non-value-adding operations, Appl. Soft Comput. 105 (2021) 107280.
[16] P. Farahani, M. Grunow, H.-O. Günther, Integrated production and distribution planning for perishable food products, Flex. Serv. Manuf. J. 24 (2012) 28–51, http://dx.doi.org/10.1007/s10696-011-9125-0.
[17] P. Amorim, M.A.F. Belo-Filho, F.M.B. Toledo, C. Almeder, B. Almada-Lobo, Lot sizing versus batching in the production and distribution planning of perishable goods, Int. J. Prod. Econ. 146 (2013) 208–218, http://dx.doi.org/10.1016/j.ijpe.2013.07.001.
[18] C. Meinecke, B. Scholz-Reiter, A heuristic for the integrated production and distribution scheduling problem, Int. Sci. Index. 8 (2014) 290–297.
[19] R.F. Tavares-Neto, M.S. Nagano, An iterated greedy approach to integrate production by multiple parallel machines and distribution by a single capacitated vehicle, Swarm Evol. Comput. 44 (2019) 612–621, http://dx.doi.org/10.1016/j.swevo.2018.08.001.
[20] I. Dayarian, G. Desaulniers, A branch-price-and-cut algorithm for a production-routing problem with short-life-span products, Transp. Sci. (2019) http://dx.doi.org/10.1287/trsc.2018.0854, trsc.2018.0854.
[21] L.-L. Fu, M.A. Aloulou, C. Triki, Integrated production scheduling and vehicle routing problem with job splitting and delivery time windows, Int. J. Prod. Res. 55 (2017) 5942–5957.
[22] Y. Adulyasak, J.-F. Cordeau, R. Jans, The production routing problem: A review of formulations and solution algorithms, Comput. Oper. Res. 55 (2015) 141–152, http://dx.doi.org/10.1016/j.cor.2014.01.011.
[23] X. Zou, L. Liu, K. Li, W. Li, A coordinated algorithm for integrated production scheduling and vehicle routing problem, Int. J. Prod. Res. 56 (2018) 5005–5024.
[24] F. Marandi, S.M.T. Fatemi Ghomi, Integrated multi-factory production and distribution scheduling applying vehicle routing approach, Int. J. Prod. Res. 57 (2019) 722–748.
[25] T. Nishi, Y. Hiranaka, I.E. Grossmann, A bilevel decomposition algorithm for simultaneous production scheduling and conflict-free routing for automated guided vehicles, Comput. Oper. Res. 38 (2011) 876–888.
[26] H. Liu, Z. Guo, Z. Zhang, A hybrid multi-level optimisation framework for integrated production scheduling and vehicle routing with flexible departure time, Int. J. Prod. Res. (2020) 1–18.
[27] J. Long, P.M. Pardalos, C. Li, Level-based multi-objective particle swarm optimizer for integrated production scheduling and vehicle routing decision with inventory holding, delivery, and tardiness costs, Int. J. Prod. Res. (2021) 1–20.
[28] M. Ganji, H. Kazemipoor, S.M.H. Molana, S.M. Sajadi, A green multi-objective integrated scheduling of production and distribution with heterogeneous fleet vehicle routing and time windows, J. Clean. Prod. 259 (2020) 120824.
[29] F. Jafari Nozar, J. Behnamian, Hyper-heuristic for integrated due-window scheduling and vehicle routing problem for perishable products considering production quality, Eng. Optim. (2020) 1–20.
[30] F. Neves-Moreira, B. Almada-Lobo, J.-F. Cordeau, L. Guimarães, R. Jans, Solving a large multi-product production-routing problem with delivery time windows, Omega 86 (2019) 154–172, http://dx.doi.org/10.1016/j.omega.2018.07.006.
[31] D. Biskup, J. Herrmann, J.N.D. Gupta, Scheduling identical parallel machines to minimize total tardiness, Int. J. Prod. Econ. 115 (2008) 134–142, http://dx.doi.org/10.1016/j.ijpe.2008.04.011.
[32] X.-S. Yang, S. Deb, Cuckoo search via Lévy flights, in: 2009 World Congr. Nat. Biol. Inspired Comput., 2009, pp. 210–214, http://dx.doi.org/10.1109/NABIC.2009.5393690.
[33] A. Sharma, A. Sharma, V. Chowdary, A. Srivastava, P. Joshi, Cuckoo search algorithm: A review of recent variants and engineering applications, in: Metaheuristic Evol. Comput. Algorithms Appl., Springer, Singapore, 2021, pp. 177–194, http://dx.doi.org/10.1007/978-981-15-7571-6_8.
[34] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proc. ICNN'95 - Int. Conf. Neural Networks, Vol. 4, 1995, pp. 1942–1948, http://dx.doi.org/10.1109/ICNN.1995.488968.
[35] A. Mittal, A. Pattnaik, A. Tomar, Different variants of particle swarm optimization algorithms and its application: A review, in: Metaheuristic Evol. Comput. Algorithms Appl., Springer, Singapore, 2021, pp. 131–163, http://dx.doi.org/10.1007/978-981-15-7571-6_6.
[36] N. Mladenović, P. Hansen, Variable neighborhood search, Comput. Oper. Res. 24 (1997) 1097–1100, http://dx.doi.org/10.1016/S0305-0548(97)00031-2.
[37] P. Hansen, N. Mladenović, J. Brimberg, J.A.M. Pérez, Variable neighborhood search, in: Handb. Metaheuristics, 2019, pp. 57–97, http://dx.doi.org/10.1007/978-3-319-91086-4_3.
[38] Y. Qiu, L. Wang, X. Xu, X. Fang, P.M. Pardalos, A variable neighborhood search heuristic algorithm for production routing problems, Appl. Soft Comput. 66 (2018) 311–318, http://dx.doi.org/10.1016/j.asoc.2018.02.032.
[39] J. Huang, L. Gao, X. Li, An effective teaching-learning-based cuckoo search algorithm for parameter optimization problems in structure designing and machining processes, Appl. Soft Comput. 36 (2015) 349–356, http://dx.doi.org/10.1016/j.asoc.2015.07.031.
[40] S. Cheng, H. Zhan, Z. Shu, An innovative hybrid multi-objective particle swarm optimization with or without constraints handling, Appl. Soft Comput. 47 (2016) 370–388, http://dx.doi.org/10.1016/j.asoc.2016.06.012.