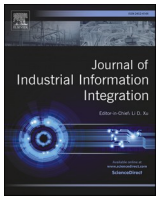
Journal of Industrial Information Integration 38 (2024) 100574

Contents lists available at ScienceDirect

Journal of Industrial Information Integration

journal homepage: www.sciencedirect.com/journal/journal-of-industrial-information-integration



The multi-factory two-stage assembly scheduling problem

Hamed Kazemi [a,b,*], Mustapha Nourelfath [a,b], Michel Gendreau [b,c]

[a] Department of Mechanical Engineering, Laval University, Quebec, Canada
[b] Interuniversity Research Center on Entreprise Networks, Logistics and Transportation (CIRRELT), Canada
[c] Department of Mathematical and Industrial Engineering, Polytechnique Montréal, Montréal, Canada

ARTICLE INFO

Keywords:
Multi-factory scheduling
Mixed-integer programming
Branch and bound method
Heuristic algorithms

ABSTRACT

The recent notable focus on distributed production management in academic and industrial contexts has underscored the importance of scheduling across multiple factories. Accordingly, this study investigates a new multi-factory configuration in which non-identical factories produce different components of a final product in the first stage. Each factory is considered as a classical flow-shop, which can manufacture a unique component. These components are assembled into final products in the assembly factory, which is located in the second stage. Unlike other distributed scheduling problems, to determine a united production sequence in such a system, there is no need to find a suitable factory to assign a job since each factory is qualified for a particular task. In real-world applications, these systems encounter challenges that span from information architectures and negotiation mechanisms to the development of scheduling algorithms. The objective of this research is to schedule the jobs in each factory to minimize the makespan of the entire process. For this purpose, a mixed-integer programming model is developed to deal with small-size instances. Then, the lower bound is derived and incorporated to develop a branch and bound method. Furthermore, to deal with larger instances, five heuristic methods are developed, and the worst-case analysis is carried out. Computational experiments are conducted for different test classes to compare and to highlight the performance of the proposed solution procedures.

1. Introduction

The globalization trend in the current century has led the classical production units to form a production network. As market requirements, technological trends, and labor costs vary throughout the regions, geographically decentralized factories are more flexible to fit these variations and provide competitive advantages [1–3]. Many industries, such as aerospace, aircraft, and electronics, produce their products in such distributed environments [4]. The critical fact in distributed manufacturing environments is integration. By aggregating all the possible resources, such production networks can produce complex products while companies are still capable of reconfiguring themselves to catch new markets [5,6].

Unlike traditional single-factory scheduling problems, In the multi-factory production network, the extra task is to assign the jobs to potentially suitable factories [3]. This research, however, investigates a new kind of collaborative manufacturing in which each job should be done via a specified factory. In the investigated problem, the different components of the final products are manufactured via different factories in the first stage. Each factory is considered an independent unit, which is qualified to produce a unique component. These components are then assembled into a final product in an assembly factory, which is located in the next stage. Although the factories are considered as independent entities, they are integrated for the common objectives in the network. In order to provide a united schedule in such a network, there is no need to make decisions about assigning the jobs to potentially suitable factories. This difference primarily distinguishes this study from the distributed assembly scheduling problem introduced by [7] and other distributed scheduling problems.

The introduced problem is a natural extension of the two-stage assembly scheduling problem addressed by [8,9]. Laptop manufacturing is a good example of this configuration of factories since components like the CPU and graphics card are manufactured separately in different factories. These components, along with the surface and keyboard, are assembled at the destination factory. In addition, such a distributed configuration of the factories could be observed in the wood industry in Canada, which contributed a total of $19.8 billion to the country's GDP in 2013 [10].

The relationship among the factories in collaborative manufacturing networks has been analyzed through different views. Johansen et al.

* Corresponding author at: Department of Mechanical Engineering, Laval University, Quebec, Canada.
E-mail address: hamed.kazemi.1@ulaval.ca (H. Kazemi).

https://doi.org/10.1016/j.jii.2024.100574

Available online 12 January 2024

[11] explained vertical structures against horizontal structures. In a vertical structure, the factory may be connected to the suppliers (upstream) or the customers (downstream). In the horizontal structure, there are some competitor factories producing similar products. In supply chain scheduling, however, the horizontal or parallel structure means that the objective is to assign the orders to the suitable factories to respond to the customer's need in minimum time or by minimum cost. Lohmer and Lasch [3] provided a comprehensive review of multi-factory network scheduling and classified the literature with respect to the location of the factories within the network, machine arrangement in the factories, objective functions, and solution procedures.

According to the literature, the collaboration between component manufacturing factories and the assembly factory could be considered as a vertical structure. The configuration of the factories in this research is entitled "The multi-factory two-stage assembly scheduling problem". In the next section, the literature of multi-factory supply chain scheduling will be provided. Classified with respect to the location of the factories, this literature will highlight the contribution of this research.

2. Literature review

Factories could be located in different positions in a supply chain. If the factories are placed in series, the jobs are processed respectively in each factory. In this case, the products of a factory will be considered as the raw material for the next factory. On the other hand, factories may be in parallel structures in which there are multiple factories that can produce various types of products. The other cases in which factories are not located exactly in series or parallel will be entitled as "hybrid position factories".

2.1. Serial factory

Serial factory scheduling problems has been addressed by [12–16]. Sawik [12] considered a three-stage supply chain where a supplier of product-specific materials will produce and deliver the orders for assembly stage and then the final products are dispatched to a set of customers. The objective is to minimize the total supply chain inventory holding cost and the production line startup and part shipment costs. H'Mida and Lopez [13] addressed the problem of integrated production and transportation scheduling while managing the resource capacity and material flows in a serial multi-site manufacturing environment. They motivated the problem by addressing an industrial case study. According to the nature of the case, the constraint satisfaction approach has been used to tackle the problem. Huang and Yao [14] considered the optimal sequencing, lot-sizing and scheduling decisions for several products which should be manufactured via several plants in a serial-type supply chain aiming to minimize the sum of setup and inventory holding costs while satisfying a given demand from customers. A three-phase heuristic is developed to solve the mentioned NP-hard problem. Karimi and Davoudpour [15] proposed a branch and bound algorithm for minimizing the tardiness and transportation costs in a serial multi-factory scheduling problem. The transportation time among the factories is considered in this research. It is also assumed that all jobs are ready at zero time in the system. Since the mentioned problem is NP-hard, it is necessary to develop meta-heuristic methods to deal with larger instances. Therefore, in another research, Karimi and Davoudpour [16] proposed a mixed-integer linear programming model and a knowledge-based imperialistic competitive algorithm to find the approximate optimum solution for a similar problem.

2.2. Parallel factory

Naderi and Ruiz [17] introduced the distributed permutation flow shop scheduling problem (DPFSP) in which there are total F identical factories or shops, each one includes m machines located in series. A group of n jobs are to be divided among the factories, and the production sequence must be determined for all the jobs assigned to each factory. The optimization criterion is that of minimizing the maximum completion time among the factories. In their later research, Naderi and Ruiz [18] improved the solution procedure for DPFSP by proposing a scatter search method. Ruiz et al. [19] developed iterated greedy procedures that perform better than the other methods in the literature of DPFSP. Hamzadayı [20] modified the mathematical model from the literature of DPFSP and proposed exact benders decomposition algorithms via the new version of the mathematical formulation. The extension of sequence-dependent setup times is added to the DPFSP by [21]. Miyata and Nagano [22] considered both maintenance activities and setup times in a distributed no-wait flow-shop environment.

Fernandez-Viagas et al. [23] addressed the DPFSP with respect to minimizing the total flow-time for the first time. Pan et al. [24] proposed constructive heuristics and meta-heuristics to minimize total flow-time in DPFSP. Zhu et al. [25] Considered due windows for job completion and aimed to minimize the sum of weighted earliness and tardiness in distributed no-wait flow shop.

The distributed hybrid flow-shop sequencing problem is investigated by [26–28] in which the jobs need to be assigned to identical factories where each factory is a hybrid flow-shop. No-wait flow-shop scheduling problem in a distributed environment is also addressed by [29,30] with respect to minimizing the makespan. Qin et al. [31] addressed the integrated production-distribution scheduling in distributed hybrid flow-shop scheduling problem aiming to minimize the sum of tardiness, earliness, and delivery costs. Ying and Lin [32] studied the blocking flow-shop scheduling problem in a distributed environment and developed iterated greedy algorithms with different characteristics to minimize the makespan. Sequence-dependent setup times have been considered in distributed blocking flow-shop scheduling problems so as to minimize the make span [33].

Zhang and Xing [34] investigated the more realistic DPFSP by considering the extension of limited buffer in each factory. They proposed differential evolution algorithms with respect to the distributed properties of the problem. Deng and Wang [35] addressed a multi objective DPFSP and proposed a competitive memetic algorithm so as to minimize the makespan and total tardiness. Another multi-objective optimization study is investigated by [36] in the distributed-reentrant permutation flow-shop sequencing environment aiming to minimize the total cost, makespan, and average tardiness simultaneously.

Behnamia and Fatemi Ghomi [37] investigated a multi-factory scheduling model where each factory has multiple identical machines in parallel. Machines among the factories may have different processing speeds hence the processing time of the same job in different factories may differ. The selected objective was that of minimizing the general makespan i.e., the maximum makespan among factories. They proposed a mathematical programming model and a genetic algorithm to solve the problem. Yazdani et al. [38] analyzed the model and algorithm proposed by [37] and showed that they suffer from serious shortcomings. To handle this, three mathematical models for makespan and total completion time objectives and also three effective metaheuristics based on artificial bee colony algorithm is proposed by [38].

Xiong et al. [39] considered a problem where there are identical factories, each one consisted a set of M dedicated machines to produce the components of a job and an assembly machine at the second stage to finalize the production process. Setup times are also considered independent of the sequence. They developed a heuristic based on SPT rule and hybrid algorithms based on VNS and GA to minimize the total completion time. Xiong and Xing [40] investigated the same problem with respect to minimizing the weighted sum of mean completion time and makespan. Deng et al. [41] also addressed the same problem with respect to makespan criterion. For this problem, a new mixed integer linear programming model is developed, which is able to achieve optimal solution in reasonable time for the instances with up to 12 jobs. For the large-scale problem, a competitive memetic algorithm is developed that outperforms the formerly VNS and GA-RVNS proposed by

Table 1
Indices, parameters and variables used in MILP model.

Index	Description
i, j	index of the components / products, $i, j = 0, 1, \ldots, n$, where 0 indicates a dummy component / product
k	index of machines at each factory, $k = 1, \ldots, m$.
f	index of factories
Parameters	Description
n	number of final products, number of components to be processed in each factory in the first stage
m	number of machines
P_{jkf}	processing time of component j on machine k in factory f
s_j	assembly time of product j
t_f	transportation time between the factory f and the assembly factory
M	a large positive number
Variables	Description
X_{ijf}	equals to 1 if component j follows component i in factory f
Y_{ij}	binary variable equals to 1 if product j follows product i in the assembly factory
C_{jkf}	completion time of component j on machine k in factory f
CC_{jf}	completion time of component j in factory f plus transportation time to the assembly factory
R_j	ready time of product j for assembly operation
CT_j	completion time of product j in assembly factory
C_{max}	makespan

[39].

Some works addressed the transportation time or cost in the multi-factory supply chain with a parallel structure. Hou et al. [42] investigated the integrated production-distribution scheduling in DPFSP where the completed orders need to be dispatched to the customers via capacitated trucks. The customers are spread in different regions and the objective is to respond on their due windows with minimum total weighted tardiness and earliness. Sun et al. [43] studied a new and practical approach for multi-factory job allocation and production-distribution scheduling problems in which inland distance-dependent transportation lead time and maritime transport limits and variations are taken into consideration. The objective is to minimize the total operating costs, i.e., cost of production, storage, distribution and tardiness penalty. Behnamian [44] discussed a distributed parallel factory scheduling problem with transshipment lead time among the factories in which each factory had different objective i.e., processing cost minimization and profit maximization. a hybrid variable neighborhood search/tabu search algorithm was proposed to solve the mentioned problem. Another research accomplished by [45] addressed the decentralized factories scheduling problem, where a set of transporters is used for shipping goods among parallel factories to minimize the production costs over all of the factories. This problem was handled by a novel mixed integer linear programming model, a meta-heuristic method and variable neighborhood search procedure.

2.3. Hybrid position factory

In the third case, the location of factories in the supply chain can be a combination of series and parallel. Chung et al. [46] investigated the multi-factory scheduling problems where each factory includes different machines able to perform different operations. Some factories might finish the final product and send it to the customers and the other factories might just carry out intermediate products and transfer them to other factories. The objective is minimizing the makespan via proper collaboration among the factories. Hatami et al. [7] introduced the distributed assembly permutation flow-shop scheduling problem (DAPFSP) which is the generalization of the problem considered by [17]. In this case, there are identical factories in the first stage where each one is considered as a classical Flow-shop. These factories are able to produce various components that are to be assembled in the second stage in the assembly factory. The objective is to assign the components to the suitable factory and to determine the sequence of the jobs in each factory to minimize the makespan. Huang et al. [47] proposed a memetic algorithm to minimize total tardiness in DAPFSP. Several other researchers have proposed various heuristics and meta-heuristics to solve the problem of distributed assembly flow-shop scheduling considering different objective functions and extensions [31,48–65].

Because of the complexity of the assembly process of the products such as automobile engines [66], some researchers considered the flexible assembly process in the literature, i.e., the assembly operation is done with multiple machines [67–69]. This extension is added to the distributed scheduling problems by [70,71]. Zhang et al. [66] considered the distributed flexible-assembly permutation flow-shop scheduling problem which belongs to Hybrid Position Factory as classified in this research. They formulated the problem by MILP model and proposed memetic algorithm in order to minimize the makespan. Yang and Xu [72] considered the possibility of batching and delivery and flexible assembly in DAPFSP and discussed some strategies for allocating the batches. They proposed heuristics, iterated greedy algorithms, and variable neighborhood descent methods to minimize the total cost of tardiness and delivery.

De Giovanni and Pezzella [73] investigated a scheduling problem in which jobs are processed by a system of multiple distributed manufacturing units corresponding to the factories in a multi-factory environment or cells in a multi-cell context. The objective is assigning each job to one manufacturing unit and determining the optimal schedule for each unit to minimize the overall completion time in the distributed job shop environment. Luo et al. [74] considered worker arrangement in the distributed job-shop and aimed to minimize the makespan, the maximum workload of machines, and the simultaneous workload of workers.

The recent research accomplished by [75] addressed the integrated production-distribution scheduling in a supply chain where some factories produce intermediate products and some other factories carry out the final product. The objective is to determine the production schedule and the route of the transport vehicle in order to minimize the total cost of tardiness and transportation.

2.4. Paper contribution

This paper investigates a new multi-factory configuration in the context of hybrid position factories. Regarding the geographical location, in the first stage, there are non-identical factories where each factory will produce a special component of a final product. When all the components are transported to the assembly factory in the second stage, a single assembly program is supposed to create the final product. To the best of the authors' knowledge, there is no existing paper dealing with this problem. The most similar work is done by [7] where the factories in the first stage are identical and each one is able to manufacture various components of a final product. Therefore, a decision should be made on finding a suitable factory to produce each component. This research differs from [7] and other distributed scheduling problems in that no decision is made about which factories to assign the orders to. The factories in the first stage are independent, non-identical and each one is qualified to manufacture a special component. Therefore, the collaboration among the factories is completely different and the existing solution procedures cannot be used to efficiently solve the formulated problem. Note that the context covered by this research is important in a practical setting. To deal with this problem, novel algorithms are developed to find the optimal or near-optimal solutions. Therefore, the contribution of this paper is two-fold. First, a mixed-integer programming model to find the optimum solution for small-size instances. Second, a branch and bound method and five constructive heuristic algorithms are proposed to deal with larger instances.

3. Problem statement and formulation

A set of n final products are to be scheduled and processed via a

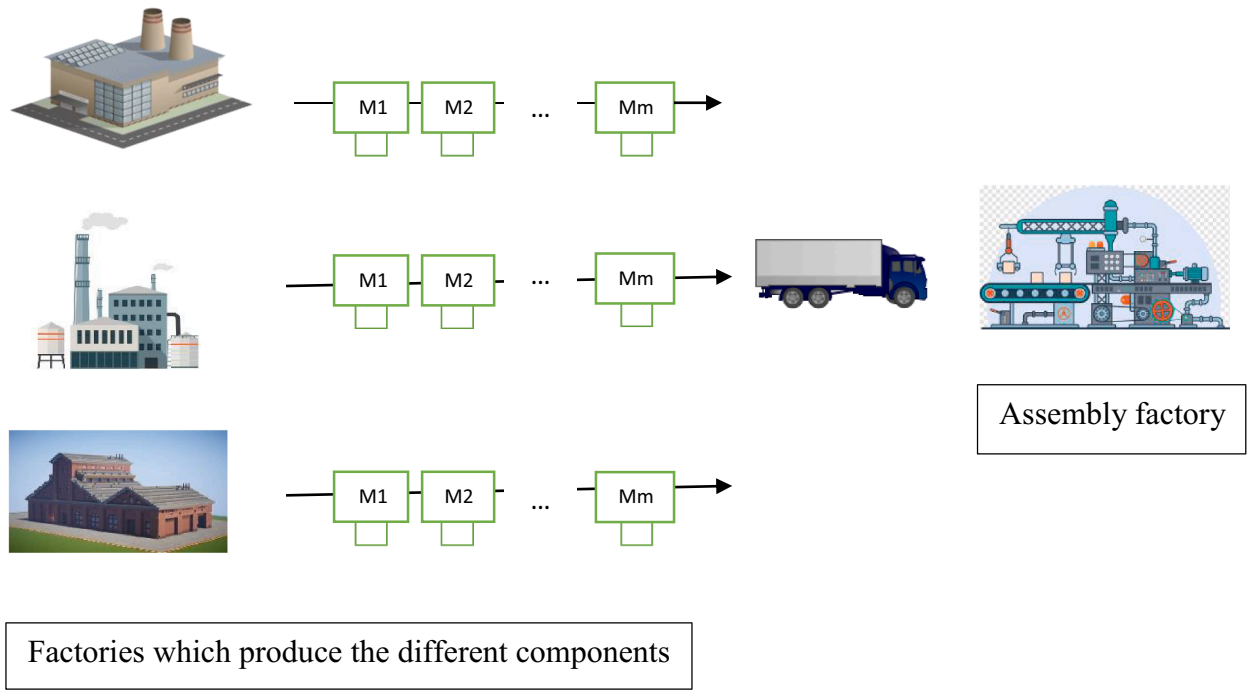


Fig. 1. Location of the factories in the multi-factory two-stage assembly framework.

network of factories. Each product consists of F different components that need to be processed through a total of F non-identical factories in the first stage. Therefore, a set of n components are to be processed in each factory in the first stage. This set is shown by $i,j \in \{1,2,\ldots,n\}$ in Table 1. All the component manufacturing factories are considered an m-machine flow-shop. The assembly operation is done in the assembly factory in the second stage. It can be started when all the components of a product have been transported to the assembly factory. At this point, a single assembly program will assemble the components to a final product. Since the common objective of the network is minimizing the makespan of the entire process, all the factories in the first stage will share their sequence process which is coordinated with the assembly factory.

The transportation times between each first-stage factory and the assembly factory is shown by t_f where f is the index of factories in the first stage. There are sufficient trucks available to transport the components to the assembly factory. Fig. 1 shows the configuration of the factories in this research.

If each factory is converted to a single machine, then the problem will be converted to the two-stage assembly scheduling problem, which is introduced by [8]. This problem is NP-hard in a strong sense with respect to minimizing the Makespan. In addition, it is proven that there exists an optimal solution within a set of permutation schedules for this problem and the complexity is $O(n!)$. The proposed MILP formulation is able to find the optimal solution whether it is a permutation or non-permutation schedule.

$$Min\ z = C_{max} \tag{1}$$

Subject to:

$$\sum_{i=0,i\neq j}^{n} X_{ijf} = 1 \qquad \forall j,f \tag{2}$$

$$\sum_{j=1,i\neq j}^{n} X_{ijf} \leq 1 \qquad \forall i,f \tag{3}$$

$$X_{ijf} + X_{jif} \leq 1 \qquad \forall j \in \{1,2,\ldots,n-1\},\ j > i, \forall f \tag{4}$$

$$C_{jkf} \geq C_{jk-1f} + P_{jkf} \quad (c_{0kf} = 0) \qquad \forall j,k,f \tag{5}$$

$$C_{jkf} \geq C_{ikf} + P_{jkf} + (X_{ijf} - 1).M \quad \forall i \neq j,k,f \tag{6}$$

$$CC_{jf} \geq C_{jkf} + t_f \qquad \forall j,f,k \tag{7}$$

$$R_j \geq CC_{jf} \qquad \forall j,f \tag{8}$$

$$CT_j \geq R_j + s_j\ (CT_0 = 0)\ \forall j \tag{9}$$

$$\sum_{i=0,i\neq j}^{n} Y_{ij} = 1 \qquad \forall j \tag{10}$$

$$\sum_{j=1,i\neq j}^{n} Y_{ij} \leq 1 \qquad \forall i \tag{11}$$

$$Y_{ij} + Y_{ji} \leq 1 \qquad \forall j \in \{1,2,\ldots,n-1\},\ j > i \tag{12}$$

$$CT_j \geq CT_i + s_j + (Y_{ij} - 1).M\ \forall i,j \tag{13}$$

$$C_{max} \geq CT_j \qquad \forall j \tag{14}$$

Eq. (1) shows the objective function that is minimizing the Makespan of the entire process. Constraint (2) ensures that each component has exactly one prior component in sequence. Constraint (3) ensures that each component has at most one following component in sequence. Constraint (4) guarantees that a component cannot be both prior and following of another component at the same time. Constraint (5) enforces component j to be processed on machine k when processing on machine $k-1$ has been completed. Constraint (6) ensures the processing of component j on machine k could not start until its prior component is completed. Constraint (7) calculates the completion time of component j in factory f plus its transportation time to the assembly factory. Constraint (8) calculates the time that product j is ready for assembly operation. Constraint (9) ensures that the assembly process of product j can not start until all its components are transported to the assembly factory. Constraint (10) ensures that each product has exactly one prior in assembly sequence. Constraint (11) ensures that each

product has at most one following product in assembly sequence. Constraint (12) guarantees that a product cannot be both prior and following of another product at the same time in assembly sequence. Constraint (13) ensures the processing of product j could not start until its prior product is completed. Finally, Constraint (14) calculates the Makespan, i.e., the time that the last product is completed.

4. Lower bound

the assembly factory for the remaining non-scheduled components. Regarding the assembly operation, it is just needed to add the assembly time of the remaining non-scheduled products.

In order to facilitate the notation, the set of all products is shown by $J = \{j_1, j_2, \ldots, j_n\}$, and $J(\sigma)$ denotes a set of products in a partial schedule (σ). The remaining non-scheduled products can be shown by $J \setminus J(\sigma)$ as a set minus operator. The lower bound of Makespan in the partial schedule (σ) can be obtained as follows:

$$LB(\sigma) = \max\left\{ CT_j(\sigma) + \sum_{J\setminus J(\sigma)} s_j + \min_{J\setminus J(\sigma)}\left\{ \max_f\left\{ CT_j(\sigma), CC_{jf}(\sigma) + \sum_{1\le k\le m} p_{jkf}\right\} - CT_j(\sigma)\right\}, \max_f\left\{ \sum_{1\le j\le n}\sum_{1\le k\le m} P_{jkf} + t_f\right\} + \min_{J\setminus J(\sigma)}(s_j)\right\}$$

As it was explained, if each factory is converted to a single machine, then the problem will be converted to the 3-machine assembly flow-shop introduced by [9]. In this research, the idea used by [9] is expanded to obtain a lower bound for the current problem. This idea is based on adding the total lengths of the remaining operations on the three machines to the partial schedule. In our research, instead of adding the total lengths of the remaining operations on a machine, we need to add the sum of processing times in each factory plus the transportation time to

In the above equation, $CT_j(\sigma)$ indicates the completion time of a final product in the assembly stage under (σ), and $CC_{jf}(\sigma)$ indicates the completion time of the final component in each factory under (σ) respectively.

In the proposed branch and bound algorithm, all the possible n nodes will be created at the first level. Then using the frontier-search heuristic, the node with the minimum lower bound will be branched. From this node, all the possible nodes will be generated. If more than one node at the same level achieves the minimum lower bound, the branching will be done from all these nodes, and the lower bounds of the resulting nodes will be compared in the next level. This procedure will be repeated until there is no unsearched node with a smaller lower bound than the incumbent node. This is a heuristic, non-exact search procedure.

Consider an example of 4 products, each one includes 3 components that are to be processed in 3 factories in the first stage. Each factory has 3 machines in series. The details of processing times in each factory and assembly times are shown in Table 2. Also, we consider $t_1 = 5$, $t_2 = 7$, $t_3 = 8$. The data is derived from a similar case study of bedroom furniture manufacturing in [76], in which the components of the

Table 2
Input data for the example.

		Product 1	Product 2	Product 3	Product 4
Factory 1	M1	14	4	10	12
	M2	21	11	18	13
	M3	4	9	13	8
Factory 2	M1	12	10	6	14
	M2	18	21	13	5
	M3	6	15	14	5
Factory 3	M1	8	6	11	13
	M2	16	10	5	9
	M3	5	3	6	7
Assembly times		50	46	35	40

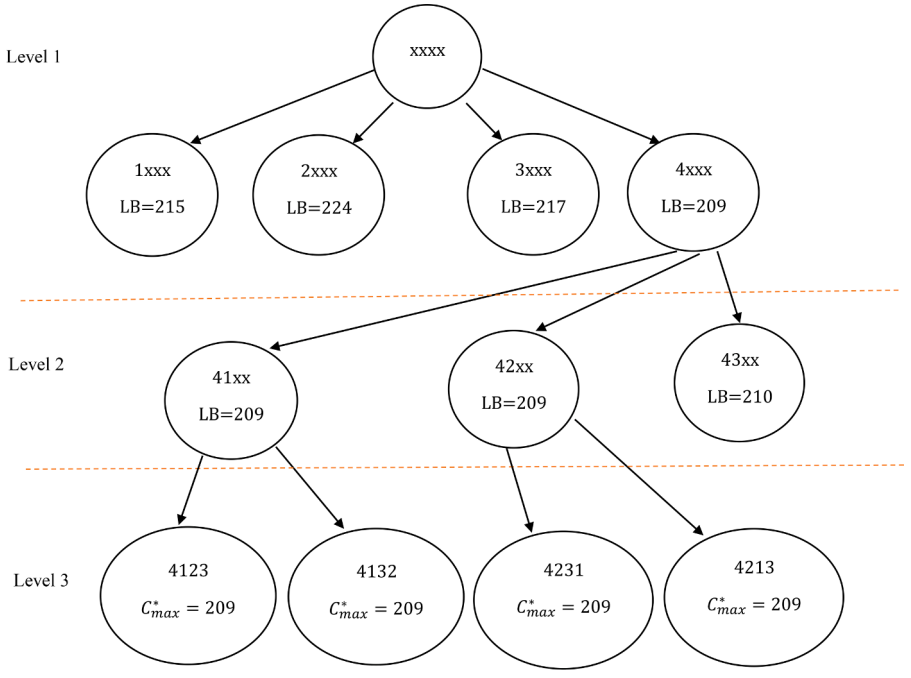


Fig. 2. The branch and bound procedure for the described example.

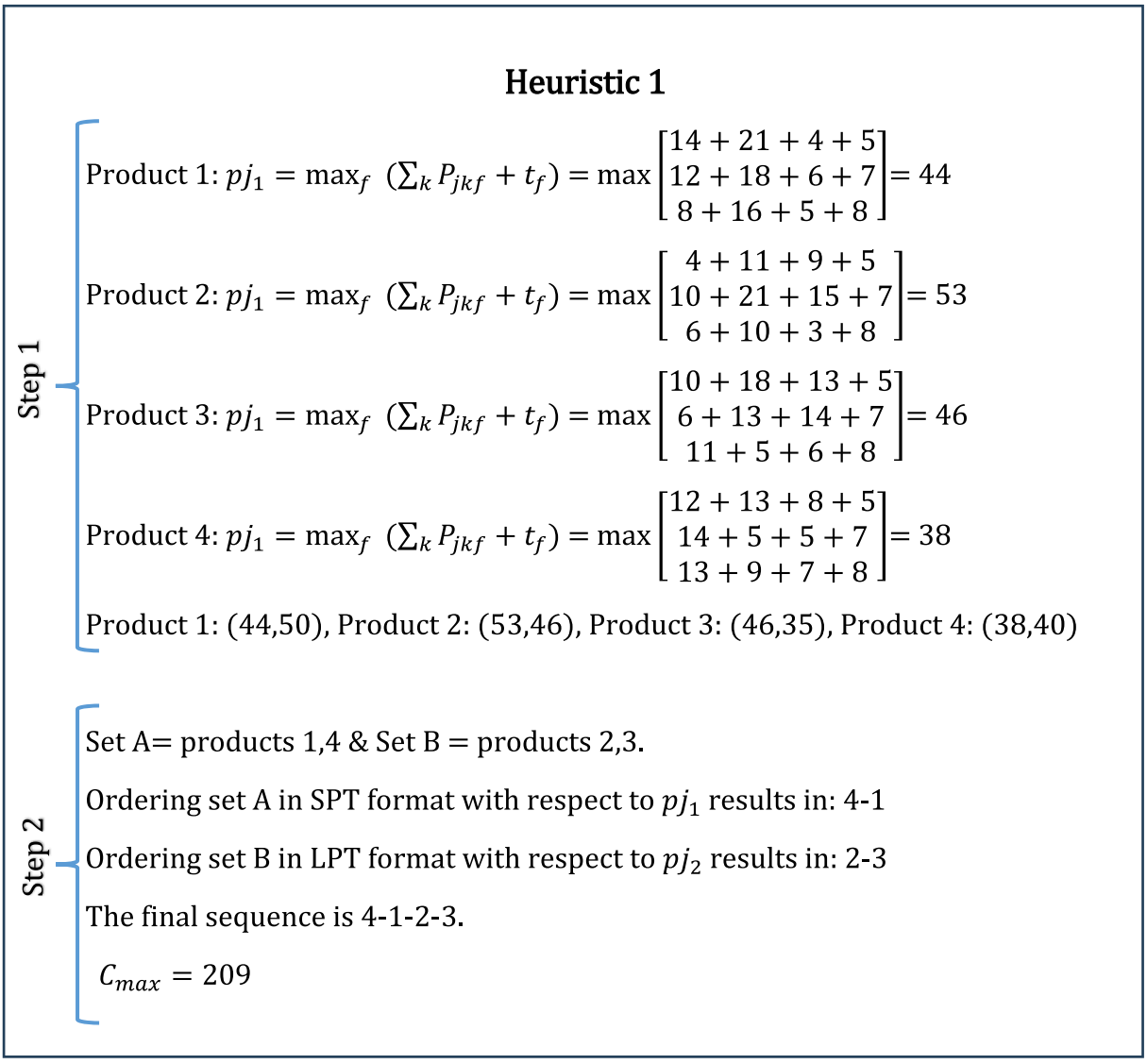


Fig. 3. The steps of Heuristic 1 to restrict the problem and reach the solution.

bedroom furniture are produced in decentralized workshops and assembled in the second stage.

As shown in Fig. 2, at the first level, the node starting with product 4 has obtained the minimum lower bound. This node will be branched to create all three possible nodes at level 2. In this level, the minimum lower bound is 209, which is obtained by the nodes 4-1 and 4-2. Branching from these nodes and creating the final sequences shows that the four solutions at level 3 obtained the same makespan, equal to 209. Since there is no unsearched node with a lower quantity, the algorithm will stop here. The optimal makespan of the problem is 209.

5. Heuristics

As mentioned before, the problem considered in this research is the expanded version of the two-stage assembly scheduling problem. The most accessible heuristics for the assembly-type scheduling problems are based on converting the problem to the classical flow-shop and then applying Johnson's algorithm to the problem. Following Johnson's rule in a two-machine flow-shop, if i and j are two successive jobs and

$$\min\{pi_1, pj_2\} \leq \min\{pj_1, pi_2\}$$

Where p shows the processing times and 1 and 2 indicate the machines, then a schedule in which job j follows job i has either smaller or equal Makespan compared to the schedule in which job i follows job j [9]. According to this, Johnson's algorithm tries to find the shortest processing time in each step. If it belongs to the first machine, the related job is put in the first position in the sequence. If the shortest processing time is related to machine 2, the related job is put in the last position in the sequence.

[9] developed three heuristics based on Johnson's algorithm by converting the fixed, 3-machine assembly flow-shop problem into the 2-machine flow-shop problem with processing times (pj_1, pj_2) as follows:

$H_1: pj_1 = \max\{pj_a, pj_b\}$ where a and b are the index of machines in the first stage

$H_2: pj_1 = pj_a$ *if* $\sum_{j=1}^{n} pj_a \geq \sum_{j=1}^{n} pj_b$; pj_b *otherwise*

$$H_3: pj_1 = \frac{pj_a + pj_b}{2}$$

For all of the mentioned heuristics, pj_2 is equal to the assembly time of the related job. This method will be generalized for the investigated problem in this research. In fact, we need to show each product by two parameters like (pj_1, pj_2). Then applying Johnson's algorithm will lead to the solution. Following this method, the proposed heuristics are described as follows:

Heuristic 2

Step 1

$$\max_f\left(\sum_j \sum_{1\le k\le m} P_{jkf} + t_f\right) =$$

$$max\begin{bmatrix}14+21+4+4+11+9+10+18+13+12+13+8+5,\\12+18+6+10+21+15+6+13+14+14+5+5+7,\\8+16+5+6+10+3+11+5+6+13+9+7+8\end{bmatrix} = 146$$

The maximum sum of processing times is occurred in factory 2 which is 146. Therefore, for each product, the sum of processing times of the component in factory 2 is considered as the first parameter.

Product 1: (43,50), Product 2: (53,46), Product 3: (40,35), Product 4: (31,40)

Step 2

Set A= products 1,4 & Set B = products 2,3.

Ordering set A in SPT format with respect to pj_1 results in: 4-1

Ordering set B in LPT format with respect to pj_2 results in: 2-3

The final sequence is 4-1-2-3.

$C_{max} = 209$

Fig. 4. The steps of Heuristic 2 to restrict the problem and reach the solution.

Heuristic 3

Step 1

Product 1: $pj_1 = \frac{\sum_f \sum_{1\le k\le m} P_{jkf}+t_f}{F} = \frac{14+21+4+5+12+18+6+7+8+16+5+8}{3} = 41.33$

Product 2: $pj_1 = \frac{\sum_f \sum_{1\le k\le m} P_{jkf}+t_f}{F} = \frac{4+11+9+5+10+21+15+7+6+10+3+8}{3} = 36.33$

Product 3: $pj_1 = \frac{\sum_f \sum_{1\le k\le m} P_{jkf}+t_f}{F} = \frac{10+18+13+5+6+13+14+7+11+5+6+8}{3} = 38.66$

Product 4: $pj_1 = \frac{\sum_f \sum_{1\le k\le m} P_{jkf}+t_f}{F} = \frac{12+13+8+5+14+5+5+7+13+9+7+8}{3} = 35.33$

Product 1: (41.33,50), Product 2: (36.33,46), Product 3: (38.66,35), Product 4: (35.33,40)

Step 2

Set A= products 1,2,4 & Set B = products 3.

Ordering set A in SPT format with respect to pj_1 results in 4-2-1.

Ordering set B in LPT format with respect to pj_2 results in placing product 3 in the last sequence.

The final sequence is 4-2-1-3.

$C_{max} = 209$

Fig. 5. The steps of Heuristic 3 to restrict the problem and reach the solution.

Heuristic 1

Step 1: Set $pj_1 = \max_f \left(\sum_k P_{jkf} + t_f\right)$ and $pj_2 = s_j \ \forall j$

Step 2: Divide the products into the two sets; set A including all products with $pj_1 < pj_2$, and set B including all products with $pj_1 > pj_2$. If $pj_1 = pj_2$, the product could be put either in A or B.

Step 3: Order set A in SPT format with respect to pj_1 and set B in LPT format with respect to pj_2.

Step 4: The ordered $A - B$ is the optimal sequence.

Heuristic 2

Step 1: Find the factory in which $\max_f \left(\sum_j \sum_{1\le k\le m} P_{jkf} + t_f\right)$ is occurred. Set $pj_1 = pj_f$ and $pj_2 = s_j \ \forall j$

The remaining steps are the same as Heuristic 1.

Heuristic 3

Step 1: Set $pj_1 = \frac{\sum_f \sum_{1\le k\le m} P_{jkf}+t_f}{F}$ and $pj_2 = s_j \ \forall j$

The remaining steps are the same as Heuristic 1.

For better understanding of the structure of the suggested heuristics, we apply them to the instance detailed in Section 4. The functionality of Heuristic 1, Heuristic 2, and Heuristic 3 in addressing this instance is

Table 3
The configuration of the small size instances.

Sets	n	f	m
Set 1	4	2	3
Set 2	4	3	3
Set 3	6	2	3
Set 4	6	3	3
Set 5	8	2	3
Set 6	8	3	3
Set 7	10	2	3
Set 8	10	3	4
Set 9	10	4	4
Set 10	12	2	3
Set 11	12	3	4
Set 12	12	4	4

depicted in Figs. 3–5, respectively. In the mentioned figures, the first step shows how each heuristic restricts each product to show it by two parameters like (pj_1, pj_2). In the second step, Johnson's algorithm is applied to find the best sequence of the products.

6. Worst case analysis

In order to simplify the notations, some new variables are introduced as follows:

$$TT_f = \sum_j \sum_k p_{jkf} + t_f \qquad (\forall f)$$

$$T_{max} = \max_f TT_f$$

$$T_2 = \sum_j s_j$$

C_{H_i} = Makespan of Heuristic H_i $(i = 1,2,3)$
C^* = Optimum Makespan

Theorem 1. *If Heuristic H_i produce a permutation schedule for the multi-factory two-stage assembly scheduling problem, then we have:*

$$C_{H_i} \le 2C^* \tag{15}$$

Proof: Most of the time, both the component manufacturing stage, and assembly stage are working, therefore we have:

$$C_{H_i} \le T_{max} + T_2 \tag{16}$$

Since $T_{max} \le C^*$ and $T_2 \le C^*$, by replacing in (16) it can be resulted that $C_{H_i} \le 2C^*$.

Theorem 2. $C_{H_3} \le 2C^* - \frac{C^*}{F}$

Proof: We know that Heuristic 3 calculates the average sum of processing time of the components among the factories as the first parameter for each product. Consider the schedule $S = \{j_1, j_2, \ldots, j_n\}$ obtained after applying Heuristic 3 for the original problem. Now if for each product we delay the processing time of each component by $\frac{\max_f (\sum_k P_{jkf} + t_f) - \min_f (\sum_k P_{jkf} + t_f)}{F}$, it will result in a new instance S' in which the upper bound of the makespan can be calculated as follow:

$$C_{H_3} + \frac{\sum_j \max_f \left(\sum_k P_{jkf} + t_f\right) - \sum_j \min_f \left(\sum_k P_{jkf} + t_f\right)}{F} \tag{17}$$

It should be noted that for a set of n positive number ($n \ge 2$), the following equation is always confirmed:

$$average + \frac{max - min}{n} \le max \tag{18}$$

Looking at the Eq. (18) and replacing in (17), we can immediately result the following inequality:

$$C_{H_3} + \frac{\sum_j \max_f \left(\sum_k P_{jkf} + t_f\right) - \sum_j \min_f \left(\sum_k P_{jkf} + t_f\right)}{F} \le C_{H_1} \tag{19}$$

Noting the fact that the contribution of $\frac{\sum_j \max_f (\sum_k P_{jkf} + t_f) - \sum_j \min_f (\sum_k P_{jkf} + t_f)}{F}$ is less than $\frac{C^*}{F}$ when it is squeezed down to the original problem, we result that $C_{H_3} + \frac{C^*}{F} \le C_{H_1}$. From the Theorem 1 we know that $C_{H_1} \le 2C^*$, therefore by replacing we have: $C_{H_3} \le 2C^* - \frac{C^*}{F}$.

7. Improved heuristics

The major difference of the assembly type flow-shop compared to the classical flow-shop is that in a partial schedule, adding a job to the current sequence will add different amounts to the completion times on the machines in the first stage. Therefore, it is better to calculate and record the sum of processing times on the machines in the first stage at each iteration of the algorithm. [77] used this idea to develop powerful heuristics based on Johnson's algorithm for the fixed, 3-machine, assembly-type flow-shop scheduling problem.

In this work we will consider how to adopt and apply this idea for the multi-factory two stage assembly scheduling problem. In the new version of the heuristics which will be described in this section, the sum of processing times in each factory at each iteration (i) will be updated according to the previous iterations. In general, at each iteration, a product will be assigned to the first part or last part of the sequence. In this notation, j denotes a product that is assigned to the first part of the sequence in the previous iteration.

Heuristic 4

Step 1: Initialize $i = 1$, $J = \{j_1, j_2, \ldots, j_n\}$, $CC^0_{jf} = 0$;

Step 2: For each product calculate the completion time of the components among all the factories plus transportation time to the assembly

Table 4
The results of the small-size problems.

Sets	H1 Gap%	H2 Gap%	H3 Gap%	H4 Gap%	H5 Gap%	MILP Time (*sec*)	Gap%	B&B Gap%
1	0	0	0	0	0	7	0	0
2	0	0	0	0	0	9	0	0
3	0	0	0	0	0	58	0	0
4	0	0	0	0	0	37	0	0
5	0	0	0	0	0	580	0	0
6	0	0	0	0	0	947	0	0
7	0	0	0	0	0	2047	0	0
8	0	0	0	0	0	2439	0	0
9	0	0.3	0	0	0.5	3600	0	0
10	0	0	0	0	0	2894	0	0
11	0.6	0.6	0.6	0.3	0.6	3600	0	0
12	0	0.91	0	0	0	3600	0	0
Average	0.050	0.151	0.050	0.025	0.092		0	0

Table 5
The gap percentage for class *A*.

$n,f,m=3$	H1	H2	H3	H4	H5	B&B
15,2	0	0	0	0	0	0
15,3	0	0	0	0	0	0
20,2	0	0	0	0	0	0
20,3	0	0	0	0	0	0
25,2	0	0	0	0	0	0
25,3	0	0	0	0	0	0
25,4	0	0	0	0	0	0
30,2	0	0	0	0	0	0
30,3	0	0	0	0	0	0
30,4	0	0	0	0	0	0
35,2	0	0	0	0	0	0
35,3	0	0	0	0	0	0
35,4	0	0	0	0	0	0
40,2	0	0	0	0	0	0
40,3	0	0	0	0	0	0
40,4	0	0	0	0	0	0
40,5	0	0	0	0	0	0
45,2	0	0	0	0	0	0
45,3	0	0	0	0	0	0
45,4	0	0	0	0	0	0
45,5	0	0	0	0	0	0
50,2	0	0	0	0	0	0
50,3	0	0	0	0	0	0
50,4	0	0	0	0	0	0
50,5	0	0	0	0	0	0
60,3	0	0	0	0	0	0
60,4	0	0	0	0	0	0
60,5	0	0	0	0	0	0
70,3	0	0	0	0	0	0
70,4	0	0	0	0	0	0
70,5	0	0	0	0	0	0
80,3	0	0	0	0	0	0
80,4	0	0	0	0	0	0
80,5	0	0	0	0	0	0
90,3	0	0	0	0	0	0
90,4	0	0	0	0	0	0
90,5	0	0	0	0	0	0
100,3	0	0	0	0	0	0
100,4	0	0	0	0	0	0
100,5	0	0	0	0	0	0
Average						

Table 6
The gap percentage for class *B*.

$n,f,m=3$	H1	H2	H3	H4	H5	B&B
15,2	0	0	0	0	0	0
15,3	0	0.47	0	0	0.35	0
20,2	0	0.11	0	0	0.11	0
20,3	0	0.7	0	0	0.41	0
25,2	0	0	0	0	0	0
25,3	0	0.66	0.09	0	0.00	0
25,4	0	0.21	0.5	0	0.21	0
30,2	0	0	0	0	0	0
30,3	0	0.64	0	0	0.11	0
30,4	0	0	0	0	0	0
35,2	0	0	0	0	0.1	0
35,3	0	1.14	0	0	0.79	0
35,4	0	0.37	0	0	0.15	0
40,2	0	0	0.04	0	0	0
40,3	0	0.71	0	0	0	0
40,4	0	0	0	0	0.84	0
40,5	0	0	0.05	0	0.05	0
45,2	0	0.27	0.16	0	0	0
45,3	0	0	0	0	0.22	0
45,4	0	0	0.16	0	0.16	0
45,5	0	0	0	0	0.2	0
50,2	0	0	0	0	0.52	0
50,3	0	0.59	0.16	0	0.16	0
50,4	0	0	0	0	0	0
50,5	0	0.27	0	0	0.21	0
60,3	0	0.61	0	0	0.24	0
60,4	0	0.11	0.13	0	0	0
60,5	0	0	0	0	0	0
70,3	0	0.48	0	0	0.25	0
70,4	0	0.77	0	0	0	0
70,5	0	0.66	0.14	0	0.05	0
80,3	0	0.16	0.16	0	0.16	0
80,4	0	0.58	0	0	0	0
80,5	0	0.41	0	0	0.28	0
90,3	0	0.46	0	0	0.37	0
90,4	0	0.28	0	0	0.4	0
90,5	0	0	0	0	0.34	0
100,3	0	0.27	0.15	0	0.13	0
100,4	0	0.14	0.14	0	0.48	0
100,5	0	0	0	0	0.41	0
Average	0	0.277	0.047	0.000	0.193	0

factory as follows:

$CC^i_{jf} = \sum_{k=1}^{m} P_{jkf} + t_f + CC^{i-1}_{jf} \quad \forall f, \forall j$;

Step 3: For each product calculate the maximum completion time (Ready time) among all the components as follows: $R^i_j = \max_f \{CC^i_{jf}\} \quad \forall j$;

Step 4: For each product calculate $RR^i_j = R^i_j - R^{i-1}_j$

Step 5: Find the product j which has $\min_j \{s_j,\ RR^i_j\}$;

Step 6 : If the minimum value in step 5 is related to the assembly stage, put the selected product on the last part of the sequence (immediately before the products assigned in previous stages), set: $CC^i_{jf} = CC^{i-1}_{jf}$, $i = i+1$, Otherwise, put the selected product on the first part of the sequence (immediately after the products assigned in previous stages), set: $i = i+1$

Step 7: Remove the scheduled product from the set J and back to the step 2, repeat this until no product remains and $J = \varnothing$;

Heuristic 4 can record the sum of completion times in each factory at each iteration and identify the suitable product to be added to the current schedule with respect to the Johnson's algorithm. Using this method, the component with the maximum completion time at the first stage is considered a critical component; however, if the assembly factory is busy at this time, the assembly process cannot begin. The proposed Heuristic 5 compares R^i_j which is found in step 3 with the completion time of the previous product in the sequence shown by CT^{i-1}_j. If there are some products with $R^i_j < CT^{i-1}_j$, then a product with a larger R^i_j will be assigned to the current schedule.

Heuristic 5

Step 1: Initialize $i = 1,\ J = \{j_1, j_2, \ldots, j_n\}$, $CC^0_{jf} = 0,\ CT^0_j = 0$;

Step 2: For each product calculate the completion time of the components among all the factories plus transportation time to the assembly factory as follows:

$CC^i_{jf} = \sum_{k=1}^{m} P_{jkf} + t_f + CC^{i-1}_{jf} \quad \forall f, \forall j$;

Step 3: For each product calculate the maximum completion time (Ready time) among all the components as follows: $R^i_j = \max_f \{CC^i_{jf}\} \quad \forall j$;

Step 4: For each product calculate $RR^i_j = R^i_j - R^{i-1}_j$

Step 5: If there are some products with $R^i_j < CT^{i-1}_j$, Then find the product j with $\min_j \{CT^{i-1}_j - R^i_j\}$ and go to step 8, Otherwise, go to step 6;

Step 6: Find the product j which has $\min_j \{s_j,\ RR^i_j\}$;

Step 7: If the minimum value in step 6 is related to the assembly stage, put the selected product on the last part of the sequence (immediately before the products assigned in previous stages), set $CC^i_{jf} = CC^{i-1}_{jf}$, $CT^i_j = CT^{i-1}_j$, $i = i+1$ and go to step 9 ;

Otherwise, go to step 8;

Step 8: put the selected product on the first part of the sequence (immediately after the products assigned in previous stages), set: $i = i+1$ and go to the next step;

Step 9: Remove the scheduled product from the set J and back to the

Table 7
The gap percentage for class *C*.

$n,f,m=3$	H1	H2	H3	H4	H5	B&B
15,2	0	0	0	0	2.44	0
15,3	0	3.07	0	0	0	0.58
20,2	0	1.63	0.4	0	2.75	0
20,3	0	1.73	0	0	2.34	2.34
25,2	0	0	0	0	0	0
25,3	0	0	0	0	0	0
25,4	0	0	0	0	0	2.58
30,2	0	0	0	0	0	0
30,3	0	0	0	0	0	0
30,4	0	0.34	0.34	0	0.34	0
35,2	0	0.49	0.08	0	0	0.49
35,3	0	0.87	0.21	0	0.21	0.21
35,4	0	0	0	0	0	0
40,2	0	0	0	0	0	1.06
40,3	0	0	0	0	0	0.88
40,4	0	0.66	0.13	0	0.13	0
40,5	0	0	0	0	0	0
45,2	0	0.54	0	0	0.54	0
45,3	0	1.36	0.46	0	0.17	0
45,4	0	0	0	0	0	0
45,5	0	0	0	0	0	0
50,2	0	0.47	0	0	0	0
50,3	0	0.18	0.18	0	0	0
50,4	0	0.07	0	0	0.15	0
50,5	0	0	0	0	0	0
60,3	0	0	0	0	0	0
60,4	0	0.69	0.15	0	0	0
60,5	0	0.69	0.23	0	0	0
70,3	0	0.63	0.11	0	0.63	0
70,4	0	0	0	0	0	0
70,5	0	0	0	0	0	0
80,3	0	0	0	0	0	0.34
80,4	0	0.7	0.15	0	0.26	0
80,5	0	0	0	0	0	0
90,3	0	0	0	0	0	0
90,4	0	0.91	0	0	0.21	0
90,5	0	0.47	0.22	0	0.22	0
100,3	0	0	0	0	0	0
100,4	0	0	0	0	0	0
100,5	0	0	0	0	0	0
Average	0.000	0.388	0.067	0.000	0.260	0.212

step 2, repeat this until no product remains and $J=\varnothing$;

As it can be observed in step 5, Heuristic 5 tries to find a product with a minimum waiting time to start the assembly process. For this purpose, if there is only one product with $R_j^i < CT_j^{i-1}$, then this product is assigned to the first part of the sequence. If there are more than one product with $R_j^i < CT_j^{i-1}$, then the value of $CT_j^{i-1} - R_j^i$ will be calculated for these products and the selected product is the one which obtains the minimum value.

8. Computational experiments

In this section, the proposed solution procedures are tested on different scales to analyze their performance. The proposed MILP model is coded in Gams 24.1.2 and solved using CPLEX. The heuristics and the branch and bound algorithm are programmed in Matlab R2017a and together with the MILP model are executed on a personal computer with an Intel Core i5 processor and 8GB of RAM.

It is clear that the procedure of the heuristics is based on the range of data from which the processing times and the assembly times are generated. Therefore, the test problems are provided in three classes with specified data ranges. In each class, the processing times (P), and the assembly times (s), are generated randomly via uniform distribution as follows:

Class : $P\left(1,\frac{100}{m}\right)$, $s(1,60)$
Class : $P\left(1,\frac{60}{m}\right)$, $s(1,100)$
Class : $P\left(1,\frac{100}{m}\right)$, $s(1,100)$

It is evident from class *A* that processing times are wide, and the average sum of processing times is larger than the assembly times. In class *B*, the assembly times are wide, and their average is larger than the average sum of processing times. In class *C*, the range of the processing times and the assembly times are equal. It is important to note that for each component, the sum of processing times on machines is calculated in all the proposed methods. Therefore, the upper bound of processing times is divided by m so that the problem does not depend on the value of m. The classification of the test problems concerning the range of the input parameters was previously used by [78].

For the small-size problems, we use the sets provided in Table 3 as the scale to classify the test problems. For each set in Table 3, the total number of products, factories, and machines are presented respectively. For each set, the processing times and the assembly times are generated randomly in order to provide the 10 different test problems. For each problem, the MILP model and the heuristics are executed, and the solution gap is obtained through the following equation:

$$GAP = \frac{OF_i - OF_{opt}}{OF_i} * 100 \tag{20}$$

In Eq. (20), OF_i indicates the objective function value obtained by method i, and OF_{opt} indicates the optimum value of the objective function for the related problem. Since our experiments on small-size problems did not show a significant difference among the test classes, we report all the results with respect to class *C* for this group of problems. Table 4 shows the average gap for the 12 sets. There is no gap for 9 out of the total of 12 sets, which means that all the proposed solution procedures achieved the optimal solution. For all instances, the MILP model, and branch and bound algorithm provide the best solutions. The runtime of all the heuristics and the branch and bound algorithm is less than one second hence it is not presented in Table 4. The MILP model is not able to finish the execution process for the sets 9,11, and 12 in less than 3600 s. When calculating the gap for these sets, the final MILP solution is compared to the other methods.

Regarding the large size problems, the proposed heuristics and the branch and bound algorithm are executed, and the solution gap is obtained through the following equation:

$$GAP = \frac{OF_i - OF_{best}}{OF_{best}} * 100 \tag{21}$$

In Eq. (21), OF_i indicates the objective function value obtained by the method i, and OF_{best} indicates the best value obtained among all the solution procedures. Table 5 shows the results for class A. As it can be seen, all the proposed solution procedures result in zero gap, which means the best solution was found using these methods.

Table 6 shows the results for class *B*. As can be observed, among all the methods proposed, Heuristic 1, Heuristic 4, and the branch and bound algorithm show the best performances. The solutions obtained by these methods are the best solutions for all the tested problems in class *B*. As for the other methods, Heuristics 3, 5, and 2 are the next in line.

Table 7 shows the gaps related to class *C*. Heuristic 1, and Heuristic 4 show the best performance in this class with no gap. Heuristic 3, and the branch and bound method also perform well in this class with a negligible gap. Heuristic 5 and Heuristic 2 are ranked next, while Heuristic 5 has a smaller gap than Heuristic 2.

A comparison of the performance of the proposed solution methods with respect to the different classes is shown in Fig. 6. From this illustration, we can result that all the proposed methods show the best performance in class *A*. In class *B*, Heuristic 2, Heuristic 3, and Heuristic 5 show the gap increment while the other methods still have no gap. In class *C*, the overall average gap is more than that of class *B* since the gap between heuristics 2,3,5 and the B&B algorithm has increased. Heuristic 1 and Heuristic 4 still have no gap in class *C*.

In order to have a better view of the performance of the proposed solution procedures, larger instances with a maximum of 200 products are tested. The gap of the proposed solution procedures is obtained

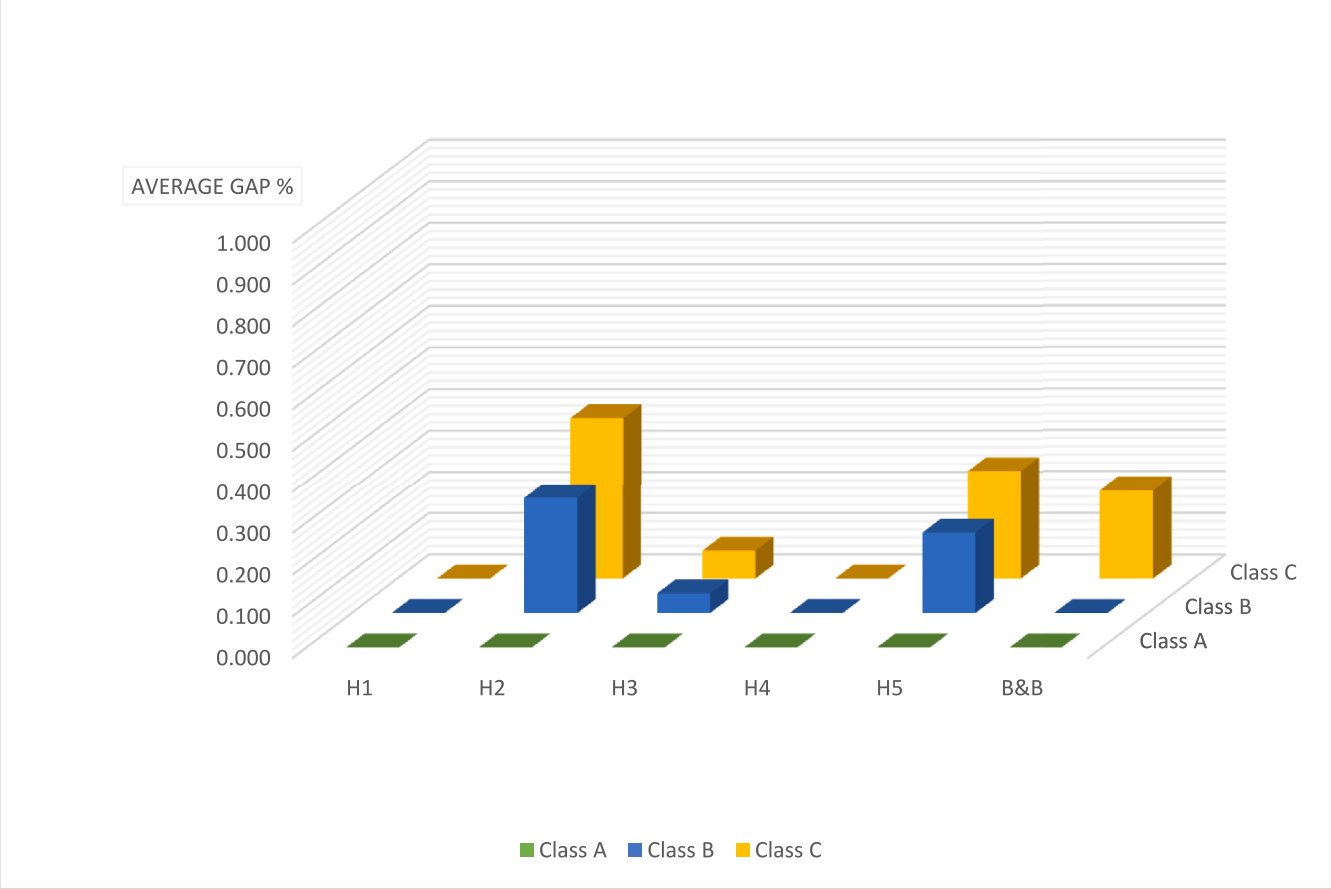


Fig. 6. The comparative performance of the proposed algorithms in different classes.

Table 8
The large-scale result of class *A*.

$n.f.m = 3$	H1	H2	H3	H4	H5	B&B
120,3	0	0	0	0	0	0
120,4	0	0	0	0	0	0
120,5	0	0	0	0	0	0
140,3	0	0	0	0	0	0
140,4	0	0	0	0	0	0
140,5	0	0	0	0	0	0
160,3	0	0	0	0	0	0
160,4	0	0	0	0	0	0
160,5	0	0	0	0	0	0
180,3	0	0	0	0	0	0
180,4	0	0	0	0	0	0
180,5	0	0	0	0	0	0
200,3	0	0	0	0	0	0
200,4	0	0	0	0	0	0
200,5	0	0	0	0	0	0
Average						

Table 9
The large-scale result of class *B*.

$n.f.m = 3$	H1	H2	H3	H4	H5	B&B
120,3	0	0	0	0	0	0
120,4	0	0.26	0.15	0	0.26	0
120,5	0	0.13	0	0	0.13	0
140,3	0	0.06	0.06	0	0.06	0
140,4	0	0.02	0	0	0.14	0
140,5	0	0.21	0	0	0.14	0
160,3	0	0	0	0	0	0
160,4	0	0.21	0.07	0	0.07	0
160,5	0	0	0	0	0	0
180,3	0	0.1	0	0	0	0
180,4	0	0.14	0	0	0	0
180,5	0	0.08	0	0	0.22	0
200,3	0	0.1	0.1	0	0.18	0
200,4	0	0.08	0	0	0.16	0
200,5	0	0	0.1	0	0	0
Average	0	0.093	0.032	0	0.091	0

through the same method from Eq. (21). The results of the three classes are provided in Tables 8–10 respectively. As shown in Table 8, for all the test problems in class *A*, the proposed solution procedures result in zero gap. According to Table 9, Heuristic 1, Heuristic 4, and the branch and bound method can find the best solution for all the tested instances in class *B*. The other methods have a negligible gap in this class. Regarding class *C*, Heuristic 4 can find the best solution for all the tested instances. Heuristic 1 has a minimal gap and, except for one set, produces the best solution for all other test instances. Heuristic 3 has a negligible gap which equals 0.054. Heuristic 1, Heuristic 5, and the branch and bound algorithm are ranked next in class *C*.

For the larger instances, the comparative performance of the proposed solution methods is shown in Fig. 7. The critical point is that when the size of the instances increases, the gap of the heuristics decreases, however, the gap in the branch and bound algorithm increases. All of the heuristics still have no gap in class *A*. Class *B* shows a gap increment of the heuristics 1,3, and 5; however, the other methods show no gap. The maximum gap between the proposed solution procedures can be seen in class *C*. Heuristic 4 still keeps its best performance in all the classes with no gap.

As the size of the instances increases, it is important to note the increment trend of the running time of the proposed algorithms. For the largest instances, Heuristics 1, 2, and 3 run in less than one second. A comparison of the running times of Heuristics 4 and 5 and the branch

Table 10
The large-scale result of class *C*.

$n,f,m = 3$	H1	H2	H3	H4	H5	B&B
120,3	0	0.22	0	0	0	0.22
120,4	0	0.15	0.15	0	0.25	0
120,5	0	0	0.01	0	0.19	0.11
140,3	0	0	0	0	0	1.05
140,4	0	0.12	0.05	0	0.19	0
140,5	0	0.13	0.05	0	0.19	0.28
160,3	0	0.16	0	0	0	0.41
160,4	0.12	0.54	0.12	0	0.12	0.32
160,5	0	0.24	0.11	0	0.24	0.24
180,3	0	0	0	0	0.44	0.5
180,4	0	0	0.18	0	0.18	0
180,5	0	0.16	0	0	0	0.16
200,3	0	0	0	0	0.31	0.37
200,4	0	0.04	0.08	0	0.26	0.08
200,5	0	0	0	0	0	0
Average	0.008	0.117	0.054	0	0.158	0.249

and bound algorithm versus the number of products can be seen in Fig. 8. According to Fig. 8, the branch and bound algorithm has a much faster growth rate than Heuristics 4 and 5. Heuristic five still has a faster growth rate of running time compared to Heuristic 4.

9. Discussion and future direction

The multi-factory two-stage assembly scheduling problem is investigated in this research. In the studied problem, each factory is qualified to perform a specific task in the network. The factories located in the first stage will manufacture the different components of a final product independently. The assembly factory in the second stage will assemble the components to finish the production process. Although the factories are considered as independent entities, they are integrated for the common objective in the network. The objective is minimizing the makespan of the entire process. To reach this objective, all the factories in the first stage will share their sequence order, which is coordinated with the assembly factory.

The computational experiments indicate that the performance of the proposed solution procedures is sensitive to the range of data from which the processing times and assembly times are generated. When the assembly times are tight, close to each other, and generally less than the processing times, almost all the solution procedures can find the best solutions. Conversely, when the processing times are tight, close to each other, and generally less than the assembly times, the branch and bound algorithm and Heuristics 1 and 4 show the best performance. When the processing times and the assembly times are wide and generated in the same range, Heuristic 4 shows the best performance. Overall, Heuristic 4 shows the best performance among all the tested problems. The other proposed methods still have a very small gap that indicates their efficiency.

This research assumes sufficient trucks are available to transport the components to the assembly factory, as stated earlier. Researchers could expand this work by considering the capacity limitation of the trucks, as well as routing and shipping methods for this new challenging problem. In this case, the solution procedures can be compared to the state of the art methods including [79–83].

In general, such a supply chain can be faced with many objectives according to different stakeholders; hence many-objective optimization algorithms with learning mechanisms to adjust the problem features like [84] are very important to be used in this area. This can be compared with other dynamic scheduling methods in the literature such as [85, 86]. The mentioned methods generally rely on centralized intelligent algorithms and data analytics, assuming that all manufacturing data could be gathered on time. On the other hand, the incorporation of decentralized and self-organizing methodologies, including smart

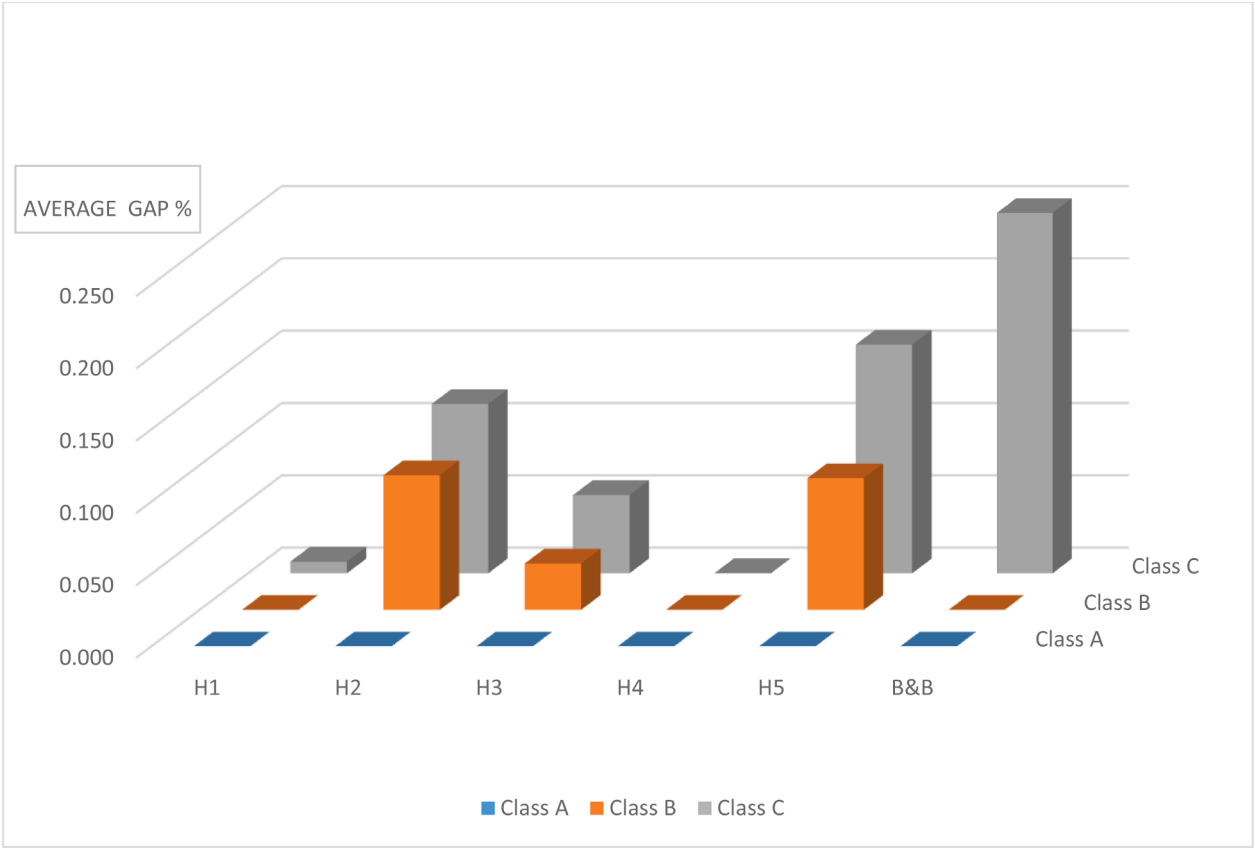


Fig. 7. The comparative performance of the algorithms for the larger-scale problems.

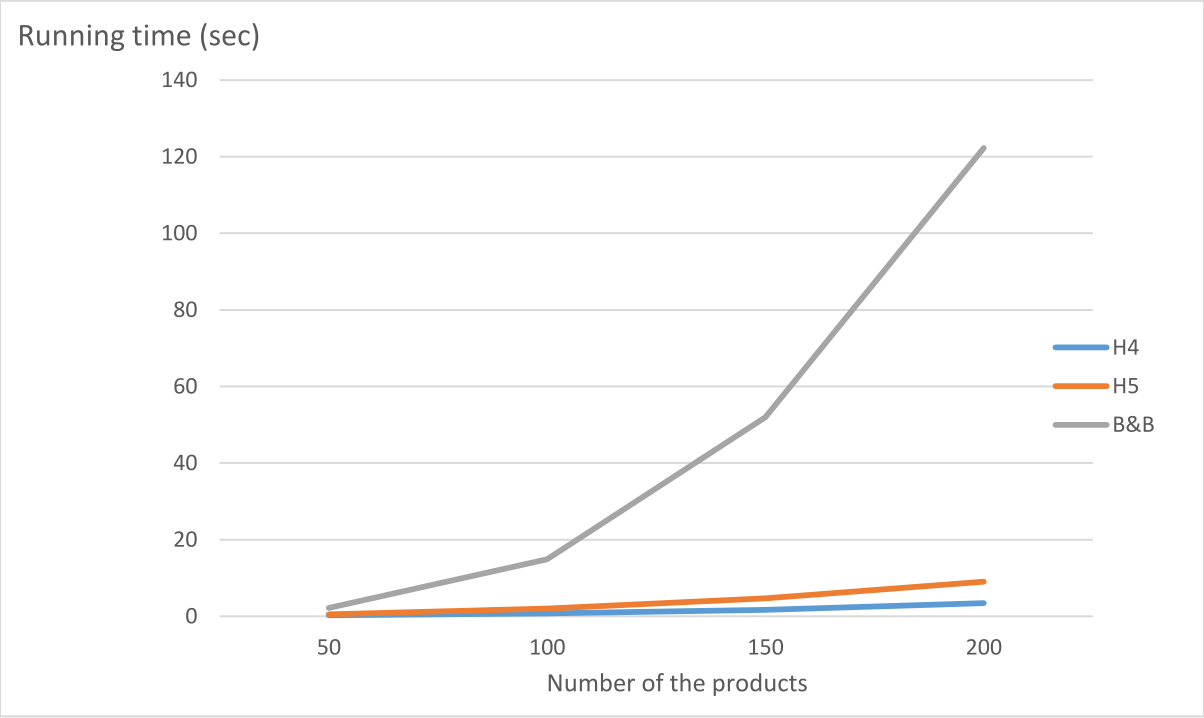


Fig. 8. The running time increase trend versus the number of products.

contracts based on blockchain technology and chemical signatures that enhance the capabilities of multi-agent autonomous process control, holds significant attraction for addressing this issue. These kinds of methods typically construct a multi-agent system using auction models or contract nets, like blockchain-based smart contracts, assuming that the data are spread across various nodes.

Data availability statement

The numerical data with respect to the three test classes, which support the findings of this study is provided in https://www.structure.plus/. It will be available from the corresponding author, [Hamed Kazemi], upon reasonable request.

CRediT authorship contribution statement

Hamed Kazemi: Conceptualization, Methodology, Software, Data curation, Writing – original draft. **Mustapha Nourelfath:** Supervision, Validation, Writing – review & editing. **Michel Gendreau:** Supervision.

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] J. Behnamian, S.M.T. Fatemi Ghomi, A survey of multi-factory scheduling, J. Intell. Manuf. 27 (1) (2016) 231–249.
[2] A. Hamzadayı, M.A. Arvas, A. Elmi, Distributed assembly permutation flow shop problem; Single seekers society algorithm, J. Manuf. Syst. 61 (2021) 613–631.
[3] J. Lohmer, R. Lasch, Production planning and scheduling in multi-factory production networks: a systematic literature review, Int. J. Prod. Res. 59 (7) (2021) 2028–2054.
[4] Q. Liu, et al., Mathematical model and discrete artificial Bee Colony algorithm for distributed integrated process planning and scheduling, J. Manuf. Syst. 61 (2021) 300–310.
[5] C. Wang, Z. Bi, L.D. Xu, IoT and cloud computing in automation of assembly modeling systems, IEEE Trans. Ind. Inform. 10 (2) (2014) 1426–1434.
[6] H.P. Wiendahl, S. Lutz, Production in Networks, CIRP Ann. 51 (2) (2002) 573–586.
[7] S. Hatami, R. Ruiz, C. Andrés-Romano, The distributed assembly permutation flowshop scheduling problem, Int. J. Prod. Res. 51 (17) (2013) 5292–5308.
[8] C.N. Potts, et al., The two-stage assembly scheduling problem: complexity and approximation, Oper. Res. 43 (2) (1995) 346–355.
[9] C.-Y. Lee, T.C.E. Cheng, B.M.T. Lin, Minimizing the makespan in the 3-machine assembly-type flowshop scheduling problem, Manage Sci. 39 (5) (1993) 616–625.
[10] A.M. Fathollahi-Fard, L. Woodward, O. Akhrif, Sustainable distributed permutation flow-shop scheduling model based on a triple bottom line concept, J. Ind. Inform. Integr. 24 (2021) 100233.
[11] K. Johansen, M. Comstock, M. Winroth, Coordination in collaborative manufacturing mega-networks: a case study, J. Eng. Technol. Manage. 22 (3) (2005) 226–244.
[12] T. Sawik, Coordinated supply chain scheduling, Int. J. Prod. Econ. 120 (2) (2009) 437–451.
[13] F. H'Mida, P. Lopez, Multi-site scheduling under production and transportation constraints, Int. J. Comput. Integr. Manuf. 26 (3) (2013) 252–266.
[14] J.-Y. Huang, M.-J. Yao, On the optimal lot-sizing and scheduling problem in serial-type supply chain system using a time-varying lot-sizing policy, Int. J. Prod. Res. 51 (3) (2013) 735–750.
[15] N. Karimi, H. Davoudpour, A branch and bound method for solving multi-factory supply chain scheduling with batch delivery, Expert Syst. Appl. 42 (1) (2015) 238–245.
[16] N. Karimi, H. Davoudpour, A knowledge-based approach for multi-factory production systems, Comput. Oper. Res. 77 (2017) 72–85.
[17] B. Naderi, R. Ruiz, The distributed permutation flowshop scheduling problem, Comput. Oper. Res. 37 (4) (2010) 754–768.
[18] B. Naderi, R. Ruiz, A scatter search algorithm for the distributed permutation flowshop scheduling problem, Eur. J. Oper. Res. 239 (2) (2014) 323–334.
[19] R. Ruiz, Q.-K. Pan, B. Naderi, Iterated Greedy methods for the distributed permutation flowshop scheduling problem, Omega (Westport) 83 (2019) 213–222.
[20] A. Hamzadayı, An effective benders decomposition algorithm for solving the distributed permutation flowshop scheduling problem, Comput. Oper. Res. 123 (2020) 105006.
[21] Z. Shao, W. Shao, D. Pi, Effective heuristics and metaheuristics for the distributed fuzzy blocking flow-shop scheduling problem, Swarm Evol. Comput. 59 (2020) 100747.
[22] H.H. Miyata, M.S. Nagano, Optimizing distributed no-wait flow shop scheduling problem with setup times and maintenance operations via iterated greedy algorithm, J. Manuf. Syst. 61 (2021) 592–612.
[23] V. Fernandez-Viagas, P. Perez-Gonzalez, J.M. Framinan, The distributed permutation flow shop to minimise the total flowtime, Comput. Ind. Eng. 118 (2018) 464–477.
[24] Q.-K. Pan, et al., Effective heuristics and metaheuristics to minimize total flowtime for the distributed permutation flowshop problem, Expert Syst. Appl. 124 (2019) 309–324.

[25] N. Zhu, et al., A discrete learning fruit fly algorithm based on knowledge for the distributed no-wait flow shop scheduling with due windows, Expert Syst. Appl. 198 (2022) 116921.
[26] W. Shao, Z. Shao, D. Pi, Modeling and multi-neighborhood iterated greedy algorithm for distributed hybrid flow shop scheduling problem, Knowl. Based Syst. 194 (2020) 105527.
[27] K.-C. Ying, S.-W. Lin, Minimizing makespan for the distributed hybrid flowshop scheduling problem with multiprocessor tasks, Expert Syst. Appl. 92 (2018) 132–141.
[28] J. Cai, R. Zhou, D. Lei, Dynamic shuffled frog-leaping algorithm for distributed hybrid flow shop scheduling with multiprocessor tasks, Eng. Appl. Artif. Intell. 90 (2020) 103540.
[29] W. Shao, D. Pi, Z. Shao, Optimization of makespan for the distributed no-wait flow shop scheduling problem with iterated greedy algorithms, Knowl. Based Syst. 137 (2017) 163–181.
[30] W. Shao, Z. Shao, D. Pi, Effective constructive heuristics for distributed no-wait flexible flow shop scheduling problem, Comput. Oper. Res. 136 (2021) 105482.
[31] H. Qin, et al., Integrated production and distribution scheduling in distributed hybrid flow shops, Memetic Comput. 13 (2) (2021) 185–202.
[32] K. Ying, S. Lin, Minimizing Makespan in Distributed Blocking Flowshops Using Hybrid Iterated Greedy Algorithms, 5, IEEE Access, 2017, pp. 15694–15705.
[33] F. Zhao, et al., A heuristic and meta-heuristic based on problem-specific knowledge for distributed blocking flow-shop scheduling problem with sequence-dependent setup times, Eng. Appl. Artif. Intell. 116 (2022) 105443.
[34] G. Zhang, K. Xing, Differential evolution metaheuristics for distributed limited-buffer flowshop scheduling with makespan criterion, Comput. Oper. Res. 108 (2019) 33–43.
[35] J. Deng, L. Wang, A competitive memetic algorithm for multi-objective distributed permutation flow shop scheduling problem, Swarm Evol. Comput. 32 (2017) 121–131.
[36] A.P. Rifai, H.-T. Nguyen, S.Z.M. Dawal, Multi-objective adaptive large neighborhood search for distributed reentrant permutation flow shop scheduling, Appl. Soft Comput. 40 (2016) 42–57.
[37] J. Behnamian, S.M.T. Fatemi Ghomi, The heterogeneous multi-factory production network scheduling with adaptive communication policy and parallel machine, Inf. Sci. (Ny) 219 (2013) 181–196.
[38] M. Yazdani, S. Gohari, B. Naderi, Multi-factory parallel machine problems: improved mathematical models and artificial bee colony algorithm, Comput. Ind. Eng. 81 (2015) 36–45.
[39] F. Xiong, et al., Minimizing the total completion time in a distributed two stage assembly system with setup times, Comput. Oper. Res. 47 (2014) 92–105.
[40] F. Xiong, K. Xing, Meta-heuristics for the distributed two-stage assembly scheduling problem with bi-criteria of makespan and mean completion time, Int. J. Prod. Res. 52 (9) (2014) 2743–2766.
[41] J. Deng, et al., x, Int. J. Prod. Res. 54 (12) (2016) 3561–3577.
[42] Y. Hou, et al., Modelling and optimization of integrated distributed flow shop scheduling and distribution problems with time windows, Expert Syst. Appl. 187 (2022) 115827.
[43] X.T. Sun, S.H. Chung, F.T.S. Chan, Integrated scheduling of a multi-product multi-factory manufacturing system with maritime transport limits, Transp. Res. Part E: Logist. Transp. Rev. 79 (2015) 110–127.
[44] J. Behnamian, Decomposition based hybrid VNS–TS algorithm for distributed parallel factories scheduling with virtual corporation, Comput. Oper. Res. 52 (2014) 181–191.
[45] J. Behnamian, Matheuristic for the decentralized factories scheduling problem, Appl. Math. Model. 47 (2017) 668–684.
[46] S.H. Chung, et al., Application of genetic approach for advanced planning in multi-factory environment, Int. J. Prod. Econ. 127 (2) (2010) 300–308.
[47] Y.-Y. Huang, Q.-K. Pan, L. Gao, An effective memetic algorithm for the distributed flowshop scheduling problem with an assemble machine, Int. J. Prod. Res. 61 (6) (2023) 1755–1770.
[48] S. Wang, L. Wang, An estimation of distribution algorithm-based memetic algorithm for the distributed assembly permutation flow-shop scheduling problem, IEEE Trans. Syst. Man Cybern. Syst. 46 (1) (2016) 139–149.
[49] S. Hatami, R. Ruiz, C. Andrés-Romano, Heuristics and metaheuristics for the distributed assembly permutation flowshop scheduling problem with sequence dependent setup times, Int. J. Prod. Econ. 169 (2015) 76–88.
[50] J. Lin, S. Zhang, An effective hybrid biogeography-based optimization algorithm for the distributed assembly permutation flow-shop scheduling problem, Comput. Ind. Eng. 97 (2016) 128–136.
[51] J. Lin, Z.-J. Wang, X. Li, A backtracking search hyper-heuristic for the distributed assembly flow-shop scheduling problem, Swarm Evol. Comput. 36 (2017) 124–135.
[52] H.-Y. Sang, et al., Effective invasive weed optimization algorithms for distributed assembly permutation flowshop problem with total flowtime criterion, Swarm Evol. Comput. 44 (2019) 64–73.
[53] H.-B. Song, J. Lin, A genetic programming hyper-heuristic for the distributed assembly permutation flow-shop scheduling problem with sequence dependent setup times, Swarm Evol. Comput. 60 (2021) 100807.
[54] J. Huang, X. Gu, Distributed assembly permutation flow-shop scheduling problem with sequence-dependent set-up times using a novel biogeography-based optimization algorithm, Eng. Optimiz. (2021) 1–21.
[55] D. Ferone, et al., A biased-randomized iterated local search for the distributed assembly permutation flow-shop problem, Int. Trans. Operat. Res. 27 (3) (2020) 1368–1391.
[56] Y.-Y. Huang, et al., An improved iterated greedy algorithm for the distributed assembly permutation flowshop scheduling problem, Comput. Ind. Eng. 152 (2021) 107021.
[57] X. Wu, X. Liu, N. Zhao, An improved differential evolution algorithm for solving a distributed assembly flexible job shop scheduling problem, Memetic Comput. 11 (4) (2019) 335–355.
[58] F. Zhao, et al., A cooperative water wave optimization algorithm with reinforcement learning for the distributed assembly no-idle flowshop scheduling problem, Comput. Ind. Eng. 153 (2021) 107082.
[59] X. Zhang, X.-T. Li, M.-H. Yin, An enhanced genetic algorithm for the distributed assembly permutation flowshop scheduling problem, Int. J. Bio-Inspired Comput. 15 (2) (2020) 113–124.
[60] Z. Shao, W. Shao, D. Pi, Effective constructive heuristic and metaheuristic for the distributed assembly blocking flow-shop scheduling problem, Appl. Intell. 50 (12) (2020) 4647–4669.
[61] Y. Yang, X. Li, A knowledge-driven constructive heuristic algorithm for the distributed assembly blocking flow shop scheduling problem, . Expert Syst. Appl. 202 (2022) 117269.
[62] Z.-Q. Zhang, et al., A matrix cube-based estimation of distribution algorithm for the energy-efficient distributed assembly permutation flow-shop scheduling problem, Expert Syst. Appl. 194 (2022) 116484.
[63] F. Zhao, et al., A self-learning hyper-heuristic for the distributed assembly blocking flow shop scheduling problem with total flowtime criterion, Eng. Appl. Artif. Intell. 116 (2022) 105418.
[64] F. Zhao, B. Zhu, L. Wang, An estimation of distribution algorithm-based hyper-heuristic for the distributed assembly mixed no-idle permutation flowshop scheduling problem, IEEE Trans. Syst. Man Cybern. Syst. (2023) 1–12.
[65] J. Wang, D. Lei, J. Cai, An adaptive artificial bee colony with reinforcement learning for distributed three-stage assembly scheduling with maintenance, Appl. Soft Comput. 117 (2022) 108371.
[66] G. Zhang, et al., Memetic Algorithm With Meta-Lamarckian Learning and Simplex Search For Distributed Flexible Assembly Permutation Flowshop Scheduling Problem, 8, IEEE Access, 2020, pp. 96115–96128.
[67] H. Kazemi, Mohammad M. Mazdeh, M. Rostami, The two stage assembly flow-shop scheduling problem with batching and delivery, Eng. Appl. Artif. Intell. 63 (2017) 98–107.
[68] A. Mozdgir, et al., Two-stage assembly flow-shop scheduling problem with non-identical assembly machines considering setup times, Int. J. Prod. Res. 51 (12) (2013) 3625–3642.
[69] J. Navaei, et al., Heuristics for an assembly flow-shop with non-identical assembly machines and sequence dependent setup times to minimize sum of holding and delay costs, Comput. Oper. Res. 44 (2014) 52–65.
[70] G. Zhang, K. Xing, F. Cao, Scheduling distributed flowshops with flexible assembly and set-up time to minimise makespan, Int. J. Prod. Res. 56 (9) (2018) 3226–3244.
[71] K.-C. Ying, et al., Supply chain-oriented permutation flowshop scheduling considering flexible assembly and setup times, Int. J. Prod. Res. (2020) 1–24.
[72] S. Yang, Z. Xu, The distributed assembly permutation flowshop scheduling problem with flexible assembly and batch delivery, Int. J. Prod. Res. (2020) 1–19.
[73] L. De Giovanni, F. Pezzella, An improved genetic algorithm for the distributed and flexible job-shop scheduling problem, Eur J Oper Res 200 (2) (2010) 395–408.
[74] Q. Luo, et al., A distributed flexible job shop scheduling problem considering worker arrangement using an improved memetic algorithm, Expert Syst. Appl. 207 (2022) 117984.
[75] F. Marandi, S.M.T. Fatemi Ghomi, Integrated multi-factory production and distribution scheduling applying vehicle routing approach, Int. J. Prod. Res. 57 (3) (2019) 722–748.
[76] J. Navaei, et al., Two-stage flow-shop scheduling problem with non-identical second stage assembly machines, Int. J. Adv. Manuf. Technol. 69 (9) (2013) 2215–2226.
[77] X. Sun, K. Morizawa, H. Nagasawa, Powerful heuristics to minimize makespan in fixed, 3-machine, assembly-type flowshop scheduling, Eur. J. Oper. Res. 146 (3) (2003) 498–516.
[78] H. Kazemi, et al., The integrated production-distribution scheduling in parallel machine environment by using improved genetic algorithms, J. Ind. Prod. Eng. 38 (3) (2021) 157–170.
[79] M.A. Dulebenets, An adaptive polyploid memetic algorithm for scheduling trucks at a cross-docking terminal, Inf. Sci. (Ny) 565 (2021) 390–421.
[80] M. Kavoosi, et al., Berth scheduling at marine container terminals, Marit. Bus. Rev. 5 (1) (2020) 30–66.
[81] J. Pasha, et al., Exact and metaheuristic algorithms for the vehicle routing problem with a factory-in-a-box in multi-objective settings, Adv. Eng. Inform. 52 (2022) 101623.
[82] M.A. Dulebenets, A novel memetic algorithm with a deterministic parameter control for efficient berth scheduling at marine container terminals, Marit. Bus. Rev. 2 (4) (2017) 302–330.
[83] R. Masoud, O.-A. Nastaran, A.-S. Niloofar, Ambulance routing in disaster response considering variable patient condition: NSGA-II and MOPSO algorithms, J. Ind. Manage. Optimiz. 18 (2) (2022) 1035–1062.
[84] H. Zhao, C. Zhang, An online-learning-based evolutionary many-objective algorithm, Inf. Sci. (Ny) 509 (2020) 1–21.
[85] S. Yang, J. Wang, Z. Xu, Real-time scheduling for distributed permutation flowshops with dynamic job arrivals using deep reinforcement learning, Ad. Eng. Inform. 54 (2022) 101776.
[86] F. Zhao, et al., A reinforcement learning-driven brain storm optimisation algorithm for multi-objective energy-efficient distributed assembly no-wait flow shop scheduling problem, Int. J. Prod. Res. 61 (9) (2023) 2854–2872.