Operational Research (2024) 24:39
https://doi.org/10.1007/s12351-024-00844-7

ORIGINAL PAPER

Two-machine decentralized flow shop scheduling problem with inter-factory batch delivery system

Mohammad Rostami[1] · **Milad Mohammadi**[1]

Received: 13 December 2022 / Revised: 26 March 2024 / Accepted: 28 May 2024 /
Published online: 1 July 2024

Abstract

Technological advancement, the emergence of more complex production systems, and the need for greater manufacturer competition have caused production systems to shift from a centralized environment to a decentralized one. Researchers have paid more attention to distributed flow shop scheduling problems and investigated various features and issues related to them in recent years. However, special types of these problems, in which the network structure is serial and inter-factory transportation is significant, have received less attention. This study investigated the two-machine decentralized flow shop scheduling problem, in which inter-factory transportation is handled by a batch delivery system. The goal was to simultaneously reduce the costs of makespan and batch delivery. A mixed-integer linear programming model capable of solving small-size instances in a logical running time was presented to better describe the problem. Then, in order to solve large-size instances in a logical running time, a fast branch and bound algorithm with a heuristic method were developed to obtain the appropriate upper bound as well as the tight lower bounds at each node. The computational results indicated that the B&B algorithm performed very well in terms of problem-solving running time. The findings also demonstrated that the heuristic method can solve the most complex instances by 100 jobs with less than 13% error.

Keywords Flow shop scheduling · Decentralized · Inter-factory batch delivery system · Branch and bound algorithm

✉ Mohammad Rostami
rostami_m@shahroodut.ac.ir

Milad Mohammadi
mohammadi.m.milad@gmail.com

[1] Department of Industrial Engineering and Management, Shahrood University of Technology, Shahroud, Iran

1 Introduction

Flow shop scheduling problems are among the most common in the field of production planning, where several machines in a series are in charge of completing the manufacturing process. The Flow shop production system is used in most complex industries, such as automotive assembly lines, drug production, and so on. Researchers have considered the distribution of production capacity between several production sites in the form of a supply chain in recent years due to customer needs and also improving the competitive situation Olhager and Feldmann (2018). Although it has limitations such as production capacity, capabilities, and labor, production operations at multiple sites can affect system efficiency and reduce overall network costs Lohmer and Lasch (2021).

When there are multiple production sites, transportation times can affect production planning. However, most studies on distributed flow shop scheduling problems have not taken transportation times into account Lohmer and Lasch (2021). Most distributed flow shop problems are structured in such a way that the main problem is divided into two sub-problems. Each job is assigned to one of the factories in the first sub-problem. After the job assignment, the production scheduling is considered in the second sub-problem Lei et al. (2021). One of the distributed flow shop problems is when the production network structure is in series and jobs must go through at least two production stages distributed across multiple sites to complete their production process Lei et al. (2021). Inter-factory transportation should be considered in this case because it can affect the optimal production schedule. In fact, such systems are known as decentralized flow shop systems.

Recently, the subject of production scheduling in a network consisting of multiple factories has gained significant attention. Multi-factory scheduling problems are more complicated than scheduling problems with a single factory (Marandi and Ghomi 2019). There are different ways to transport jobs between factories, one of which is the use of a batch delivery system Hall and Potts (2003). Nevertheless, the use of a batch delivery system between factories may cause delays in the completion of jobs at each stage, resulting in delays in dispatching orders to customers and their dissatisfaction. As a result, deciding between delivery and completion costs is a difficult decision in the production planning of decentralized flow shop systems. Therefore, to reduce the total cost of transportation, tardiness, and holding throughout the supply chain network, it is important to consider the job transition between factories along with the scheduling of jobs. On the one hand, production managers try to schedule jobs so that the production program is completed and orders are delivered to customers as soon as possible. However, logistics managers strive to dispatch orders as close together as possible in order to reduce delivery costs. The purpose of the current study is to investigate the two-machine decentralized flow shop scheduling problem considering the inter-factory batch delivery system, minimizing the costs of makespan plus the batch delivery costs at the same time. As an innovative aspect of the present research, a fast branch and bound algorithm with a heuristic method are developed to obtain the appropriate upper bound as well as the tight lower bounds at each node.

The production scenario in the real world behind the scheduling model presented in this research is all two-stage production industries where the production location is not centralized and the time interval between the two production stages is significant from the scheduling point of view. The application of such a model is in industries such as casting (Wanapinit et al. 2021), pharmaceutical (Salehi et al. 2022), and assembly (Villalonga et al. 2021). In the current research, a real case study in the field of casting is presented. This study proves that the investigated problem has a strong practical aspect in the real world.

The rest of this paper is organized as follows. Section 2 reviews the related literature. In Sect. 3, the problem is fully described and its assumptions are stated. Section 4 then presents a MILP that can solve small-size instances in a logical running time using commercial solvers. In Sect. 5, a fast branch and bound algorithm is developed to solve large-size instances by proposing the structural features of the problem and using the appropriate lower and upper bounds. Section 6 shows the computational results related to the evaluation of the solution methods. Finally, in Sect. 7, conclusions and future studies are proposed.

2 Literature review

At the beginning of the new century, the production scheduling problem with batch delivery systems was explored by Wang and Cheng (2000). After that, during the last two decades, a lot of research has been done in the field of supply chain scheduling. According to the literature review, most of the research in this field has been done in a single-machine environment. Hall and Potts (2003) examined the complexity of different problems based on different objective functions and used dynamic programming to solve some small-size problems. He was followed by extensive research in the single-machine environment, including the research of Mazdeh et al. (2007), Mazdeh et al. (2008), Ji et al. (2007), Yin et al. (2013), Mazdeh et al. (2013), Rostami et al. (2015), Qi and Yuan (2017), Noroozi et al.(2018), Rostami et al. (2020) and Rostami (2021). Noted that they all tried to investigate different types of this problem using exact solution methods. Due to the complexities of supply chain scheduling problems, it seems that the use of heuristic and meta-heuristic methods may be more appropriate Hamidinia et al. (2012). Among the scholars who have tried to solve supply chain scheduling problems in a single machine environment with heuristic approaches, we can mention Ahmadizar and Farhadi (2015), Abedi and Seidgar (2016), Yin et al. (2018), Noroozi et al. (2019) and Gharaei and Jolai (2021).

With the advancement of technology and more complex manufacturing processes, production environments have also moved towards the use of multiple machines. Although it is very difficult to solve supply chain scheduling problems in Flow shop and parallel environments, there are a number of researches in the literature with the help of exact methods such as branch and bound algorithms and dynamic programming, including Mazdeh and Rostami (2014), Cakici et al. (2014), Gong et al. (2015) and Kong et al. (2020). However, most scholars in this field have tried to solve large-size problems with the help of heuristic methods.

Wang et al. (2017) examined the permutation Flow shop problem by considering multiple customers in a supply chain environment. To solve this problem, they proposed two heuristic methods and a new GA-TVNS method. Considering a supply chain consisting of several serial factories joined together, Karimi and Davoudpour (2017) developed a mathematical model to solve the supply chain scheduling problem with stage-dependent inventory holding cost. Kazemi et al. (2017) tried to minimize the total tardiness in a two-stage assembly flow shop environment and used the Imperialist Competitive (IC) algorithm. Moreover, Basir et al. (2018) solved the same problem with the help of a bi-level genetic algorithm. Rostami and Shad (2020) solved the two-machine flow shop scheduling problem with a batch delivery system by developing a hybrid bee algorithm. In recent years, the distributed flow shop scheduling problem with batch delivery systems has also been considered. Li et al. (2021a, b) investigated the distributed flow shop scheduling problem with the aim of minimizing makespan and energy consumption simultaneously and used a Wale Optimization (WO) algorithm to solve the problems. Yang and Xu (2021) also examined the distributed assembly flow shop scheduling problem, in which the assembly stage is flexible and uses a batch delivery system for customers. They proposed seven different heuristic and meta-heuristic algorithms to evaluate different instances. Zhang et al. (2023) investigated the distributed flow shop scheduling with reconfigurable elements. To solve this scheduling problem, they proposed a mixed-integer linear programming model (MILP) and a nested variable neighbourhood descent algorithm. Fu et al. (2023) examined a scheduling problem considering service resource routing. In their problem, the sharing strategy is applied among multiple home health care centers. They proposed a MILP model and a multi-objective artificial bee colony algorithm to solve the problem.

The literature review clearly shows that all of the studies considering the batch delivery system in combination with the production schedule have used this system to dispatch the final products to customers. However, researchers have not considered its use for inter-factory transportation. Of course, this does not imply that inter-factory transportation has been overlooked in production scheduling problems. Moon et al. (2002) as well as Chan et al. (2008) examined plant-to-plant transportation. De Matta and Miller (2004), Ruifeng and Subramaniam (2011) and Sun et al. (2015) investigated the problem of inter-factory transportation in multi-stage production environments. Table 1 presents a tabular comparison review method in the related field. A review of these studies shows that the use of a batch delivery system for flow shop scheduling problems has not been considered in multi-factory transportation planning. However, with the help of a batch delivery system, a large part of the inter-factory logistics costs can be reduced. For this reason, this study examines for the first time the two-machine decentralized flow shop scheduling problem considering the inter-factory batch delivery system. The goal is to minimize the costs of makespan plus the batch delivery costs at the same time. A fast branch and bound algorithm with a heuristic method are developed to obtain the appropriate upper bound as well as the tight lower bounds at each node.

Table 1 Comparison review of the existing studies

Author(s)	Inter-factory transportation	Batch delivery	Flow shop	Objective Function	Solution approach
(Basir et al. 2018)		✓	✓	The minimum number of weighted tardy jobs and sum of delivery costs	A MILP model and a bi-level genetic algorithm
(Marandi and Ghomi 2019)	✓	✓		Total transportation, tardiness and holding costs	A learning-oriented SA based on Q-learning algorithm and a novel hybrid cloud theory-based learning-oriented SA
(Jia et al. 2019)		✓		Total weighted delivery times	A deterministic heuristic and two hybrid meta heuristic algorithms based on ant colony optimization
(Fu et al. 2019)			✓	Minimize makespan and total tardiness in flow shop scheduling with deteriorating jobs	A hybrid multiobjective optimization algorithm
(Dorfeshan et al. 2020)			✓	Makespan	Weighted distance-based approximation
(Shao et al. 2021)	✓		✓	Makespan, total weighted earliness and tardiness, and total workload	A multi-objective evolutionary algorithm based on multiple neighborhoods local search
(Rostami 2021)		✓		Minimize the maximum tardiness	A MILP model and a branch and bound algorithm
(Li et al. 2021a, b)	✓		✓	Makespan	A discrete artificial bee colony algorithm
(Rahman et al. 2021)		✓	✓	Total cost of tardiness and delivery cost	Three metaheuristic algorithms (differential evolution, a moth flame optimization, and levy-flight moth flame optimization algorithm
(Rossi and Nagano 2021)	✓		✓	Makespan	A MILP formulation, a novel constructive heuristic and iterated greedy algorithms
(Bachtenkirch and Bock 2022)		✓		Total sum of holding costs and transportation costs	Branch and Bound algorithm

Table 1 (continued)

Author(s)	Inter-factory transportation	Batch delivery	Flow shop	Objective Function	Solution approach
(Pan et al. 2022)			✓	Minimize the maximum completion time (Makespan) in flow shop lot-streaming scheduling	A mathematic model and five meta-heuristics
(Fu et al. 2022)			✓	Processing quality of jobs is maximized while meeting makespan requirements	A two-stage hybrid shuffled frog-leaping algorithm
(Cui et al. 2023)	✓		✓	Makespan	An improved multi-population genetic algorithm
(Rostami and Yousefzadeh 2023)		✓	✓	The cost of total completion time plus batch delivery costs	A MILP and a novel gamified teaching–learning based optimization algorithm
Zhang et al. (2024)	✓		✓	Makespan and total energy consumption	A multiobjective memetic algorithm with particle swarm optimization and Q-learning-based local search
The present study	✓	✓	✓	The costs of makespan and batch delivery	A MILP and a fast branch and bound algorithm with a heuristic method

Table designed: The authors

3 Problem definition

In the problem under study, there are n jobs that must be processed in series by two machines, the distance between the two machines is significant. After processing the jobs in the first machine, these jobs must be dispatched to the second machine using a batch delivery system. The capacity of the batches is unlimited and they have the same delivery cost. In the current batch delivery system, the continuous batching strategy is selected to form the batches. Under this strategy, the jobs that make up a batch are processed continuously. Both machines are available at time zero. Moreover, in both stages, each machine can process only one job at a time. Preemption in both stages is not allowed. The goal is to minimize the maximum completion times and batch delivery costs. Since the first part of the objective function is of the time type, it can be easily converted to cost with the help of the time-to-cost conversion factor β.

According to the standard classification of Chen (2010), the problem under study in this research can be represented in the form of $F2|De|V(\infty,\infty)\|\beta C_{\max}+\gamma D$. In this classification, $F2$ represents the two-machine flow shop environment. De means that the machines are decentralized, as a result of which the time distances between the machines will be significant. $V(\infty,\infty)$ indicates that there are enough vehicles in the batch delivery system that have sufficient capacity. In the fourth part of this classification, the number of customers is determined, which is empty due to the fact that it is without the presence of the customer. Finally, the objective function of the problem is expressed, where γ and D represent the number of dispatched batches and the cost of dispatching each batch from the first machine site to the second machine site, respectively.

4 Mixed-integer linear programming model

In this section, in order to better describe the problem as well as solve small-size instances, a mixed-integer linear programming model (MILP) is presented. The main idea of the mathematical modeling method is borrowed from the research of Mazdeh et al. (2013). Given that the capacity of vehicles is unlimited and a sufficient number of them is available, so initially as the number of jobs, i.e. n, the hypothetical batch, each of which has n positions, is considered. After completing the process on the first machine, each job is assigned to a specific batch and a specific position. Therefore, in each position of each batch, a maximum of one job can be placed. However, some batches and some positions may be left blank. After this allocation, the production scheduling process will be such that jobs in batches and positions with lower numbers will have a higher priority for processing. For a simpler explanation of this idea, Fig. 1 shows how to assign 3 jobs to batches. As can be seen, jobs 3 and 2 are initially processed, respectively, and dispatched together in a batch. Job 1 is then processed and dispatched in a

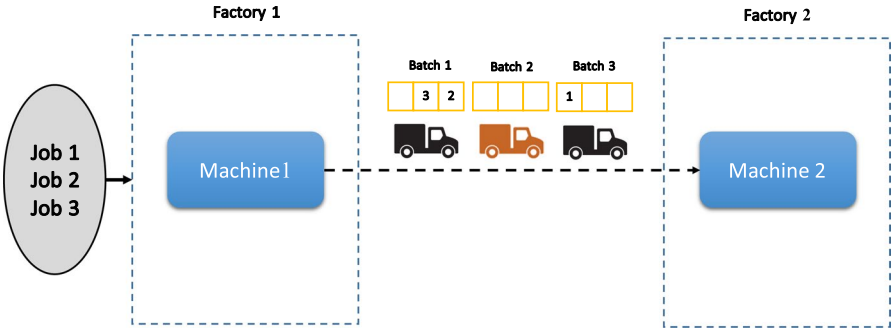


Fig. 1 An example of how jobs are assigned to different batch positions

separate batch to the second machine site. It should be noted that some hypothetical batches may remain unallocated (such as Batch 2) that will not affect the solution.

The notations used in the mathematical model are defined as follows:

Sets:

I Jobs ($i \in I$) $i=1,\ldots,n$

J Positions ($j \in J$) $j=1,\ldots,n$

K Batches ($k \in K$) $k=1,\ldots,n$

Parameters:

p_{1i} Processing time related to job i on machine 1

p_{2i} Processing time related to job i on machine 2

Dis Time distance between machine 1 and 2

D Batch delivery cost

β Time to cost coefficient

M A positive big number

Binary variables:

x_{ijk} Equals to 1 if job i is assigned to batch k and position j

y_k Equals to 1 if batch k is formed

x_{ij} Equals to 1 if job i is assigned to position j on machine 2

Positive variables:

Cb_k Completion time related to batch k

C_{jk} Completion time related to position j of batch k

S_i Arrival time of job i to machine 2

CT_j Completion time of job assigned to position j on machine 2

$C_{\max}$ Maximum completion time of jobs on machine 2

Accordingly, the linear programming model related to the problem is as follows:

$$Min\, z = \beta C_{\max} + \sum_{k \in K} y_k D \tag{1}$$

s.t:

$$C_{\max} \geq CT_j \, \forall j \tag{2}$$

$$\begin{cases} CT_j \geq CT_{j-1} + \sum_{i \in I} x_{ij} p_{2i} & \forall j (CT_0 = 0) \\ CT_j + M\left(1 - x_{ij}\right) \geq S_i + p_{2i} & \forall i, j \end{cases} \tag{3}$$

$$\begin{cases} \sum_{j \in J} x_{ij} = 1 \ \ \forall i \\ \sum_{i \in I} x_{ij} = 1 \ \ \forall j \end{cases} \tag{4}$$

$$\sum_{j \in J} \sum_{k \in K} x_{ijk} = 1 \forall i \tag{5}$$

$$\sum_{i \in I} x_{ijk} \leq 1 \forall j, k \tag{6}$$

$$\begin{cases} -\sum_{i\in I}\sum_{j\in J} x_{ijk} + My_k \geq 0 \quad \forall k \\ \sum_{i\in I}\sum_{j\in J} x_{ijk} + M\left(1 - y_k\right) > 0 \ \forall k \end{cases} \tag{7}$$

$$S_i + M\left(1 - \sum_{j\in J} x_{ijk}\right) \geq Cb_k + Dis \,\forall i, k \tag{8}$$

$$Cb_k \geq C_{jk} \,\forall j, k \tag{9}$$

$$C_{jk} \geq C_{j-1,k} + \sum_{i\in I} x_{ijk} p_{1i} \,\forall j, k \tag{10}$$

$$\begin{cases} C_{0k} = C_{n,k-1} \ \ \forall k \\ C_{n0} = 0 \end{cases} \tag{11}$$

$$\begin{aligned} &x_{ijk}, x_{ij}, y_k \in \{0, 1\} \\ &C_{\max}, Cb_k, C_{jk}, S_i, CT_j \geq 0 \end{aligned} \tag{12}$$

where Eq. (1) minimizes the value of the objective function. This objective function includes the maximum completion time, which is converted to cost by β, as well as the sum of batch delivery costs. Constraint (2) is given to calculate the makespan. Constraints (3) and (4) are given in order to calculate the completion times of the positions on the second machine. What is clear is that the process of a job on the second machine can start when, firstly, the completed part of it has arrived from the first machine (the arrival time of job i) and secondly, the second machine is empty. Equation (5) forces that each job must be assigned to only one position of a batch. Constraint (6) forces that in each position of a batch a maximum of one job can be assigned. Constraints (7) are provided to specify the formation of batches. Constraint (8) determines the arrival time of each job to the location of the second machine. Constraint (9) Calculates the completion time of each batch, which is equivalent to the maximum completion time of the positions in that batch. Constraints (10) and (11) are provided to calculate the completion time of the positions in each batch. Finally, the constraint (12) defines the decision variables.

Since the value of M has a significant effect on the performance of CPLEX, so it is important to determine the appropriate value for it so that it does not cause validity problems for the model and is not too large. Here it is suggested that $M = \sum_{i\in I} p_{1i} + \sum_{i\in I} p_{2i} + Dis$ which has both of the above conditions.

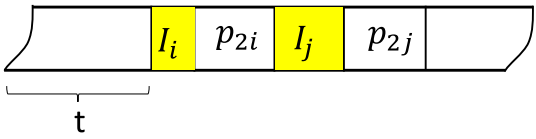


Fig. 2 Sequence of jobs in S

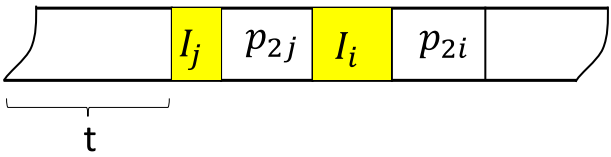


Fig. 3 Sequence of jobs in S

5 Branch and Bound algorithm

5.1 Structural features

In this section, some theorems and lemmas are presented that are used for the branch and bound algorithm.

Theorem 1. *On the second machine, the optimal sequence of jobs is based on the Earliest Arrival Time (EAT) rule. It means that the jobs with lower arrival times have priority over others.*

Proof (By interchange). After processing the first stage of the jobs on machine 1 and dispatching them by the batch delivery system, each job arrives at the second factory site (machine 2) at a specific time. Therefore, in the second factory (machine 2), the problem can be converted to the form of $1|r_i|\beta C_{\max}$, in which the release time of each job on machine 2 is equal to the arrival time of its work in process (S_i), which can be calculated from the previous stage.

It should be noted when release times exist, the Makespan will be equal to the sum of jobs processing times plus the sum of idle machine time, i.e. $\sum_i p_{2i} + \sum_i I_i$. Since $\sum_i p_{2i}$ has always a constant value, minimizing $\sum_i I_i$ can lead to minimized Makespan.

Suppose there are two jobs i and j, and $S_i > S_j$. Figure 2 illustrates the sequence S where i is followed by j, while Fig. 3 shows the sequence S' where i is preceded by j.

Here, I_i and I_j are idle machine times occur before jobs i and j in each sequence, respectively. These variables are obtained from Eqs. (13)–(16):

$$I_i^S = \max\left\{0, S_i - t\right\} \tag{13}$$

$$I_j^S = \max\left\{0, S_j - t - p_{2i} - I_i^S\right\} = 0 \tag{14}$$

$$I_j^{S'} = \max\left\{0, S_j - t\right\} \tag{15}$$

$$I_i^{S'} = \max\left\{0, S_i - t - p_{2j} - I_j^{S'}\right\} \tag{16}$$

In the above relations, t is the time that has passed before starting the partial schedule. It must be taken into account in Eq. (14) that the job with a bigger arrival time in the sequence S is processed first. Therefore, the completion time for job i is greater than the arrival time of job j, and as a result, the machine idle time before job j will always be zero. With regard to the equations below and knowing that the maximum value of relation (16) is equal to $S_i - t - p_{2j}$:

$$I_i^{S'} + I_j^{S'} = \begin{cases} 0 + \max\left\{0, S_i - t - p_{2j}\right\} & \text{if } S_j \le t \\ S_j - t + \max\left\{0, S_i - S_j - p_{2j}\right\} & \text{if } S_j > t \end{cases} \tag{17}$$

Four states can be considered:

$$\begin{aligned} &State1.\ If\ S_j \le t, S_i \le t \\ &\Rightarrow \begin{cases} I_i^S + I_j^S = \max\left\{0, S_i - t\right\} \\ I_i^{S'} + I_j^{S'} = \max\left\{0, S_i - t - p_{2j}\right\} \end{cases} \Rightarrow I_i^S + I_j^S = I_i^{S'} + I_j^{S'} = 0 \end{aligned}$$

$$\begin{aligned} &State2.\ If\ S_j \le t, S_i > t \\ &\Rightarrow \begin{cases} I_i^S + I_j^S = S_i - t \\ I_i^{S'} + I_j^{S'} = \max\left\{0, S_i - t - p_{2j}\right\} \end{cases} \Rightarrow I_i^S + I_j^S > I_i^{S'} + I_j^{S'} \end{aligned}$$

$$\begin{aligned} &State3.\ If\ S_j > t, S_i \le t \\ &\Rightarrow Infeasible \end{aligned}$$

$$\begin{aligned} &State4.\ If\ S_j > t, S_i > t \\ &\Rightarrow \begin{cases} I_i^S + I_j^S = S_i - t \\ I_i^{S'} + I_j^{S'} = S_j - t + \max\left\{0, S_i - S_j - p_{2j}\right\} \end{cases} \\ &\Rightarrow I_i^S + I_j^S > \max\left\{I_i^{S'} + I_j^{S'}\right\} \Rightarrow I_i^S + I_j^S > I_i^{S'} + I_j^{S'} \end{aligned}$$

The conclusion of these four states demonstrates that if two jobs i and j were $S_i > S_j$, then preceding i by j can decrease the total machine idle times and Makespan.

Lemma 1. *In the optimal solution of the problem under consideration, the sequence of the jobs is the same on both machines.*

Proof. Based on the result of Theorem 1, a job on the second machine has a priority for processing that has an earlier arrival time. On the other hand, jobs that are processed earlier on the first machine and dispatched with earlier batches have earlier arrival times. Therefore, it can be said that the jobs that are processed on the first machine earlier, have a higher priority for processing on the second machine.

Corollary 1. *The jobs dispatched through the same batch to the second machine have the same arrival time. Therefore, according to Theorem 1, their sequence on the second machine has no effect on the optimal solution. As a result, it can be claimed that there is no unique optimal sequence.*

Lemma 2. *In each batch, the sequence of jobs for processing on machine 1 has no effect on optimality.*

Proof. It is clear that in each batch the completion time of all jobs on machine 1 is equal to the completion time of the last job in which the batch is processed, i.e. Makespan. Due to the fact that in a single machine scheduling problem, the sequence of jobs has no effect on Makespan, so changing the sequence of jobs in each batch at the first machine has no effect on the arrival time of the jobs to the second machine and so does not affect the optimality.

Theorem 2. *In the two-machine decentralized flow shop scheduling problem with the inter-factory batch delivery system, for a set of formed batches, the sequences arranged by the Effective Johnson (EJ) rule are optimum with regards to minimizing maximum completion time, with effective processing times p_{1b} and p_{2b} for batch b on machines 1 and 2, respectively: $p_{1b} = \sum_{i \in I_b} p_{1i}$ and $p_{2b} = \sum_{i \in I_b} p_{2i}$, where I_b is a set of jobs in batch b.*

Proof (Through contradiction).

Based on Johnson's rule, the optimal sequence is found in such a way that batches are partitioned into two sets, i.e. Set I and Set II. All batches with $p_{1b} \leq p_{2b}$ assign to set I and other jobs with $p_{2b} < p_{1b}$ assign to set II. Then, the optimal schedule is obtained based on the *SPT(1)-LPT(2)* rule: All batches of Set I are scheduled first in increasing order of p_{1b}(*SPT*). Other batches in Set II are scheduled afterwards in decreasing order of p_{2b}(*LPT*).

Suppose that the sequence *S* (see Fig. 4) is the optimal schedule for the problem in which batch *b1* is followed by *b2* such that one of the following states holds:

State 1. Batch *b1* belongs to Set II and batch *b2* belongs to Set I.

State 2. Batch *b1* and *b2* belong to Set I and $p_{1b1} > p_{1b2}$.

State 3. Batch *b1* and *b2* belong to Set II and $p_{2b1} < p_{2b2}$.

Now, suppose sequence S$^{'}$ where batch *b1* and *b2* are swapped (see Fig. 4). In Fig. 4, batches *l* and *h* are scheduled exactly before and after batches *b1* and *b2*, which remain unchanged. It is clear that the start time of batch *h* on machine 1 will not change. Therefore, it is enough to obtain the start time of this batch (equivalent to the completion time of the previous batch) on machine 2 in sequences *S* and $S^{'}$ and compare them with

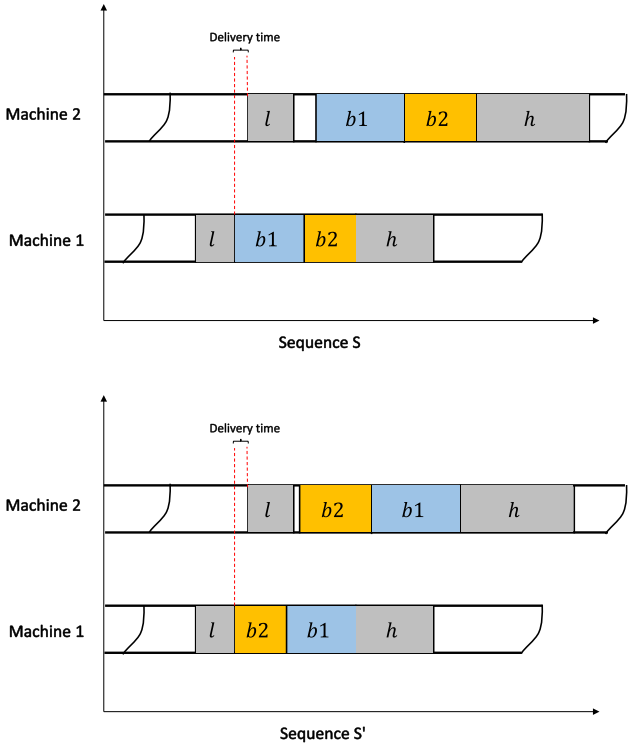


Fig. 4 Sequence of batches on two-machine flow shop environment

each other. C_{mk} and C'_{mk} are completion times of batch k on machine m in sequence S and $S^{'}$, respectively:

$$\begin{aligned} C_{2b2} &= \max\left\{\max\left\{C_{2l}, C_{1l}+p_{1b1}+Dis\right\}+p_{2b1}, C_{1l}+p_{1b1}+p_{1b2}+Dis\right\}+p_{2b2} \\ &= \max\left\{C_{2l}+p_{2b1}+p_{2b2}, C_{1l}+p_{1b1}+Dis+p_{2b1}+p_{2b2}, C_{1l}+p_{1b1}+p_{1b2}+Dis+p_{2b2}\right\} \end{aligned} \tag{18}$$

And similar to the above equation:

$$C'_{2b1} = \max\left\{C_{2l}+p_{2b2}+p_{2b1}, C_{1l}+p_{1b2}+Dis+p_{2b2}+p_{2b1}, C_{1l}+p_{1b2}+p_{1b1}+Dis+p_{2b1}\right\} \tag{19}$$

In state 1: $p_{1b1} > p_{2b1}$ and $p_{1b2} \leq p_{2b2}$

$$\Rightarrow C_{1l}+p_{1b1}+p_{1b2}+Dis+p_{2b2} > C_{1l}+p_{1b2}+Dis+p_{2b2}+p_{2b1}$$

$$\Rightarrow C_{1l}+p_{1b1}+Dis+p_{2b1}+p_{2b2} \geq C_{1l}+p_{1b2}+p_{1b1}+Dis+p_{2b1}$$

$$\Rightarrow C_{2b2} \geq C'_{2b1} \Rightarrow \text{reached a contradiction}$$

In state 2: $p_{1b1} \leq p_{2b1}$, $p_{1b2} \leq p_{2b2}$ and $p_{1b1} > p_{1b2}$

$$\Rightarrow C_{1l} + p_{1b1} + Dis + p_{2b1} + p_{2b2} \geq \begin{cases} C_{1l} + p_{1b2} + Dis + p_{2b2} + p_{2b1} \\ C_{1l} + p_{1b2} + p_{1b1} + Dis + p_{2b1} \end{cases}$$

$$\Rightarrow C_{2b2} \geq C'_{2b1} \Rightarrow \text{reached a contradiction}$$

In state 3: Similar to Case 2 (reversibility property):

$$\Rightarrow C_{2b2} \geq C'_{2b1} \Rightarrow \text{reached a contradiction}$$

Therefore, it was proved that the theorem is true.

Lemma 3. *In a feasible schedule, if there is at least one batch (or single-job-batch) between priorities j to n-1 that Relation (20) is satisfied for it, then that schedule is not optimal. Where* $I_{[i]}^{middle}$ *is the middle idle times of the second machine before batch in priority i and* $p_{1[j]}$ *is the processing time of batch in priority j on the first machine.*

$$\sum_{i=j\ to\ n} I_{[i]}^{middle} + \frac{D}{\beta} > p_{1[j]} for\ j = 2, ..., n-1 \tag{20}$$

Proof. Without losing the generality, it is assumed that there are two batches in priority *k-1* and *k* that Relation (20) is true for batch *[k]*. Now suppose that batch *[k]* is inserted into batch *[k-1]*. It is clear that if the batches of *[k-1]* and *[k]* are combined, the arrival time of batches *[k-1]* will increase by $p_{1[k]}$. Therefore, the completion time of the batches of *[k-1]* and *[k]* on the second machine increases by $p_{1[k]}$. In this case, two states may occur:

State 1. $\sum_{i=k\ to\ n} I_{[i]}^{middle} \geq p_{1[k]} \Rightarrow$ In this case, the start time of batch *[k+1]* on the second machine and thus the value of makespan do not change. Therefore, by inserting the batch *[k]* to *[k-1]*, the number of dispatching batches is reduced by one unit, and so the objective function is reduced.

State 2. $\sum_{i=k\ to\ n} I_{[i]}^{middle} < p_{1[k]} \Rightarrow$ In this case, the start time of batch *[k+1]* on the second machine and thus the value of makespan increase by $p_{1[k]} - \sum_{i=k\ to\ n} I_{[i]}^{middle}$. By inserting the batch *[k]* to *[k-1]*, the number of dispatching batches is reduced by one unit, and thus the objective function is reduced. According to Relation (20), the increase in makespan costs is less than the cost of dispatching a batch $(D > \beta\left(p_{1[k]} - \sum_{i=k\ to\ n} I_{[i]}^{middle}\right))$, and therefore the objective function will eventually decrease.

Lemma 4. *In a partial sequence, where some batches have been formed and no decision has been made yet on batching the remaining 'un-batched' jobs, the lower bound in the makespan is equal to the makespan obtained in a sequence formed by remarking each un-batched job as a single-job-batch and scheduling all batches with EJ rule.*

Proof. Forming any batch with more than one job from un-sequenced jobs certainly increases the arrival time of some jobs and can therefore affect the makespan. Hence, by considering un-sequenced jobs as single-job-batch this negative impact is prevented. Consequently, the schedule obtained from the EJ rule guarantees that this sequence has the lowest makespan, according to Theorem 2.

5.2 Branch and bound procedure

Based on what was specified in Sect. 4, three decisions must be made in this matter: (1) scheduling the processing of jobs on the first machine (2) how to batch the jobs processed on the first machine (3) scheduling the processing of jobs on the second machine. According to lemma 1, lemma 2 and theorem 2, it can be claimed that decisions 1 and 3 can be answered if the batching scheme is specified. Because first of all, it was proved that the sequence of jobs is the same on both machines. Second, the sequence of jobs in each batch has no effect on optimality, and third, the optimal batch sequence will be achieved according to the EJ rule. Therefore, in this section, a fast branch and bound (B&B) algorithm is presented to determine the batching scheme and obtain the optimal solution of the problems. This algorithm has a binary structure that uses a depth-first search strategy. At each stage, it is checked whether a job forms a batch with another job (or batch) or not.

At root, all jobs are considered as single-job-batch, and their optimal sequence is obtained based on the EJ rule. The first job in this sequence is considered as a *basic batch,* and the next job is considered as a *transition job*. At each stage, two children are generated from the parent node. One examines the insertion of the *transition job* to the *basic batch* (merge node) and the other examines the non-batching of the two jobs together (Non-merge node). After each merge node, the basic batch is updated, which includes the previous basic batch plus the transition job. In this case, the job after the previous transition job is considered as a new transition job and this process continues until it reaches the leaf node. After each None-merge node, only the transition job is updated, which is equivalent to the next job of the previous transition job. This process continues until there is no job to be considered as a transition job. After that, the basic batch must be changed, which is equivalent to a single-job batch following the last one. The continuation of the process will be the same as the previous description.

In this tree, the *NB* parameter is defined to calculate the minimum batch needed to dispatch jobs in each node. This parameter is equal to 1 at the root, and for each basic batch that has changed, one unit is added only for the first related non-merge node ($NB = NB + 1$). The pseudocode related to the branching and production of children in this algorithm is as follows:

Procedure 1: Branching operation and the generation of new nodes

Step 1. *(Initial setting)*
- Sort jobs according to EJ rule in root node,
- Set NB=1
- consider the first job in the sequence as the basic batch and select the job in the next position, as transition job (j)
- partition root into two child nodes
Step 2. *(Generate child nodes)*
Repeat
- Checking whether transition job adds to the basic batch (merge node) or not (non -merge node)
- j=j+1
*-**If** the none-merge node is the first one for a basic batch **Then** NB= NB + 1*
- partition each node into two child nodes
***Until** no job remains to be compared with the basic batch*
Step 3. *(Changing the basic batch)*
***If** the next job after old basic batch is not the last job in the sequence*
***then** consider this job as new basic batch, and the next single -job-batch as transition job and go to step 2*
Step 4. *End.*

Procedure 1 Branching operation and the generation of new nodes

Note that the above method of generating a tree guarantees that it will generate all possible feasible solutions in the most pessimistic state. Therefore, it is claimed that the proposed branch and bound algorithm can reach the optimal solution.

5.3 Heuristic upper bound based Decreasing Neighborhood Search (HDNS)

For complex problems, one of the best ways to get good solutions is to use heuristic methods. Heuristic methods are generally able to obtain near-global optimum solutions in a very short time. Heuristic methods can also be used in the structure of branch and bound algorithms in order to prune more nodes. It is necessary to use a heuristic method in the body of the branch and bound algorithm to obtain an appropriate and acceptable initial solution. The methods based on neighborhood search are one of the most agile methods to find the appropriate upper bound in the B&B algorithm. Their advantage over meta-heuristic algorithms is their high speed of operation along with appropriate accuracy (Mazdeh et al. 2013).

The main contribution in the field of solution method in the current research is to present the exact B&B method to obtain the global optimal solution. The heuristic method is an auxiliary tool to speed up the B&B algorithm. Although meta-heuristic methods such as genetic algorithm are used in some researches in the body of B&B, it should be noted that the use of meta-heuristic methods in the body of the B&B method is generally a hybridization with this method (Ozkan et al. 2020). In this way, the power of the method in creating feasible solutions or improving the resulting solutions will increase by integrating the B&B method with the meta-heuristic method.

Therefore, the significant aspect of providing the HDNS method is not to provide a new solution method but to help the newly developed B&B method to speed up reaching the optimal global solution. However, the solution is not guaranteed by heuristic and meta-heuristic methods such as the genetic algorithm.

As stated in subSect. 5.2 if there is a batching scheme, an optimal schedule can be generated. The heuristic method developed in this section has two phases. In the first phase, a local optimum solution is created. In the second phase, some improvements are made to the initial solution with the help of lemma 3 and a variant of the neighborhood search method so that we can move towards optimization.

In Phase I, jobs are first sorted according to the EJ rule. Then the first job is selected as the *basic batch* and the next job is selected as the *transition job* (j). The insertion process then begins, during which the eligibility of inserting the transition job into the basic batch is examined in terms of the objective function. If this change improves the objective function, it will be fixed; otherwise, the next job will be selected as the new transition job and the above process will be repeated until the checking last job. After that, the basic batch is changed and the next job is then selected as the new basic batch and the above process is repeated again to finally reach a good feasible solution.

In phase II, the first improvement occurs with the help of lemma 3 results. If we represent the total number of batches generated in the phase I solution with K, then the counter θ is defined as $\theta = K - 1$. For the batch in θ, if relation (20) is true, then this batch merges with the batch in priority *K-2*. In this case, the value of K and the objective function are updated and $\theta = K - 1$ again. But if Eq. (20) is not true, $\theta = \theta - 1$ and the above process is repeated until $\theta = 1$ is reached.

The second improvement occurs using the Decreasing Neighborhood Search (DNS) method developed by Mazdeh et al. (2013) for a scheduling problem. One of the disadvantages of the initial solution created in phase I is that it generally generates more batch than the optimal solution. This is because the search process is based on very close neighbors, and jobs at a greater distance are ignored. To do this, the output of the first improvement with K batches is considered as the input of the second improvement. First K neighbors are created. The neighborhood structure is such that two batches are randomly selected and merged with each other. After generating the neighbors their objective functions are calculated and if the input solution improves, the upper bound value is replaced with it and the above process continues until the stop condition is reached. The stop condition is when either the generated neighborhoods do not improve the input solution or $K=1$. Pseudocode related to the HDNS method is as follows:

Procedure 2: Heuristic method based Decreasing Neighborhood Search (HDNS)

Phase I:

Step1. *Initial sequence (S):*
- Sort jobs (single-job- batches) by EJ rule
- Set job 1 as basic batch (bb)
Step2. *Set the job after basic batch as transition job (j=bb+1);*
If *j>n go to step 6*
Step3. *Let Ś be the new sequence by inserting transition job in basic batch*
Step4. ***If*** *the objective function has decreased*
then *S =Ś, else set j=j+1*
Step5. ***If*** *j ≤ n go to step* 3
Step6. ***If*** *bb< n* ***then*** *set bb=bb+1 and go to step2*
Step7. *Set K=number of batches and end of Phase I.*

Phase II:

Improvemnet 1:
Step1.
- Set initial sequence (S) from Phase I
- Set Θ=K-1
Step2. *While Θ>1 do*
If *Relation (20) is true for batch in priority Θ*
Then *merge this batch with the batch in K-2 priority , update sequence S, objective function and K*
Set Θ=K-1
Else *Θ=Θ1*
End of while.
Step3. *Set K=number of batches and end of Improvement 1.*
Improvemnet 2:
Step 1. *Set initial sequence (S) from Improvement 1,*
Step 2. ***If*** *K=1 go to step 5,*
Step 3. *Generate K neighbors with special process,*
Step 4. *Evaluate objective functions,*
If *the objective function of S has decreased,*
Then *update sequence S and K. Go to step 2,*
Otherwise *go to step 5,*
Step 5. *End of Improvement 2.*
End of Phase II

5.4 Lower bound

The performance of the branch and bound algorithm is highly dependent on the tightness of the lower bound at each node. The tighter lower bound, the higher the pruning rate of the nodes and the lower running time. In each node, according to lemma 4, the minimum value of makespan can be easily calculated using the EJ rule, so that all jobs that have not yet been evaluated are considered as single-job-batch. It should be noted that the makespan lower bound is updated only in the merge nodes, where two separated jobs are batched together. In other nodes, the makespan lower bound is inherited from the father.

Also, in order to calculate the lower bound for batch delivery costs, it is assumed in the root that in the best case, all jobs will be dispatched with one batch ($NB=1$). For other nodes, based on the NB calculated in each node, the minimum number of dispatching batches is specified and the minimum value can be calculated based on Eq. (21):

$$LB_{node} = \beta \times Makespan^{L}_{node} + NB_{node} \times D \tag{21}$$

where $Makespan^{L}_{node}$ represents the lower value of makespan in that node and D is the cost of dispatching a batch.

5.5 Fathoming and backtracking

A node is fathomed if:

1. It is a leaf node, meaning all the jobs in the sequence are considered as transition jobs in one of the father's nodes.
2. The related makespan lower bound for the node is equal to or greater than the upper bound.

When a node is fathomed, backtracking happens. If there are no nodes that should be visited in the algorithm, then the branching procedure is interrupted and the current upper bound would be considered as the optimal solution.

6 Computational results

This section tries to examine the validity of the proposed solution methods with various evaluations on a number of instances. First, to better understand the branch and bound procedure, a small example is solved step by step. Then, a number of instances in different dimensions are generated based on benchmark problems in the literature, and with the help of it, solution methods are evaluated. The MILP model is coded in GAMS software and solved by CPLEX in a computer with a CPU of Intel Core i7 1.9 GHz and 8 GB of RAM. Also, the proposed branch and bound algorithm and HDNS method are coded by C# and are run on the same computer.

6.1 Illustrated example

In this section, in order to better describe the branching method as well as the calculations related to the upper and lower bounds in each node, a small example is solved step by step. In this example, there are three jobs that must be processed on two machines in a series structure. The distance between the two machines is equal to 6 h and the cost of dispatching each batch between the two machines is equal to $ 300. Moreover, here $\beta=70$. Information related to processing times on both machines for these three jobs is shown in Table 2.

First, the above problem is solved using the HDNS method to obtain upper bound for starting the B&B algorithm:

Table 2 Processing time information related to the illustrated example

Jobs	Processing time on machine 1	Processing time on machine 2
1	2	6
2	5	8
3	5	4

In phase I, the sequence S is first obtained based on the EJ rule, which will be $1 \rightarrow 2 \rightarrow 3$. Therefore, job 1 is considered as a basic batch and job 2 is considered as a transition job ($j=2$). In this section, the value of the objective function is denoted by $f(S)$. The value of the objective function of sequence S will be equal to the cost of makespan plus the cost of dispatching three batches, i.e. $f(S) = (70 \times 26) + (3 \times 300) = 2720$.

Then we generate the sequence S' by inserting job 2 in the single-job-batch 1. With this change, the total delivery costs are reduced by one batch cost, but on the other hand, the makespan cost is equal to 2170. Therefore, the value of the objective function of the new sequence is equal to $f(S') = (70 \times 31) + (2 \times 300) = 2770$, and as a result, this change is not fixed and the S sequence does not change.

According to the previous step, $j=2+1=3$, and since $j \leq 3$, then the sequence S' is generated again by inserting job 3 in the single-job-batch 1. With this change, the total delivery costs will be reduced by one batch cost, but on the other hand, the makespan cost will be equal to 2030. Therefore, the value of the objective function of the new sequence is equal to $f(S') = (70 \times 29) + (2 \times 300) = 2630$, and as a result, this change is fixed and $S=S'$.

Now, inserting job 2 into the new basic batch is considered. With this change, the total delivery costs are reduced by one batch cost, but on the other hand, the makespan cost is equal to 2520. Therefore, the value of the objective function of the new sequence is equal to $f(S') = (70 \times 36) + (1 \times 300) = 2820$, and as a result, this change is not fixed and the S sequence does not change.

According to the previous step, $j=3+1=4$ and $j > 3$, so $bb=1+1=2$, and $j=bb+1=3$. Here we have only 2 batches, i.e. *(1,3), 2* and the end of the algorithm will be reached in phase I.

In phase II, it is not possible to run improvement 1 because $\theta = 1$. For the second improvement, given that $K=2$, two neighbors must be generated. To generate each neighbor, two batches must be selected randomly. But because there are two batches in total, only one neighbor in the form *(1, 2, 3)* can be generated. The objective function of this neighborhood is equal to 2820 and therefore no improvement is achieved. Therefore, the final solution of phase I is determined as the output of the HDNS method. Figure 5 shows the scheduling of the HDNS solution.

Now we enter the process of branch and bound algorithm. Figure 6 shows a fully formed tree related to this example. Nodes are numbered according to their priority. The jobs enclosed by () and [] are assigned to a batch. Those enclosed with () can add more jobs to them in the continuation of the tree and are so-called open batch. But those enclosed with [] are closed batches and it is not possible to add any job to them in

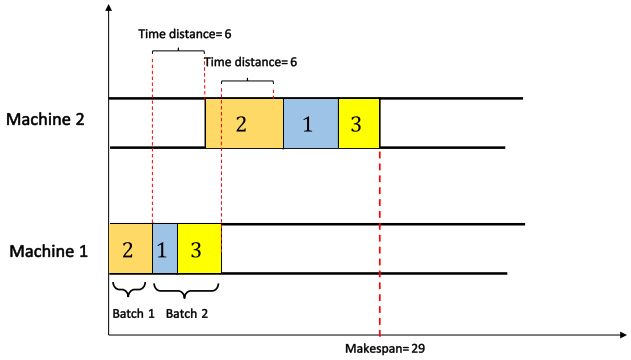


Fig. 5 Jobs schedule related to HDNS

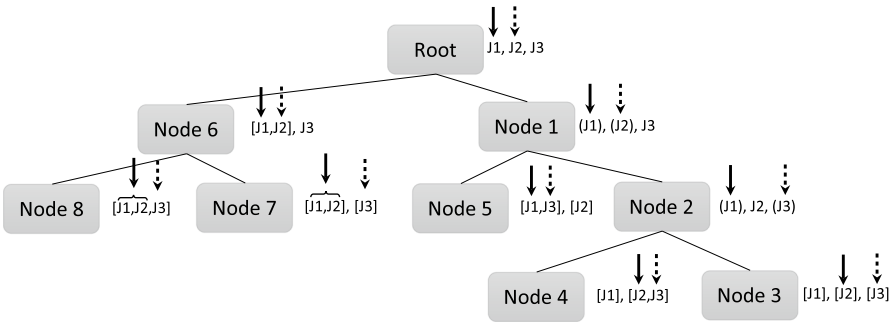


Fig. 6 The fully formed B&B tree corresponds to the illustrated example

his children. Jobs (or batches) shown with a black arrow are basic batches. Square dot black arrows are also related to transition jobs.

The calculations in each node are as follows:

$$Root: NB_{Root} = 1, UB = 2630$$
$$LB_{root} = (70 \times 26) + (1 \times 300) = 2120 < 2630; \therefore movedown.$$

$$Node1: 1st\ non - merge\ node, NB_{Node1} = NB_{Root} + 1 = 2, UB = 2630$$
$$LB_{Node1} = (70 \times 26) + (2 \times 300) = 2420 < 2630; \therefore movedown$$

$$Node2: 2nd\ non - merge\ node, NB_{Node2} = NB_{Node1} = 2, UB = 2630$$
$$LB_{Node2} = (70 \times 26) + (2 \times 300) = 2420 < 2630; \therefore movedown$$

$$Node3: Leaf\ node, NB_{Node3} = NB_{Node2} + 1 = 3, UB = 2630$$
$$LB_{Node3} = (70 \times 26) + (3 \times 300) = 2720 \not< 2630; \therefore backtracking.$$

$$Node4 : Leaf\ node,\ NB_{Node4} = NB_{Node2} = 2,\ UB = 2630$$
$$LB_{Node4} = (70 \times 30) + (2 \times 300) = 2700 \not< 2630; \therefore backtracking.$$

$$Node5 : Leaf\ node,\ NB_{Node5} = NB_{Node1} = 2,\ UB = 2630$$
$$LB_{Node5} = (70 \times 29) + (2 \times 300) = 2630 \not< 2630; \therefore backtracking.$$

$$Node6 : Merge\ node,\ NB_{Node6} = NB_{Root} = 1,\ UB = 2630$$
$$LB_{Node6} = (70 \times 31) + (1 \times 300) = 2470 < 2630; \therefore movedown.$$

$$Node7 : Leaf\ node,\ NB_{Node7} = NB_{Node6} + 1 = 2,\ UB = 2630$$
$$LB_{Node7} = (70 \times 31) + (2 \times 300) = 2770 \not< 2630; \therefore backtracking.$$

$$Node8 : Leafnode, NB_{Node8} = NB_{Node6} = 1, UB = 2630$$
$$LB_{Node7} = (70 \times 36) + (1 \times 300) = 2820 \not< 2630; \therefore backtracking$$

End of B&B algorithm and return current UB as optimal solution

According to the result of the branch and bound algorithm, the solution obtained by HDNS is the global optimum solution. It should be noted that in order to demonstrate different calculations in this algorithm, an example was solved in which all nodes are formed in the tree.

6.2 Experimental results

In this section, in order to evaluate the proposed solution methods, inspired by Taillard (1993) benchmark Flow shop problems, a number of random instances in different sizes are generated and their solving results are reported by all solution methods. To generate these instances, the processing times on the machines are aided by benchmarking problems. Given that the benchmark problems are for flow shops with 5, 10, and 20 machines, the processing times for the first two machines of the cases are considered as input. The value of *Dis* follows the uniform distribution function $\left[0.9 \times \frac{\sum_{i\in I} p_{1i} + \sum_{i\in I} p_{2i}}{2n}, 1.1 \times \frac{\sum_{i\in I} p_{1i} + \sum_{i\in I} p_{2i}}{2n}\right]$, i.e., choosing this parameter in the range close to the average processing times. The value of β is considered to be equal to 70. It should be noted that the reason for choosing the above values for the parameters is that a proper balance should be created between the two parts of the objective function, i.e. the cost of makespan and the delivery cost. The imbalance between the two parts may cause the results and analyses to be invalid. Finally, the value of D is considered equal to $\alpha \times Dis$, where α is the input parameter for generating the problem, which is determined in experiments. In fact, the value of α determines the amount of attention paid to delivery costs compared to the cost of makespan. The larger this parameter, the more significant the delivery costs and therefore the greater the tendency to batch jobs together.

In order to evaluate, 360 random instances in three groups of small, medium and large sizes have been solved by the methods, and their results are shown in Tables 3, 5. In each setting, 10 instances are solved and related information is reported. In this study, a reasonable running time to solve instances is considered 3600 s. If a method does not reach the final solution by this time, the last solution obtained will be considered as output. Given that the HDNS algorithm is a heuristic method, it does not guarantee global optimization. On the other hand, MILP may have reached the running time limitation and its output may not necessarily be a global optimum solution, especially for large instances. For this reason, Eq. (22) is used to calculate the Relative error:

$$Relativeerror = \frac{1}{10} \times \sum_{j=1\ to\ 10} \frac{UB_j - OPT_j}{OPT_j} \tag{22}$$

In relation (22), UB_j is the solution obtained by MILP or HDNS and OPT_j is the global optimum solution obtained by B&B. Table 3 illustrates the results of evaluating small-size instances. This table shows information about the CPU running times of different instances solving by MILP, B&B and HDNS methods as well as the relative error rate. This table also specifies the number of instances that have been interrupted by the MILP when the running time limit (reslim) or optimization condition (optcr) is reached.

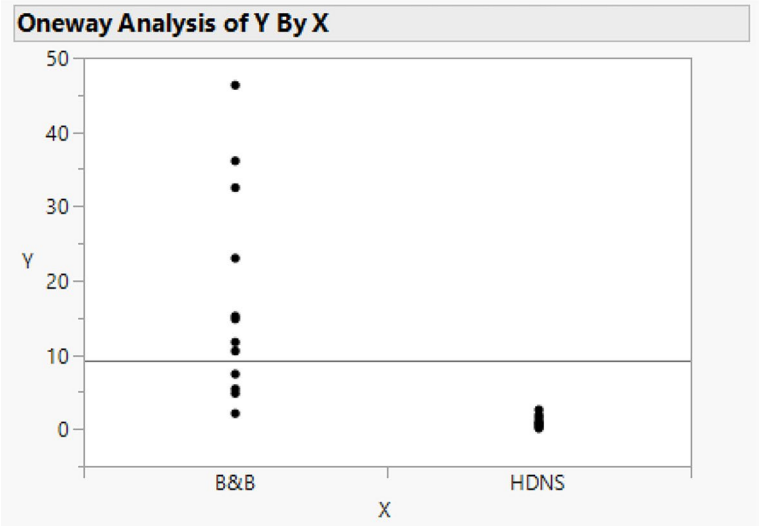
In this section, in order to compare the average CPU running time of algorithms, statistical tests are used. Wilcoxon signed-rank test is a non-parametric test, an alternative for ANOVA, where the normality assumption is not required. Wilcoxon signed-rank test is used to compare the locations of two populations using two matched samples. Here the null hypothesis is that the CPU running time of each pairwise method is equal. Figures 7, 8, 9 show the results of statistical tests related to the comparison of algorithms for small-size instances.

The p-value is lower than the alpha value of 0.05; therefore, the null hypothesis is rejected. As it turns out, the HDNS algorithm is the fastest method to get the final solution, followed by the B&B and MILP algorithms, respectively. The CPU running time in MILP increases dramatically through increasing the number of jobs so that for most instances with 15 jobs and all instances with 20 jobs, the commercial solver stops the process due to reaching the maximum running time. However, the results of the relative error show that most of the instances in which MILP is interrupted by reslim have reached the global optimum solution. Therefore, it can be concluded that CPLEX was not able to terminate the process with the optcr condition due to a weak lower bound. Another result of Table 3 is that by increasing α, the running time decreases. The reason for this is that with the increase in batch delivery costs, the importance of batching has increased and jobs are more inclined to batch together. Therefore, the search space is pruned quickly and the optimal solution can be reached with less searching. Figure 10 shows CPU running times for solving small-size instances based on different values of α.

Tables 4 and 5 show the results for medium- and large-size instances. These tables indicate information about B&B and HDNS running times as well as the relative error of the heuristic method. Figures 11, 12 show the results of statistical tests

Table 3 Performance of solving methods for small-size instances

Setting	#Jobs	Alpha	MILP CPU time (Sec.)			B&B CPU time (Sec.)			HDNS CPU time (Sec.)			MILP interruption type		Avg. relative error	
			Avg	Min	Max	Avg	Min	Max	Avg	Min	Max	#reslim	#optcr	MILP	HDNS
S1		40	31	17	84	7.4	2.9	9.1	0.2	0.1	0.3	0	10	0.000	0.007
S2	5	60	24	11	93	4.8	1.8	6.5	0.2	0.1	0.4	0	10	0.000	0.009
S3		80	21	8	67	2.1	0.7	3.2	0.1	0.0	0.2	0	10	0.000	0.014
S4		40	829	492	1913	15.2	6.9	24.1	0.8	0.3	1.6	0	10	0.000	0.019
S5	10	60	801	455	2028	11.7	5.7	22.3	0.6	0.2	1.2	0	10	0.000	0.018
S6		80	685	431	1575	5.4	2.1	7.9	0.3	0.2	0.5	0	10	0.000	0.027
S7		40	3600	3600	3600	36.1	27.1	44.3	1.5	0.6	2.2	10	0	0.000	0.023
S8	15	60	3566	3255	3600	23.0	18.6	30.3	1.1	0.4	1.4	9	1	0.000	0.026
S9		80	3306	2419	3600	10.5	5.7	18.2	0.6	0.3	1.1	6	4	0.000	0.030
S10		40	3600	3600	3600	46.3	33.5	54.0	2.6	1.1	3.6	10	0	0.014	0.027
S11	20	60	3600	3600	3600	32.5	25.6	35.8	1.9	1.3	2.8	10	0	0.011	0.029
S12		80	3600	3600	3600	14.8	9.4	17.2	1.0	0.6	1.6	10	0	0.000	0.035



Wilcoxon / Kruskal-Wallis Tests (Rank Sums)

Level	Count	Score Sum	Expected Score	Score Mean	(Mean-Mean0)/Std0
B&B	12	221.000	150.000	18.4167	4.072
HDNS	12	79.000	150.000	6.5833	-4.072

2-Sample Test, Normal Approximation

S	Z	Prob>|Z|
79	-4.07209	<.0001*

1-Way Test, ChiSquare Approximation

ChiSquare	DF	Prob>ChiSq
16.8180	1	<.0001*

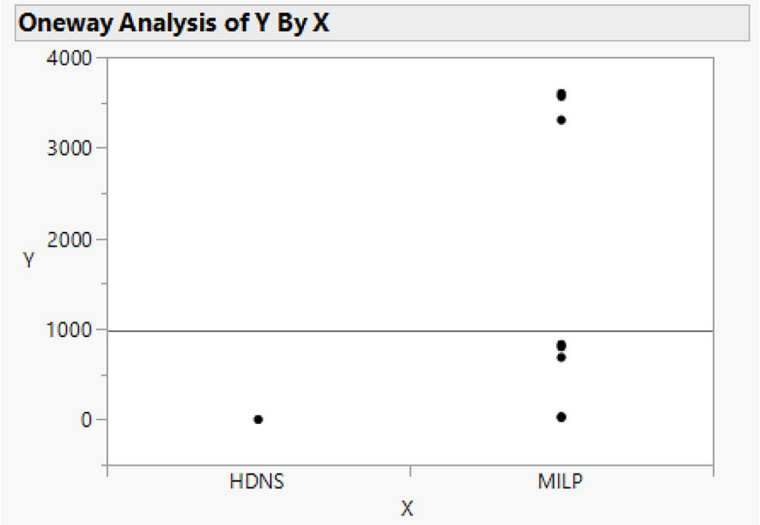
Fig. 7 HDNS via B&B for small-size instances

related to the comparison of B&B and HDNS for medium and large-size instances, respectively.

The p-value is lower than the alpha value of 0.05; therefore, the null hypothesis is rejected. As it turns out, the HDNS algorithm, despite its excellent performance in terms of speed (see Fig. 13), does not guarantee global optimization, and as the size of the instance increases, the error rate also increases (see Fig. 14). The average HDNS error rate for the largest instances is 12.4%.

As shown in Fig. 13, by increasing α, the CPU running time decreases. The reason for this is that with the increase in batch delivery costs, the importance of batching has increased and jobs are more inclined to batch together. Therefore, the search space is pruned quickly and the optimal solution can be reached with less searching.

Moreover, the results of Fig. 15 show that the speed of the HDNS method is much higher than B&B so that it solves the largest instances in less than 25 s.



Wilcoxon / Kruskal-Wallis Tests (Rank Sums)

Level	Count	Score Sum	Expected Score	Score Mean	(Mean-Mean0)/Std0
HDNS	12	78.000	150.000	6.5000	-4.139
MILP	12	222.000	150.000	18.5000	4.139

2-Sample Test, Normal Approximation

S	Z	Prob>|Z|
222	4.13887	<.0001*

1-Way Test, ChiSquare Approximation

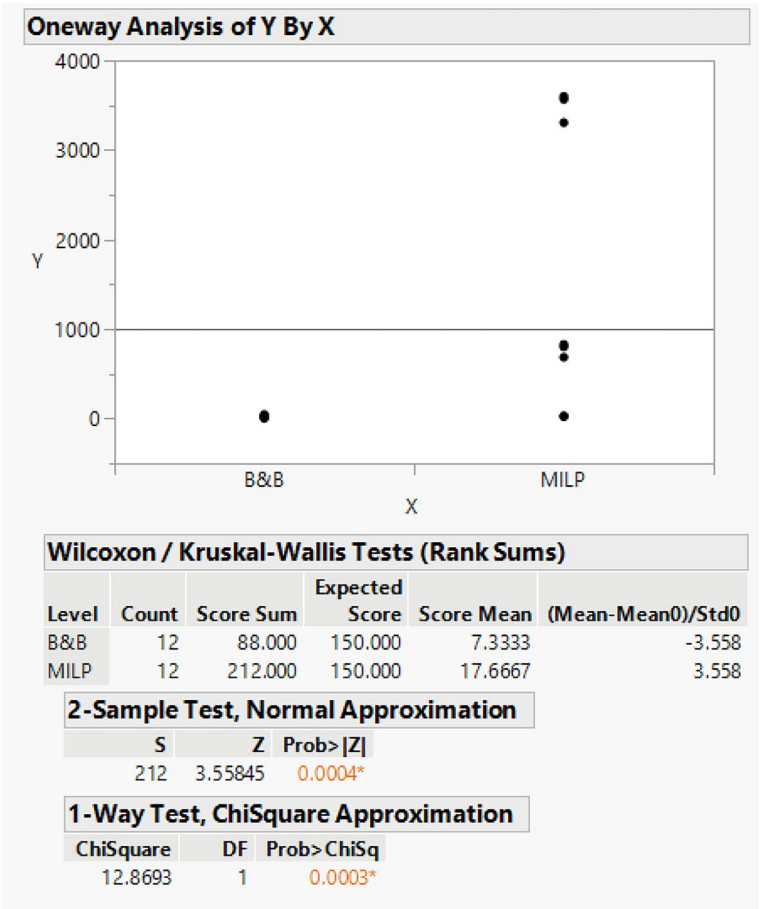
ChiSquare	DF	Prob>ChiSq
17.3706	1	<.0001*

Fig. 8 HDNS via MILP for small-size instances

6.3 Case study

In the current study, Pars Investment Casting Company (PICC) is considered as a case study. PICC as one of the subsidiary companies of Oil Turbo Compressor Companies (OTC) and the new owner of Noor Experience Turbine Parts Company produces all kinds of hot gas turbines. The factory is located in Shahroud, Iran. For the production of its parts, this company has two stages of casting and machining in two different cities.

Figure 16 represents the process of production and transportation of semi-finished parts between two production stages. All the information presented in this section has been extracted in real terms.



Wilcoxon / Kruskal-Wallis Tests (Rank Sums)

Level	Count	Score Sum	Expected Score	Score Mean	(Mean-Mean0)/Std0
B&B	12	88.000	150.000	7.3333	-3.558
MILP	12	212.000	150.000	17.6667	3.558

2-Sample Test, Normal Approximation

S	Z	Prob>|Z|
212	3.55845	0.0004*

1-Way Test, ChiSquare Approximation

ChiSquare	DF	Prob>ChiSq
12.8693	1	0.0003*

Fig. 9 B&B via MILP for small-size instances

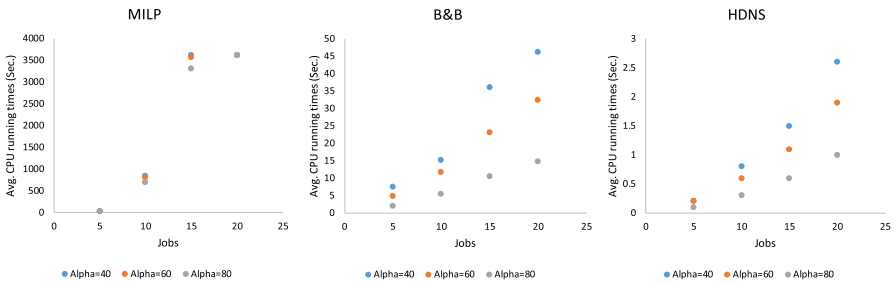


Fig. 10 Avg. CPU running times for small-size instances

The time interval between the two cities is 3 h, which is significant compared to the processing time of the production stages. Table 6 represents the processing time of parts in the two stages of machining and casting.

Table 4 Performance of solving methods for medium-size instances

Setting	#Jobs	Alpha	B&B CPU time (Sec.)			HDNS CPU time (Sec.)			Avg. HDNS Relative error
			Avg	Min	Max	Avg	Min	Max	
S13	30	40	73.9	51.7	90.8	3.7	3.3	4.0	0.031
S14		60	59.1	44.2	72.7	3.2	2.9	3.9	0.035
S15		80	27.6	20.5	35.1	1.5	1.2	1.9	0.036
S16	40	40	118.8	92.0	130.9	6.6	5.1	7.4	0.036
S17		60	84.7	69.8	95.1	5.4	4.2	6.0	0.040
S18		80	40.9	33.5	45.3	2.7	2.3	3.2	0.040
S19	50	40	185.4	150.4	196.3	8.0	6.8	9.2	0.038
S20		60	119.5	107.6	134.3	7.3	6.4	7.8	0.043
S21		80	56.2	51.3	64.0	3.8	3.4	4.5	0.048
S22	60	40	214.5	199.6	220.7	12.2	11.3	13.0	0.043
S23		60	149.9	138.4	154.5	9.8	9.2	10.7	0.052
S24		80	72.4	64.1	77.0	5.0	4.3	6.2	0.055

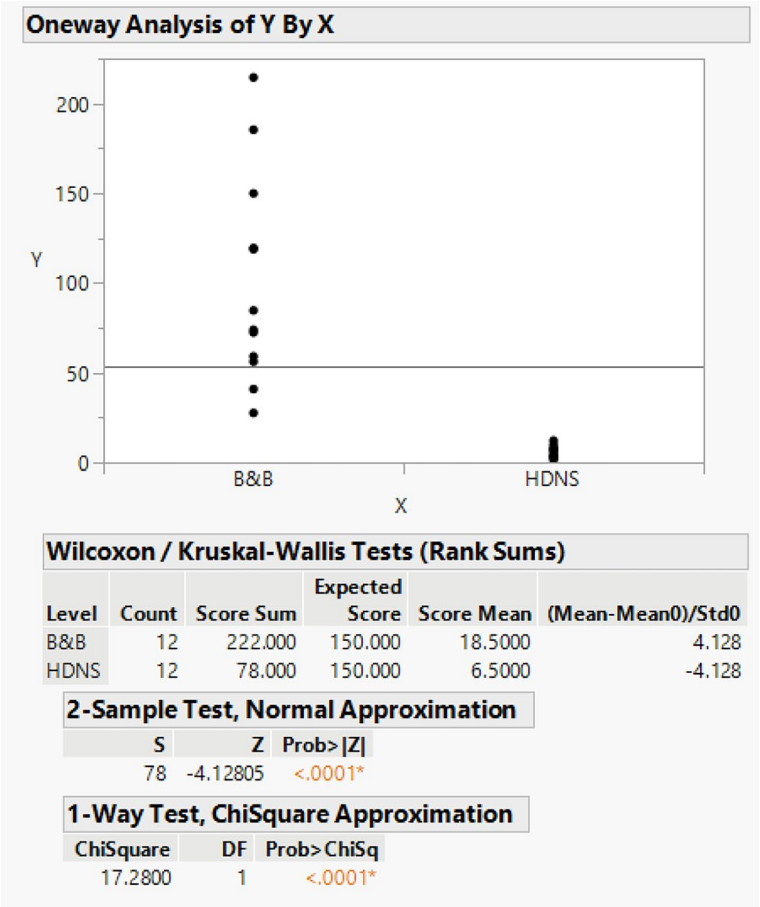
Table 5 Performance of solving methods for large-size instances

Setting	#Jobs	Alpha	B&B CPU time (Sec.)			HDNS CPU time (Sec.)			Avg. HDNS Relative error
			Avg	Min	Max	Avg	Min	Max	
S25	70	40	272.3	258.4	280.6	14.6	12.7	15.9	0.067
S26		60	180.2	171.4	193.2	12.1	10.9	13.5	0.069
S27		80	83.7	72.5	88.5	5.7	5.1	6.4	0.068
S28	80	40	298.6	269.4	313.8	19.4	17.6	21.0	0.070
S29		60	223.0	211.0	232.4	15.9	14.5	17.4	0.079
S30		80	108.6	100.6	112.1	8.1	7.5	8.8	0.084
S31	90	40	359.4	348.2	366.8	25.2	24.2	26.5	0.093
S32		60	275.4	265.4	282.0	21.4	19.3	23.0	0.101
S33		80	133.0	119.2	136.4	10.8	9.7	11.9	0.106
S34	100	40	446.5	427.3	455.7	30.9	28.3	33.8	0.099
S35		60	341.8	328.2	350.6	26.8	24.8	28.1	0.112
S36		80	154.2	145.9	160.2	14.5	13.6	15.4	0.124

The transportation cost is \$80 per shipment and the value of time to cost coefficient (β) is considered to be \$70. After solving the presented model, the values of positive variables C_{jk}, Cb_k, S_i and CT_j have been obtained and shown in Tables 7, 8, 9, 10 respectively.

The following binary variables have the value of 1:

$x_{1,2,4} = 1$, $x_{2,1,1} = 1$, $x_{3,2,3} = 1$,$x_{4,3,1} = 1$, $x_{1,4} = 1$, $x_{2,2} = 1$, $x_{3,3} = 1$, $x_{4,1} = 1$, $y_1 = 1$, $y_3 = 1$ and $y_4 = 1$.



Wilcoxon / Kruskal-Wallis Tests (Rank Sums)

Level	Count	Score Sum	Expected Score	Score Mean	(Mean-Mean0)/Std0
B&B	12	222.000	150.000	18.5000	4.128
HDNS	12	78.000	150.000	6.5000	-4.128

2-Sample Test, Normal Approximation

S	Z	Prob>|Z|
78	-4.12805	<.0001*

1-Way Test, ChiSquare Approximation

ChiSquare	DF	Prob>ChiSq
17.2800	1	<.0001*

Fig. 11 B&B via HDNS for medium-size instances

All relations between variables and parameters are logical. Finally, values of 64 and 4720 are obtained for the maximum completion time of jobs on machine 2 and the objective function value (total cost), respectively.

7 Conclusion

The two-machine decentralized flow shop scheduling problem with an inter-factory batch delivery system was introduced for the first time in this study. This issue affects production systems with distributed production environments and significant distances between their plants. First of all, a linear mathematical programming model capable of achieving global optimum solutions for small instances was proposed to better introduce the problem. The goal was to reduce the total makespan and inter-factory batch delivery costs. The problem's

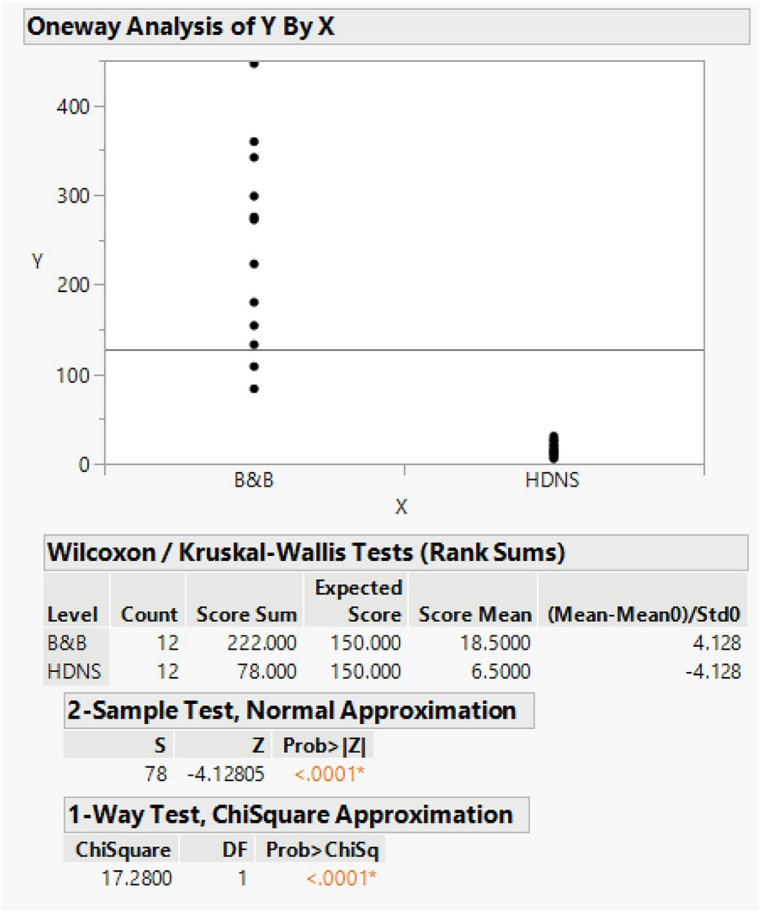


Fig. 12 B&B via HDNS for large-size instances

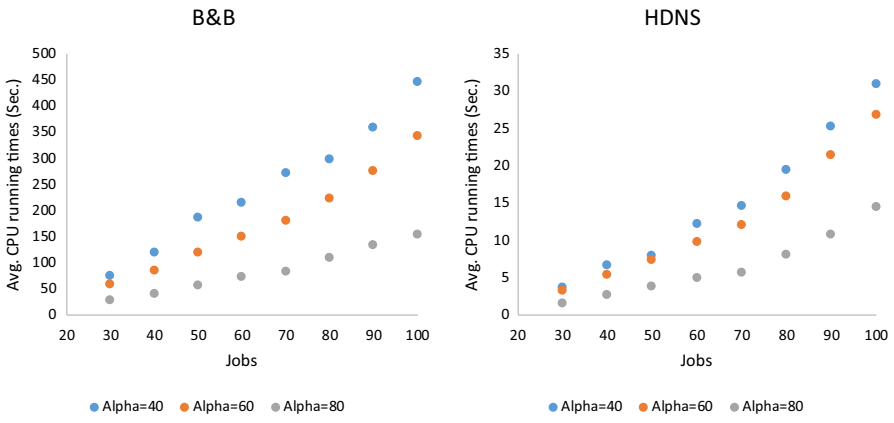


Fig. 13 Avg. CPU running times for medium- and large-size instances

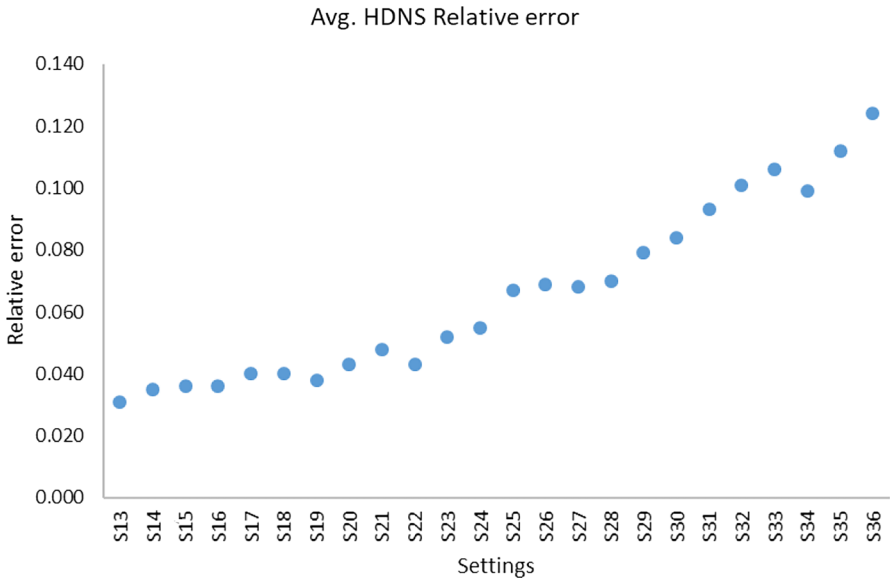


Fig. 14 Avg. HDNS relative errors for medium and large size instances

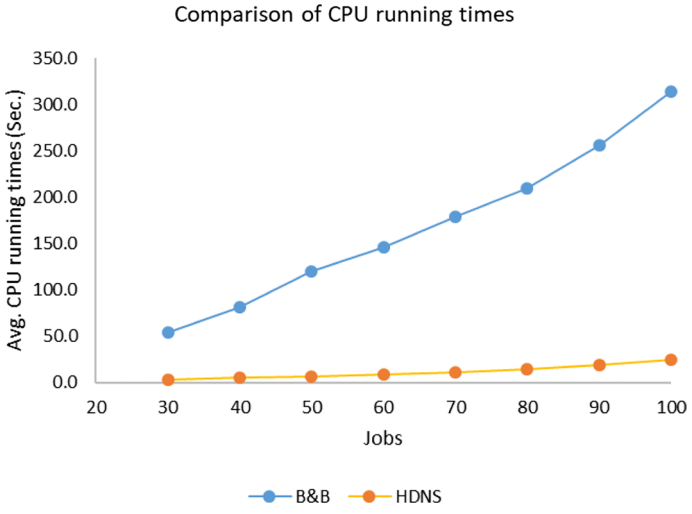


Fig. 15 Performance of methods based on CPU running time

characteristics were then examined, and several theorems and lemmas were presented to aid in the problem-solving process. These features were used as a basis for developing a fast branch and bound algorithm that can solve large instances in a reasonable amount of time. For proper tree pruning, this algorithm was outfitted with a heuristic upper bound and a tight lower bound.

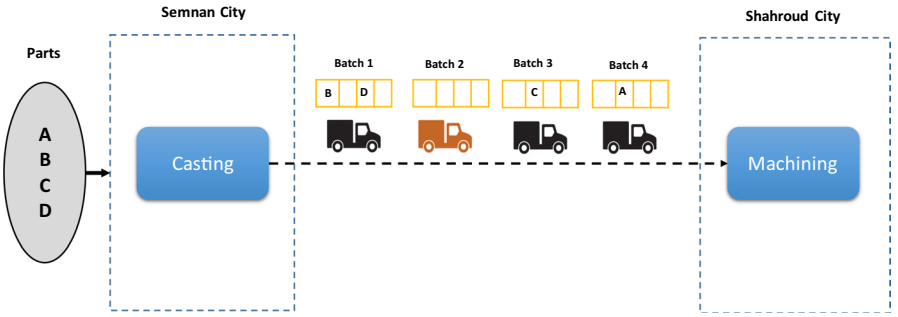


Fig. 16 Production process and transportation of parts between production stages in the PICC company

Table 6 Processing time of parts in the two stages

Duration (Hour)			Part Name (job *i*)
Total	Machining	Casting	
28	5	23	A
31	12	19	B
16	8	8	C
14	8	6	D

Table 7 Results of the completion time related to positions of batches

C_{jk}		k			
		1	2	3	4
j	1	19	25	25	33
	2	19	25	33	56
	3	25	25	33	56
	4	25	25	33	56

Table 8 Results of the completion time related to batches

Cb_k	k			
	1	2	3	4
	25	25	33	56

Table 9 Results of the arrival time of jobs to machine 2

S_i	i			
	1	2	3	4
	59	28	36	28

Table 10 Results of the completion time of job assigned to positions on machine 2

CT_j	j			
	1	2	3	4
	39	51	59	64

To make the B&B process more transparent, a simple example was solved step by step in the computational results section. Then, 360 instances were generated at random from benchmark problems in the literature, and the results of their solution were reported. These results showed that the B&B algorithm performed very well in achieving the global optimum solution, with the largest instances (with 100 jobs) being solved in less than 500 s on average. MILP, on the other hand, is inefficient even when solving instances with 20 jobs in a logical running time (3600 s). In addition to the two exact methods mentioned above, a heuristic algorithm based on the Decreasing Neighborhood Search (HDNS) method was presented in this paper, which had very good speed performance. This algorithm solved small, medium, and large instances with an average error of 2.2%, 4.1%, and 8.9%.

This study faced some limitations. The uncertainty in the parameters, particularly in the processing times and the different job release times, was not considered in this study. Future studies are hence recommended to take into account this issue. Furthermore, in both the first and second stages, production environments were assumed to be a single machine, which only covers a portion of the manufacturing industry. Future studies are recommended to model this environment as a two-stage assembly flow shop or a multi-stage assembly flow shop (more than two stages).

Another area of research for future studies is to simultaneously consider the inter-factory and outer-factory batch delivery systems in the supply chain network. In addition, one of the interesting future studies in this field could be the integration of order acceptance decisions with their production schedule in a decentralized flow shop environment.

Declarations

Conflict of interest No potential conflict of interest was reported by the authors.

References

Abedi M, Seidgar H (2016) A new bi-level meta-heuristic approach for a single machine JIT-scheduling in the batch delivery system with controllable due dates. Int J Serv Op Manag 23(2):135–152

Ahmadizar F, Farhadi S (2015) Single-machine batch delivery scheduling with job release dates, due windows and earliness, tardiness, holding and delivery costs. Comput Oper Res 53:194–205

Bachtenkirch D, Bock S (2022) Finding efficient make-to-order production and batch delivery schedules. Eur J Oper Res 297(1):133–152

Basir SA, Mazdeh MM, Namakshenas M (2018) Bi-level genetic algorithms for a two-stage assembly flow-shop scheduling problem with batch delivery system. Comput Ind Eng 126:217–231

Cakici E, Mason SJ, Geismar HN, Fowler JW (2014) Scheduling parallel machines with single vehicle delivery. J Heuristics 20(5):511–537

Chan F T Kumar V Mishra N. (2008). resolving multi plant supply chain problem: a novel swarm intelligence based approach.In: 2008 4th IEEE International Conference on Management of Innovation and Technology

Chen Z-L (2010) Integrated production and outbound distribution scheduling: review and extensions. Oper Res 58(1):130–148

Cui H, Li X, Gao L (2023) An improved multi-population genetic algorithm with a greedy job insertion inter-factory neighborhood structure for distributed heterogeneous hybrid flow shop scheduling problem. Expert Syst Appl 222:119805

De Matta R, Miller T (2004) Production and inter-facility transportation scheduling for a process industry. Eur J Oper Res 158(1):72–88

Dorfeshan Y, Tavakkoli-Moghaddam R, Mousavi SM, Vahedi-Nouri B (2020) A new weighted distance-based approximation methodology for flow shop scheduling group decisions under the interval-valued fuzzy processing time. Appl Soft Comput 91:106248

Fu Y, Zhou M, Guo X, Qi L (2019) Scheduling dual-objective stochastic hybrid flow shop with deteriorating jobs via bi-population evolutionary algorithm. IEEE Trans Syst, Man, Cybern: Syst 50(12):5037–5048

Fu Y, Ma X, Gao K, Li Z, Dong H (2023) Multi-objective home health care routing and scheduling with sharing service via a problem-specific knowledge-based artificial bee colony algorithm. IEEE Trans Intell Transp Syst 25(2):1706–1719

Fu, Y., Hou, Y., Guo, X., Qi, L., & Wang, H. (2022). Scheduling a Permutation Flow Shop in the Context of Industry 4.0. In *2022 IEEE International Conference on Networking, Sensing and Control* (pp. 1–6). IEEE.

Gharaei A, Jolai F (2021) A Pareto approach for the multi-factory supply chain scheduling and distribution problem. Oper Res Int Journal 21(4):2333–2364

Gong H, Zhang B, Peng W (2015) Scheduling and common due date assignment on a single parallel-batching machine with batch delivery. Discrete Dyn Nat Soc 2015(1):464390

Hall NG, Potts CN (2003) Supply chain scheduling: batching and delivery. Oper Res 51(4):566–584

Hamidinia A, Khakabimamaghani S, Mazdeh MM, Jafari M (2012) A genetic algorithm for minimizing total tardiness/earliness of weighted jobs in a batched delivery system. Comput Ind Eng 62(1):29–38

Ji M, He Y, Cheng TE (2007) Batch delivery scheduling with batch delivery cost on a single machine. Eur J Oper Res 176(2):745–755

Jia ZH, Zhuo XX, Leung JY, Li K (2019) Integrated production and transportation on parallel batch machines to minimize total weighted delivery time. Comput Oper Res 102:39–51

Karimi N, Davoudpour H (2017) Integrated production and delivery scheduling for multi-factory supply chain with stage-dependent inventory holding cost. Comput Appl Math 36(4):1529–1544

Kazemi H, Mazdeh MM, Rostami M (2017) The two stage assembly flow-shop scheduling problem with batching and delivery. Eng Appl Artif Intell 63:98–107

Kong M, Pei J, Xu J, Liu X, Yu X, Pardalos PM (2020) A robust optimization approach for integrated steel production and batch delivery scheduling with uncertain rolling times and deterioration effect. Int J Prod Res 58(17):5132–5154

Lei D, Su B, Li M (2021) Cooperated teaching-learning-based optimisation for distributed two-stage assembly flow shop scheduling. Int J Prod Res 59(23):7232–7245

Li H, Li X, Gao L (2021a) A discrete artificial bee colony algorithm for the distributed heterogeneous no-wait flowshop scheduling problem. Appl Soft Comput 100:106946

Li Q, Li J, Zhang X, Zhang B (2021b) A wale optimization algorithm for distributed flow shop with batch delivery. Soft Comput 25(21):13181–13194

Lohmer J, Lasch R (2021) Production planning and scheduling in multi-factory production networks: a systematic literature review. Int J Prod Res 59(7):2028–2054

Marandi F, Ghomi SF (2019) Network configuration multi-factory scheduling with batch delivery: a learning-oriented simulated annealing approach. Comput Ind Eng 132:293–310

Mazdeh MM, Rostami M (2014) A branch-and-bound algorithm for two-machine flow-shop scheduling problems with batch delivery costs. Int J Syst Sci: Op Logistics 1(2):94–104

Mazdeh MM, Sarhadi M, Hindi KS (2007) A branch-and-bound algorithm for single-machine scheduling with batch delivery minimizing flow times and delivery costs. Eur J Oper Res 183(1):74–86

Mazdeh MM, Sarhadi M, Hindi KS (2008) A branch-and-bound algorithm for single-machine scheduling with batch delivery and job release times. Comput Oper Res 35(4):1099–1111

Mazdeh MM, Rostami M, Namaki MH (2013) Minimizing maximum tardiness and delivery costs in a batched delivery system. Comput Ind Eng 66(4):675–682
Moon C, Kim J, Hur S (2002) Integrated process planning and scheduling with minimizing total tardiness in multi-plants supply chain. Comput Ind Eng 43(1–2):331–349
Noroozi A, MahdaviMazdeh M, Noghondarian K, Rasti-Barzoki M, Heydari M (2018) Evolutionary computation algorithms to coordinating order acceptance and batch delivery for an integrated supply chain scheduling. Comput Appl Math 37(2):1629–1679
Noroozi A, Mazdeh M, Heydari M, Rasti-Barzoki M (2019) Coordinating order acceptance and integrated lot streaming-batch delivery scheduling considering third party logistics. Uncertain Supply Chain Manag 7(1):73–96
Olhager J, Feldmann A (2018) Distribution of manufacturing strategy decision-making in multi-plant networks. Int J Prod Res 56(1–2):692–708
Ozkan O, Ermis M, Bekmezci I (2020) Reliable communication network design: the hybridisation of metaheuristics with the branch and bound method. J Op Res Soc 71(5):784–799
Pan Y, Gao K, Li Z, Wu N (2022) Improved meta-heuristics for solving distributed lot-streaming permutation flow shop scheduling problems. IEEE Trans Autom Sci Eng 20(1):361–371
Qi X, Yuan J (2017) A further study on two-agent scheduling on an unbounded serial-batch machine with batch delivery cost. Comput Ind Eng 111:458–462
Rahman HF, Janardhanan MN, Chuen LP, Ponnambalam SG (2021) Flowshop scheduling with sequence dependent setup times and batch delivery in supply chain. Comput Ind Eng 158:107378
Rossi FL, Nagano MS (2021) Heuristics and iterated greedy algorithms for the distributed mixed no-idle flowshop with sequence-dependent setup times. Comput Ind Eng 157:107337
Rostami M (2021) Minimizing maximum tardiness subject to collect the EOL products in a single machine scheduling problem with capacitated batch delivery and pickup systems. Comput Ind Eng 161:107634
Rostami M, Shad S (2020) A Hybrid Bee Algorithm for Two-Machine Flow-Shop Scheduling Problems with Batch Delivery. J Qual Eng Prod Optim 5(1):137–164
Rostami M, Yousefzadeh A (2023) A gamified teaching–learning based optimization algorithm for a three-echelon supply chain scheduling problem in a two-stage assembly flow shop environment. Appl Soft Comput 146:110598
Rostami M, Kheirandish O, Ansari N (2015) Minimizing maximum tardiness and delivery costs with batch delivery and job release times. Appl Math Model 39(16):4909–4927
Rostami M, Nikravesh S, Shahin M (2020) Minimizing total weighted completion and batch delivery times with machine deterioration and learning effect: a case study from wax production. Oper Res Int J 20(3):1255–1287
Ruifeng C, Subramaniam V (2011) Performance evaluation for tandem multi-factory supply chains: an approximate solution. Int J Prod Res 49(11):3285–3305
Salehi F, Al-e SMJM, Husseini SMM (2022) A 2-phase interdependent methodology for sustainable project portfolio planning in the pharmaceutical industry. Comput Ind Eng 174:108794
Shao W, Shao Z, Pi D (2021) Multi-objective evolutionary algorithm based on multiple neighborhoods local search for multi-objective distributed hybrid flow shop scheduling problem. Expert Syst Appl 183:115453
Sun X, Chung SH, Chan FT (2015) Integrated scheduling of a multi-product multi-factory manufacturing system with maritime transport limits. Transp Res Part E: Logist Transp Rev 79:110–127
Taillard E (1993) Benchmarks for basic scheduling problems. Eur J Oper Res 64(2):278–285
Villalonga A, Negri E, Biscardo G, Castano F, Haber RE, Fumagalli L, Macchi M (2021) A decision-making framework for dynamic scheduling of cyber-physical production systems based on digital twins. Annu Rev Control 51:357–373
Wanapinit N, Thomsen J, Kost C, Weidlich A (2021) An MILP model for evaluating the optimal operation and flexibility potential of end-users. Appl Energy 282:116183
Wang G, Cheng TE (2000) Parallel machine scheduling with batch delivery costs. Int J Prod Econ 68(2):177–183
Wang K, Luo H, Liu F, Yue X (2017) Permutation flow shop scheduling with batch delivery to multiple customers in supply chains. IEEE Trans Syst, Man, Cybern: Syst 48(10):1826–1837
Yang S, Xu Z (2021) The distributed assembly permutation flowshop scheduling problem with flexible assembly and batch delivery. Int J Prod Res 59(13):4053–4071
Yin Y, Cheng T, Hsu C-J, Wu C-C (2013) Single-machine batch delivery scheduling with an assignable common due window. Omega 41(2):216–225

Yin Y, Yang Y, Wang D, Cheng T, Wu CC (2018) Integrated production, inventory, and batch delivery scheduling with due date assignment and two competing agents. Naval Res Logist (NRL) 65(5):393–409

Zhang B, Lu C, Meng LL, Han YY, Sang HY, Jiang XC (2023) Reconfigurable distributed flowshop group scheduling with a nested variable neighborhood descent algorithm. Expert Syst Appl 217:119548

Zhang W, Li C, Gen M, Yang W, Zhang G (2024) A multiobjective memetic algorithm with particle swarm optimization and Q-learning-based local search for energy-efficient distributed heterogeneous hybrid flow-shop scheduling problem. Expert Syst Appl 237:121570