RAIRO-Oper. Res. 56 (2022) 4129–4143
https://doi.org/10.1051/ro/2022194

RAIRO Operations Research
www.rairo-ro.org

A HYPER-HEURISTIC FOR DISTRIBUTED PARALLEL MACHINE SCHEDULING WITH MACHINE-DEPENDENT PROCESSING AND SEQUENCE-DEPENDENT SETUP TIMES

Javad Behnamian* and Hamed Asgari

Abstract. Today, because the market is scattered around the world, manufacturing activities are not limited to a single location and have spread globally. As a result, the discussion of scheduling the factory has changed from a classic single to a network scheduling as a need in the real world. In this regard, this study considers the scheduling of multiple factories by taking into account the job transportation time between factories. The main problem here is that each job would be assigned to which factory and machine. In this research, unrelated parallel machines are considered in which the processing time of jobs depends on the machine and setup time. To minimize the makespan, first, a mixed-integer linear model was proposed in which two types of modeling have been combined. Then, a hyper-heuristic algorithm (HHA) was designed to solve the problem in a reasonable time by choosing the best method among four low-level heuristic methods that are precisely designed according to the properties of the problem. Finally, the efficiency of the proposed algorithm has been compared with the imperialist competitive algorithm (ICA) by conducting experiments. The results show that the proposed algorithm performs very well compared to the ICA and, in more than 75% of the test problems, the proposed algorithm was superior. Also, based on the analysis, in comparing the proposed algorithm with the ICA, it can be concluded that there is a significant difference between the results, and in all cases, the HHA was remarkably better. Considering the challenges and rapid changes of today's market that traditional centralized production planning does not have enough flexibility to respond to them, the results of this research are expected to be useful and attractive for planners in this field.

Mathematics Subject Classification. 90B35, 68M14, 90C59.

Received March 29, 2022. Accepted October 30, 2022.

1. Introduction

In classic scheduling research, it is assumed that a job is centrally produced in a single factory [11]. But as real problems become more complex, and with the current globalization process, the high volume of diversified demand in the markets, and the pressure on innovation to maximize benefits, the centralized approach is not effective enough in the real world. In order to maintain competitiveness in such markets, the factories, with their growth by default, configure internally to enjoy economies of scale by increasing the level of capacity.

Keywords. Cooperative production scheduling, hyper-heuristic algorithm, unrelated parallel machine, sequence-dependent setup time, transportation time.

Department of Industrial Engineering, Faculty of Engineering, Bu-Ali Sina University, Hamedan, Iran.
*Corresponding author: behnamian@basu.ac.ir

The continued growth of factories has necessitated the decision to establish a production network consisting of multi-factories. On the other hand, changes in manufacturing factories and the interest of smaller companies in entering the global market, have created a new challenge in structuring efficient operations management in geographically distributed production networks [3]. As a result, the small organizations are interested in merging to overcome the following problems:

- Response time to large stochastic disturbances in the market is not satisfactory due to the lack of agility of traditional production systems.
- Insufficient, inaccurate, and unreliable information due to the geographical size of the customer distribution has led decision-makers to make decisions based on conjecture or very little information.
- The organizational structure of traditional systems is predetermined, which makes it inflexible to the emergence of new markets and changes.

In such an environment, factories decide to merge and form a multi-factory production network to work more closely together [8]. The production system of multi-factory means that several factories are located in different geographical areas in order to achieve greater profit and to achieve globalization. This integration will allow factories to get closer to their customers, hire professional employees, adapt more to local rules, produce more efficient products, and respond more quickly to market changes. The main difference between single-factory scheduling and multi-factory production scheduling is that in single-factory production, the products are produced by a single site and then marketed. But in the production of multi-factory, the products can be produced in multi-factory, which may be located in different regions. So sometimes, they may be able to reap the benefits of such integration. In addition, not all factories are able to do all kinds of jobs. In other words, some factories perform some jobs better.

In the meantime, it should be noted that there are fundamental differences between single-factory and multi-factories production. In a single factory, the products are produced using one factory and then transferred to the points of sale, which have the task of distributing or retailing these products, while in the case of network production, this manufacturer may produce some products in multi-factories. In the mentioned network, the decision is delegated to lower levels of the organization hierarchy and resolved locally in different system institutions. The solutions are then coordinated together under a global objective function. The multi-factory production takes place in multi-factories, which may be geographically distributed in different places to satisfy the demand and adapt to the globalization trend.

In the multi-factory production scheduling, since most research have been devoted to factories with identical parallel machines, in this study, it is assumed each factory has unrelated parallel machines. In this research, considering sequence-dependent setups, an attempt was made to bring the problem closer to real-world situations. Note that the unrelated parallel machine scheduling is the general mode of the parallel machine scheduling problem, in which the processing time of the job on the machines depends not only on the type of job but also on the type of machine [11]. The problem of distributed scheduling is the same as classic scheduling when the number of factories is one, and the problem of scheduling even with two machines with the makespan objective function is an NP-hard problem [6]. In fact, the problem under our investigation is bi-assignment scheduling in which the first assignment is made when appropriate factory is chosen for each job, and the second one is made when the jobs are assigned to the factories. In this stage, appropriate machine from among the unrelated machines according to the sequence-dependent setups is selected for each job assigned to each factory. In this environment, adding new constraints, such as sequence-dependent setup times, makes the problem more complicated [24]. Therefore, our considered problem, distributed scheduling with the unrelated parallel machines, is also NP-hard, and a heuristic/meta-heuristic algorithm must be used to solve it in a reasonable time, especially in large-size instances [2]. In this regard, the present study uses a hyper-heuristic algorithm to approach the optimal solution. The proposed hyper-heuristic as a computing manager was designed to solve the problem in a reasonable time by choosing the best method among four low-level heuristic methods that are defined according to the properties of the problem, at each stage of the search process, and taking into account the properties of the problem and the solution space.

The rest of this study is organized as follows: The second section deals with the literature related to the present study. In the third section, the problem is defined and then modeled. Section four describes the algorithm used in the research, and Section five discusses the validity of the proposed algorithm in small, medium, and large-size instances, and finally, Section six summarizes, concludes and elaborates our suggestions for future research.

2. Literature review

In recent years, distributed systems have become increasingly important, with their application to a wide range of sciences (from multimedia to production control) and many efforts to use them in other areas [3]. In the field of production, industrial managers in recent years have tended to use distributed systems due to changes in the production environment. Production in an environment distributed in a multi-factory production network is one of the most attractive topics in recent years [28].

In multi-factory scheduling, various problems have been investigated, among which the research of Williams [30] can be mentioned, which is the first research conducted on the subject of distributed scheduling. In that study, Williams outlined common scheduling for the production and distribution of complex networks. To this end, he proposed a method based on dynamic programming and compared it with heuristic methods in a centralized production environment to examine performance. In this study, the objective function was to minimize production and distribution costs in each time period. One of the weaknesses of the research is that the author has assumed that the demand rate is constant, which has reduced its applicability. To minimize production costs considering the product demand rates during the planning time horizon, Giordani *et al.* [13] proposed a decentralized multi-agent scheduling system with a number of agents in the system, with a resource owner which assigns the robots to the tasks in each time period. Behdani *et al.* [4] have considered the organization of multi-factory production in the chemical industry as a complex company of producers that are distributed in different geographical locations who are responsible for the production and distribution of specific products. Given some of the realities of the problem, researchers have shown how factor-based models can reflect the dynamic behavior of a multifactorial network. To this end, they compared different mechanisms in such models to study the behavior of the proposed model under equal conditions in both centralized and distributed modes. In today's international market, none of the small factories can operate independently well. Therefore, a number of different factories are integrated to form a production network. In this case, factories can work economically separated too. With the globalization process occurring, factories are moving towards integration at different speeds. The development of cheap transportation and communications networks has also accelerated this globalization process. Lei *et al.* [15] considered a distributed unrelated parallel machine scheduling problem in a heterogeneous production network. For minimizing makespan, an imperialist competitive algorithm (ICA) was presented.

Gharaei and Jolai [12] examined the complexity of supply chain scheduling in different cases and proposed rapid heuristic methods for each case. The problem under consideration in this research is that the production and distribution network includes multi-parallel factories in foreign countries and one domestic distribution center. In this network, the job is first processed in factories and then transported to the distribution center to be distributed among retailers in other countries. Depending on the diversity of productivity and costs in different factories, the costs and processing times vary depending on which factory the job is processed. In this paper, the authors consider four different objective functions for their problem, all of which include delivery times. Finally, some heuristic methods are used to solve the presented problem.

Behnamian [6] provided a hybrid algorithm for searching for variable neighbors and tabu search based on analysis for scheduling parallel factories distributed with virtual corporations. A virtual production network is a distributed system in which smaller parts of a scheduling problem are solved by local decision-makers, who may also have different objective functions. This network is a new type of corporation and the horizontal relationship between independent factories. Factories in this network can be identical or non-identical. In fact, virtual corporation means that multiple factories work together to produce an order. Behnamian *et al.* [7] considered a multi-factory production schedule in which a group of independent-owned producers joined together to create a production network. Also known as a virtual corporation, each factory, as a separate member, usually focuses on its own interests and strives to improve them, and is less concerned with the interests of other members. The objective function minimizes the makespan in this problem. Despite the fact that the problem was single-objective, the discussion of dominant and non-dominated solutions was first proposed in the space of single-objective problems. Then, a hybrid algorithm based on particle swarm optimization and the hyper-heuristic method was proposed.

Behnamian [5] presented a mathematical model and a new algorithm to solve distributed network scheduling. In this model, parallel and non-homogeneous factories are located in different places. They then used the swarm particle optimization algorithm to minimize makespan in the network. The idea of this algorithm is derived from a society in which members behave in chaos to improve their position. This algorithm can prevent trapping in local areas. Finally, they tested the performance of the particle optimization algorithm provided, the standard particle optimization, and a genetic algorithm for the number of test problems. Recently, Behnamian and Fatemi Ghomi [9] studied a multi-factory scheduling problem with heterogeneous factories and identical parallel machines. For simultaneous minimization of the sum of the earliness and tardiness of jobs and total completion time, after modeling the problem as a mixed-integer linear program, the elastic constraints method and heuristic algorithm are proposed for this problem.

Lei and Liu [14] investigated the distributed unrelated parallel machine scheduling problems with preventive maintenance and an artificial bee colony with division is proposed to minimize makespan. Abdollahzadeh *et al.* [1] considered the integrated scheduling of production and distribution operations in a parallel-multi-factory environment with identical parallel machines available at each factory in a make-to-order production system. They proposed a mixed-integer nonlinear programming model to minimize the total costs of the supply chain, including production, distribution, and late delivery costs. To solve large-size instances, a whale optimization algorithm is developed in that research. Zhou and Yang [33] considered a dynamic flexible job shop scheduling problem and proposed four multi-objective genetic programming-based hyper-heuristic methods for it. Their proposed algorithm had two phases: job sequencing and machine assignment. In this study, the objective function includes mean weighted tardiness, maximum tardiness and mean flow time. For the distributed assembly permutation flow-shop scheduling problem, Lin *et al.* [17] proposed a backtracking search hyper-heuristic. In the proposed algorithm, ten heuristics are designed as low-level heuristics, and the backtracking search algorithm is used as the high-level strategy. Yahyaoui *et al.* [31] combined iterated local search with a variable neighborhood hyper-heuristic algorithm. They used the algorithm to minimize the makespan and total flowtime algorithm in a permutation flowshop scheduling problem. Park *et al.* [22] proposed a genetic programming-based hyper-heuristic that uses several dispatching rules to solve a dynamic job shop scheduling problem. Lin [16] studied the flexible job-shop scheduling problem with fuzzy processing and proposed a backtracking search-based hyper-heuristic to solve it. In the proposed algorithm, six low-level heuristics, as well as a backtracking search algorithm, were used.

To minimize adjustment and energy consumption, Mou *et al.* [21] proposed a model for the energy-efficient distributed permutation flow-shop inverse scheduling problem with controllable processing times and energy consumption factors. They also designed a hybrid collaborative algorithm with a cooperative search scheme. Shao *et al.* [27] proposed a mixed-integer linear programming model for a distributed flowshop scheduling problem. Their considered production network includes the heterogeneous multi-factories with different processing capabilities of machines and electricity prices in multi-factories. To solve large-size instances, they also proposed four greedy local search methods with a right-shifting procedure that move jobs between heterogeneous multi-factories. Mao *et al.* [20] studied the distributed permutation flowshop scheduling problem with preventive maintenance. To minimize the total flow time, they proposed a mathematical model and a hash map-based algorithm. They also modified the selection, crossover, and mutation operators in the proposed algorithm. To minimize the total tardiness, Wang *et al.* [29] proposed a mathematical model for a distributed flowshop group scheduling problem with sequence-dependent setup time. They also designed a two-stage iterated greedy algorithm with a two-stage structure. Luo *et al.* [19] studied the distributed flexible job shop scheduling problem with worker arrangement. To solve small-size instances, in this study a mixed-integer linear programming model was formulated. The authors also, based on the structure of NSGA-II, designed a memetic algorithm for large-size instances in which they were taken advantage of a two-level encoding as well as four heuristic decoding methods. Lu *et al.* [18] designed a Pareto-based multi-objective hybrid iterated greedy algorithm for a distributed hybrid flowshop scheduling problem. To minimize the makespan and total energy consumption, they also, embedded a knowledge-based multi-objective local search method and energy-saving technique in the proposed algorithm. Zhao *et al.* [32] proposed a memetic discrete differential evolution algorithm for the distributed permutation flowshop scheduling problem. To minimize the makespan, they applied an enhanced NEH (Nawaz–Enscore–Ham) and Taillard's acceleration methods. Schulz *et al.* [26] studied a multi-objective distributed permutation flowshop scheduling problem. To equally minimize makespan and carbon emission caused by both production and transportation, they proposed a mixed-integer programming model with the epsilon constraint method. To solve large-size instances, in this study, a multi-objective iterated greedy algorithm was also proposed. Sang and Tan [25] modeled the many-objective distributed flexible job shop problem as a mixed-integer program. To optimize the economic indicators and green indicators, they combined the improved NSGA-III and local search method. In the proposed high-dimensional many-objective memetic algorithm, the neighborhood structure, as well as a dual-mode environment selection method, were also utilized. Wang *et al.* [29] studied an energy-efficient distributed production scheduling problem with identical factories of welding flowshop. This problem was formulated as a multi-objective mixed integer programming model based on three sub-problems with allocating jobs among factories, scheduling the jobs in each factory and determining the number of machines upon each job. They also, designed a whale swarm algorithm to optimize the total energy consumption and makespan.

According to Bagheri and Behnamian [3], as well as the reviewed papers in this study, it can be said that although researchers approach many problems of distributed scheduling with different objective functions and constraints that bring the problems closer to the real world, but the vacancy of this type of problem can be seen in environments with the unrelated parallel machines. Therefore, the present study examines the problem of distributed scheduling with unrelated parallel machines to fill part of the existing research gap. It should be noted that the closest study for this research is a paper by Lei *et al.* [14], in which job transportation time among distributed factories as the main feature of the distributed environment is ignored. In this regard, not considering the time of transportation between the distributed factories will cause the problem to be out of the distributed mode, and in fact, the problem will become an integrated scheduling problem with several unrelated parallel machines in the single-site production environment. In order to eliminate such a major shortcoming, as well as to make the considered problem more realistic, in addition to taking into account the job transportation time between factories, sequence-dependent setup times are also considered in this paper. The hyper-heuristic algorithm is also applied to solve the problem, which has received less attention in previous research.

3. Problem description and modeling

In this research, two types of conventional modeling (modeling based on allocation and sequencing) have been combined. In the next step, to confirm the accuracy of the model, the problem was solved for small-size instances by GAMS software, and the optimal solution was reached.

In this study, the following assumptions are considered:

– There is a number of parallel factories with unrelated parallel machines.
– There is not any precedence between jobs.
– Each job must be assigned exactly to one of the factories.
– Setup time depends on the sequence.
– All jobs and machines are available at the beginning of the schedule.

The following are the indices, parameters, and decision variables used in the present study:

Parameters and indices

n	Number of jobs
M	Total number of all machines in all factories

F	Number of factories
i, j, k	Indices of jobs
f, q	Indices of factories
h	Indices of machines
m^f	Number of machines in factory f
P^f_{hj}	Processing time of job j on machine h in factory f
S_{hij}	Setup time of job j after job i on machine h
t^{fq}	Transportation time for each job between factory f and q
w_i	A binary variable which is one if job i processed in origin factory (that the jobs were originally ordered to them); otherwise, 0
L	A large positive number

Decision variables

x^f_{hij}	A binary variable that takes the value of one if job j is processed after job i in factory f is processed on machine h
y^f_{hi}	A binary variable that takes the value of one if job i in factory f is processed on machine h
C^f_{hi}	Completion time of job i on machine h in factory f
$C_{\max}$	Time to complete the last job (makespan)

The proposed model is as follows.

$$Z = \text{Min } C_{\max} \tag{1}$$

$$\text{s.t.} \sum_{h=1}^{M}\sum_{f=1}^{F} y^f_{hi} = 1 \qquad i = 1, 2, \ldots, n \tag{2}$$

$$\sum_{h=1}^{M}\sum_{\substack{i=1\\ i\neq j}}^{n}\sum_{f=1}^{F} x^f_{hij} = 1 \qquad j = 1, 2, \ldots, n \tag{3}$$

$$\sum_{h=1}^{M}\sum_{\substack{j=1\\ j\neq i}}^{n}\sum_{f=1}^{F} x^f_{hij} \leq 1 \qquad j = 1, 2, \ldots, n \tag{4}$$

$$\sum_{h=1}^{M}\sum_{f=1}^{F}\left(x^f_{hij} + x^f_{hji}\right) \leq 1 \qquad i = 1, 2, \ldots, n-1, j > i \tag{5}$$

$$\sum_{\substack{i=1\\ i\neq j}}^{n} x^f_{hij} + \sum_{\substack{k=1\\ k\neq j}}^{n} x^f_{hjk} = 2y^f_{hj} \qquad j = 1, 2, \ldots, n \quad f = 1, 2, 3, \ldots, F \quad h = 1, 2, \ldots, M \tag{6}$$

$$C^f_{hj} + L\left(1 - x^f_{hij}\right) \geq C^f_{hi} + S^f_{hij} + P^f_{hj} + \left(2w_j \times t^{fq}\right) \qquad i = 0, 1, \ldots, n \quad j = 1, 2, \ldots, n \quad h = 1, 2, \ldots, M \tag{7}$$

$$C^f_{hi} \geq 0 \qquad h = 1, 2, \ldots, M \quad i = 1, 2, \ldots, n \quad f = 1, 2, \ldots, F \tag{8}$$

$$C_{\max} \geq C^f_{hi} \qquad h = 1, 2, \ldots, M \quad i = 1, 2, \ldots, n \quad f = 1, 2, \ldots, F \tag{9}$$

$$x^f_{hij}, y^f_{hi} \in \{0, 1\} \qquad i = 0, 1, .., n \quad h = 1, 2, \ldots, M \quad f = 1, 2, \ldots, F \quad j \neq i. \tag{10}$$

Equation (1) is the objective function of this model, which indicates the completion time of all jobs, which should be minimized. Constraint (2) specifies that each job must be assigned to a machine in a factory. Constraint (3) states that each job in each factory is processed exactly on one machine. Constraint (4) states that each job on each machine in each factory can have a maximum of one job as the next job. Constraint (5) indicates that one job cannot be another's previous and next job at the same time. Constraint (6) states that if a job is assigned to a machine in a factory, it will have exactly one previous job and one next job (considering dummy jobs 0 and $n+1$). Constraint (7) calculates the amount of processing time of a job when assigned to a specific machine in a particular factory, including the setup time depending on the sequence and time of transportation (time of return from the original factory to another factory with the aim of achieving balance). Equation (8) indicates that the duration of each job on a particular machine in a particular factory is positive. Constraint (9) calculates the completion time of all jobs. Constraint (10) is related to model variables.

4. Hyper-heuristic algorithm

Due to the NP-hardness of the considered problem, the exact method cannot solve it in large size in a reasonable time [23]. The hyper-heuristic algorithm (HHA) was first introduced by Cowling and Soubeiga [10] as a higher-level approach of metaheuristic methods. They defined it as: "The HHA as a computing manager should be designed to solve the problem in a reasonable time by choosing the best method among the low-level heuristic methods defined for it, at each stage of the solution, and taking into account the properties of the problem and the solution space".

Hyper-heuristic algorithms are among those methods that seek appropriate solutions in a reasonable time and are applicable to a wide range of problems. The main difference between HHAs and metaheuristic methods is the generality of metaheuristic algorithms that can be modified and used depending on the type of problem, the operators and their parameters. Whereas HHAs are problem-based approaches, it is less likely that a specific HHA would be applicable to any other problem. In fact, HHAs are based on several heuristics called low-level heuristics that are designed for specific problem situations. Low-level heuristics are an inseparable part of HHAs, which are some kind of simple local search operator or heuristic. In other words, the core of the HHA has the task of choosing the heuristic method to solve the problem. Given that each heuristic has its strengths and weaknesses, it seems wise to employ one of them in each step of the solution space search, depending on the circumstances of each heuristic to obtain a better solution.

There are generally two types of HHAs: constructive and local search. Constructive HHAs have the ability to generate and improve initial solutions. Whereas in local search HHAs, a complete preliminary solution is needed to improve, using appropriate choices of local search methods in different iterations. Local search methods are usually based on some neighborhood or local search that can change the current state of the solutions. Similar to metaheuristic algorithms, to avoid getting caught up in local optimizations, here, a technique is needed to be added to the HHA to support its diversification.

4.1. Implementation details

The monte-Carlo-based hyper-heuristic algorithm sends an initial solution as the core of the computational algorithm to low-level algorithms so that they can perform a local search by their definition of neighborhood structure for a specific number of iterations, with the focus being on the solution from the core. Then each of them sends a new solution from their search that may be better, worse, or even equal to the input objective function, to the core of the algorithm. Once all solutions have been received, the core of the algorithm stores the best heuristic algorithm and its corresponding solution among all the received ones. The best low-level heuristic algorithm is the one that produces the most improvement in the current objective function or the least deterioration of the solution when none of the algorithms can improve the current solution. In such cases, the Monte Carlo criterion is used, this means that the solution which has the least deterioration is first examined, and if not accepted as the next solution, the second one is selected from the list of solutions. This operation will continue until the solution for the next step of the algorithm is created. In the worst case, when all solutions are rejected, a random change such as a mutation operator in the previous step is used as the next step input of low-level algorithms and this will continue until the algorithm stopping criterion (number of iterations) is satisfied. According to the Monte Carlo criterion, the best solution is always accepted and if there is no improvement

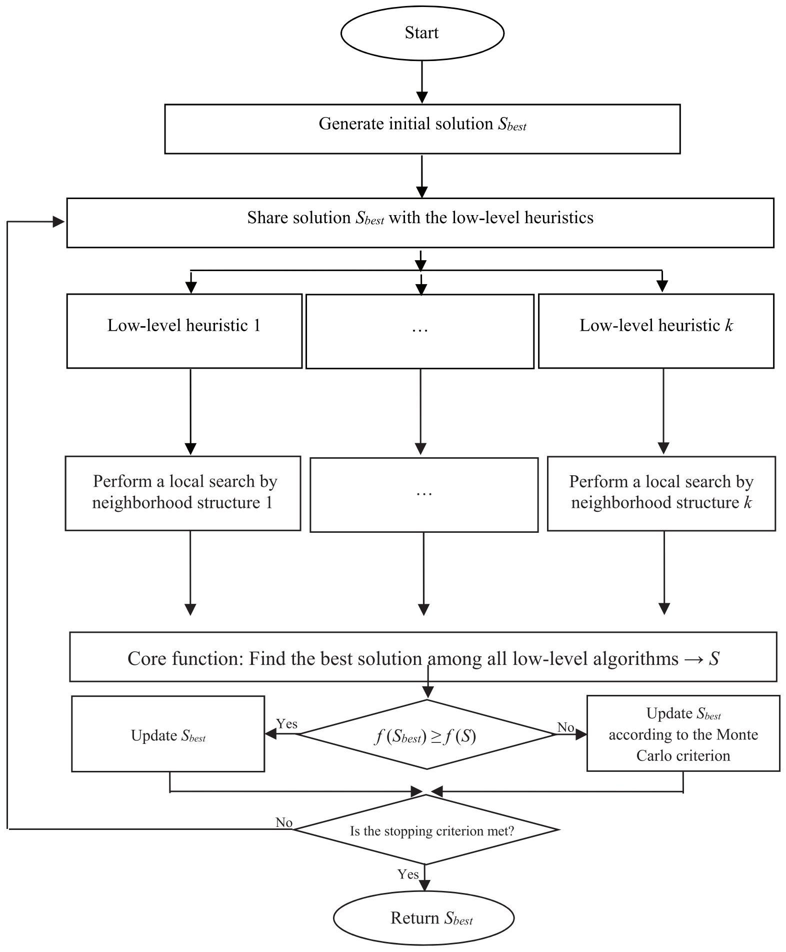

Start
Generate initial solution S_{best}
Share solution S_{best} with the low-level heuristics
Low-level heuristic 1
…
Low-level heuristic k
Perform a local search by neighborhood structure 1
…
Perform a local search by neighborhood structure k
Core function: Find the best solution among all low-level algorithms → S
Update S_{best}
Yes
$f(S_{best}) \geq f(S)$
No
Update S_{best} according to the Monte Carlo criterion
No
Is the stopping criterion met?
Yes
Return S_{best}

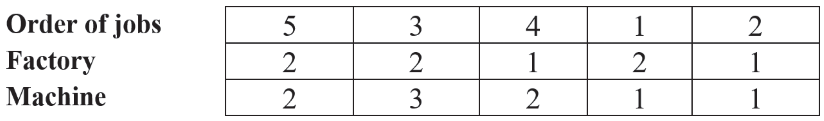
FIGURE 1. Flowchart of the proposed hyper-heuristic.

in the solution sent by the algorithm core to the low-level heuristics, the worse solution is accepted with a particular probability. This probability depends on the number of repetitions and how much worse the solution is compared to the previous one. Acceptance of the solution is as follows: if $f(s_{\text{best}}) \leq f(s)$, then $f(s_{\text{best}})$ will be accepted; otherwise, the worse solution will be accepted if $e^{-(f(s_{\text{best}})-f(s))}$.iter is smaller than or equal to the random number between zero and one. The flowchart of the proposed hyper-heuristic algorithm is shown in Figure 1.

4.2. Solution representation

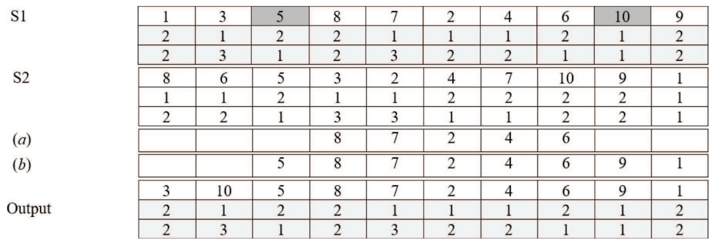
The proposed representation to solve the distributed scheduling problem is based on coding all jobs as a matrix in a 3-by-N string where N is the number of jobs. In this matrix, the first row shows the order of jobs. The second row represents the factory in which a specific job is processed, and the third row illustrates the relevant machine number in that factory.

For example, as shown in Figure 2, it is assumed to be the scheduling of five jobs in two factories with two machines in the first factory and three machines in the second factory.



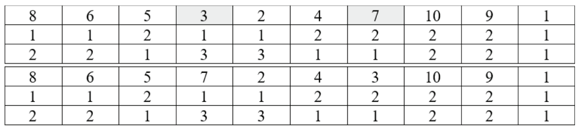
Order of jobs	5	3	4	1	2
Factory	2	2	1	2	1
Machine	2	3	2	1	1

Figure 2. Solution representation.



S1	1	3	5	8	7	2	4	6	10	9
	2	1	2	2	1	1	1	2	1	2
	2	3	1	2	3	2	2	1	1	2
S2	8	6	5	3	2	4	7	10	9	1
	1	1	2	1	1	2	2	2	2	1
	2	2	1	3	3	1	1	2	2	1
(*a*)				8	7	2	4	6		
(*b*)			5	8	7	2	4	6	9	1
	3	10	5	8	7	2	4	6	9	1
Output	2	1	2	2	1	1	1	2	1	2
	2	3	1	2	3	2	2	1	1	2

Figure 3. Local Search 1.



8	6	5	3	2	4	7	10	9	1
1	1	2	1	1	2	2	2	2	1
2	2	1	3	3	1	1	2	2	1
8	6	5	7	2	4	3	10	9	1
1	1	2	1	1	2	2	2	2	1
2	2	1	3	3	1	1	2	2	1

Figure 4. Local Search 2.

4.3. Local searches

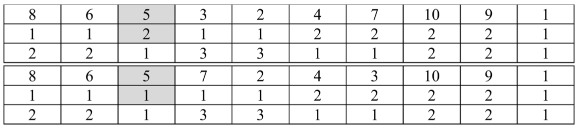
The local search method examines all current neighbor states for optimal results. The following is an introduction to the four neighborhood structures.

Local Search 1: *Knowledge sharing.* Here, in addition to the solution sent from the core (S1), a new string (S2) is also generated randomly. Then, the cells between the strings in the first rows (order of jobs) are matched. The cells between the two members are moved and the remaining cells are copied from the correspondence. For example, with ten jobs, suppose three or nine colonial members are selected. As can be seen, the cells between the no. 3 and 9, have been copied in the corresponding cells in (a). So far, in the output, jobs 8, 7, 2, 4, and 6 have been copied and 1, 3, 5, 9, and 10 have not been copied from the S1. They are also copied as they appear in (b). Finally, values 3 and 10 are copied to the empty cells in the same order as in the local search output. This local search is shown in Figure 3.

Local Search 2: *Diversification.* Since the amount of diversity may not be sufficient and more parts of the space solution may need to be searched to increase the variety of solutions. In this local search, two cells are randomly selected, and the jobs in those two cells are moved together. According to Figure 4, if the selected cells are cells four and seven, a new solution is obtained according to the original solution.

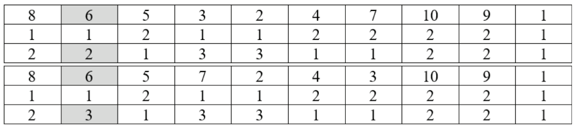
Local Search 3: *Load balancing among factories.* The purpose of this search, as shown in Figure 5, is to move a job from one factory to another in order to change the amount of workload in the factories. For this purpose, a job is selected and its factory is changed randomly. Note that the machine is randomly selected in the new factory.

Local Search 4: *Load balancing among machines.* Unlike the previous local search, here, an attempt was made to change the load on the machines of a factory. For this purpose, according to the Figure 6, a job from a machine in a specific factory should be selected randomly and assigned to another machine from the same factory.



8	6	5	3	2	4	7	10	9	1
1	1	2	1	1	2	2	2	2	1
2	2	1	3	3	1	1	2	2	1
8	6	5	7	2	4	3	10	9	1
1	1	1	1	1	2	2	2	2	1
2	2	1	3	3	1	1	2	2	1

Figure 5. Local Search 3.



8	6	5	3	2	4	7	10	9	1
1	1	2	1	1	2	2	2	2	1
2	2	1	3	3	1	1	2	2	1
8	6	5	7	2	4	3	10	9	1
1	1	2	1	1	2	2	2	2	1
2	3	1	3	3	1	1	2	2	1

Figure 6. Local Search 4.

Table 1. Numerical results for small-size instances.

Factories	Jobs	Machine	GAMS	ICA	HHA
2	2	$(2,2)$	0	0.959484	0
3	3	$(1,2,2)$	0	0.336907	0.00767
2	3	$(2,2)$	0	0.685759	0.174923
3	3	$(1,3,3)$	0	0.683432	0.008876
2	5	$(3,2)$	0	0.235725	0.054173
4	5	$(4,1,3,4)$	0	0.82381	0.026984
3	9	$(3,3,3)$	0	0.562402	0

5. Numerical results

The purpose of this section is to validate the proposed model, so the algorithm is evaluated on three scales. In this regard, the problem is divided into three categories with small, medium, and large sizes. In this study, the proposed hyper-heuristic and ICA were solved ten times for 25 test problems. The average of ten runs is used as a measure of comparison. Furthermore, for small-scale problems, the results of the algorithms are also compared with the results of the GAMS software. To evaluate the performance of the proposed algorithm, Relative Percentage Deviation (RPD) (Eq. (11)) is used.

$$\mathrm{RPD} = \frac{\mathrm{Algorithm}_{\mathrm{solution}} - \mathrm{Minimum}_{\mathrm{solution}}}{\mathrm{Minimum}_{\mathrm{solution}}} * 100. \tag{11}$$

In order to evaluate the performance of the proposed hyper-heuristic algorithm, it is necessary to compare it with another algorithm that has already been used to solve a similar problem. Due to the novelty of the present research model and the lack of research on the problem, one of the most recent research in the field has been selected, which is more similar to the present study and to investigate the performance of the proposed algorithm, it was adapted to the conditions of our problem. For this purpose, the ICA that was developed by Lei *et al.* [14] was selected.

The numerical results of solving the small-size instances by GAMS, the ICA and HHA are reported along in Table 1.

Table 2 also demonstrates the results obtained from the ICA and HHA in large-size instances.

To verify the statistical validity of the results shown in Tables 1, 2 and confirm which is the best algorithm between ICA and HHA, a Kruskal–Wallis test as a non-parametric method has been performed in which the different algorithms as a factor and the response variable as an RPD value are considered. Note that the

Table 2. Numerical results for medium and large-size instances.

Instances	Factories	Jobs	Machine	ICA	HHA
Medium-size	2	25	$(3,3)$	0.59612763	0
	3	25	$(3,3,3)$	0.495676343	0
	5	25	$(3,3,3,3,3)$	0	0.442760063
	5	25	$(2,3,2,3,2)$	0.400764575	0
	5	25	$(5,5,5,5,5)$	0	0.0739775519
	2	50	$(3,3)$	0.422704076	0
	3	50	$(3,3)$	0.3062315	0
	5	50	$(5,5,5,5,5)$	0.67737229	0
	5	50	$(2,2,3,3)$	0.7023026	0
Large-size instances	2	100	$(3,3)$	0.35550039	0
	5	100	$(2,3,2,3,2)$	0.3373313	0
	5	100	$(3,3,3,3,3)$	0.25428843	0
	5	100	$(5,5,5,5,5)$	0	0.1217753
	5	200	$(5,5,5,5,5)$	0.4363601	0
	2	200	$(5,5)$	0.40654415	0
	3	200	$(3,3,3)$	0	0.002131
	5	200	$(3,3,3,3,3)$	0.44305574	0
	5	200	$(2,3,3,2,3)$	0	0.0806031

Table 3. Kruskal–Wallis ANOVA table for methods For ICA and HHA with the RPD index.

Source	SS	df	MS	Chi-sq	Prob Chi-sq
(a) Small-size instances					
Columns	171.5	1	171.	9.82	0.0017
Error	55.5	12	4.625		
Total	227	13			
(b) Medium-size instances					
Columns	168.056	1	168.056	6.73	0.0095
Error	256.444	16	16.028		
Total	424.5	17			
(c) Large-size instances					
Columns	112.5	1	112.5	4.51	0.0338
Error	312	16	19.5		
Total	424.5	17			

parametric equivalent of the Kruskal–Wallis test is the one-way analysis of variance (ANOVA). The obtained results are shown in Table 3 and Figure 7.

Based on the analysis, in comparing the proposed algorithm with the ICA in small, medium and large-size instances, it can be concluded that there is a significant difference between the results and in all case, the HHA was remarkably better. Also, Figure 7 shows that the dispersion of the results obtained in the ICA is much higher than the proposed algorithm, and the HHA is more stable in finding solutions with a lower mean in all cases.

5.1. Analysis of results based on the number of factories

In this section, problems are categorized according to the number of factories in different groups. Then, the average RPD is considered for each category.

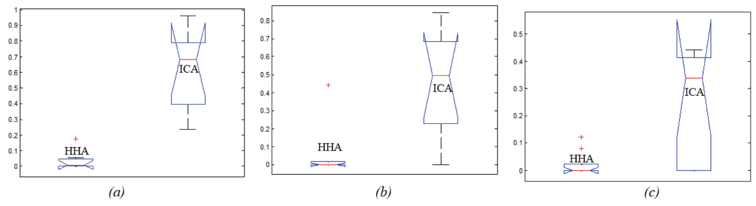


FIGURE 7. Kruskal–Wallis chart for methods For ICA and HHA with the RPD index. (a) Small-size instances. (b) Medium-size instances. (c) Large-size instances.

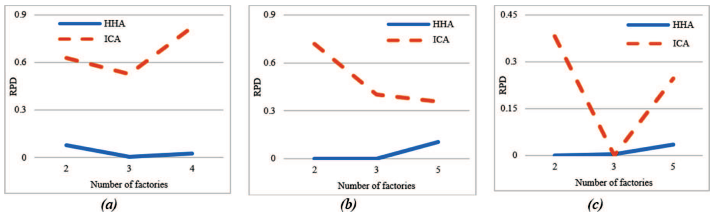


FIGURE 8. The interaction of the number of factories on algorithms. (a) Small-size instances. (b) Medium-size instances. (c) Large-size instances.

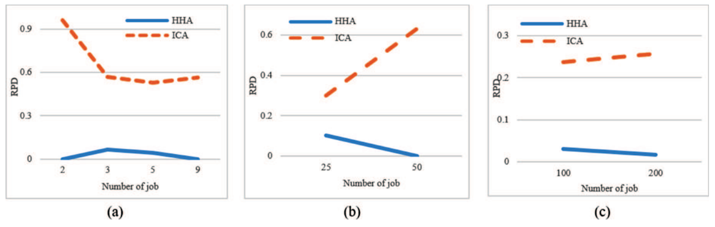


FIGURE 9. The interaction of the number of jobs on algorithms. (a) Small-size instances. (b) Medium-size instances. (c) Large-size instances.

As shown in Figure 8, in small-size instances, as the number of factories increases, the performance of the proposed hyper-heuristic algorithm is better than the ICA and provides better RPDs. In medium-size instances, the performance of both algorithms is almost the same as in small-size instances, and again the hyper-heuristic algorithm works better. In large-size instances, there is no longer a steady decline in the RPD with an increase in the number of factories, but the efficiency of the hyper-heuristic algorithm is still better than the ICA.

5.2. Analysis of results based on the number of jobs

In this subsection, the problems were categorized according to the number of jobs in each size, and the RPD of each category was considered as an indicator for comparison.

As can be seen in Figure 9, the RPD decreases with the increase in the number of jobs in all three sizes of the hyper-heuristic, and the efficiency of the proposed algorithm is still better than the ICA. It should be noted that as the number of jobs increases, the efficiency of the ICA weakens and becomes somewhat entangled in local optimization, and the algorithm is unable to achieve a relatively good solution.

6. Conclusion and future research

In the present study, the problem of scheduling unrelated parallel machines was examined for the first time in a multi-factory environment. Based on a base model, an attempt was made to model the problem, but since the base model was not suitable for the problem under consideration, for the first time, a combined model of two types of conventional modeling (modeling based on sequence and allocation) was presented. The model presented for the problem in small sizes was solved by the GAMS software and the optimal solution was reached. Due to the extrem complexity of the problem, solving the proposed model was not able to find the optimal solution for the medium and large-size instances. Therefore, a hyper-heuristic algorithm (HHA) was designed to solve the real-size instances in a reasonable time. In the proposed algorithm four low-level heuristics were designed that improved the exploitation and exploration of the search strategy. To evaluate the efficiency of the HHA, its results were compared with the results of an imperialist competitive algorithm (ICA). The obtained results showed that in all instances, the HHA performed better than the ICA. In addition to this, the analysis showed the significance of the difference between the algorithms, so that in all cases, *i.e.*, small, medium, and large-size instances, it was the HHA that obtained a better average with less variance, which showed the reliability of the results. The efficiency of the ICA weakens when the size of the problem increases, and it traps in local optima and is unable to improve the initial solutions, whereas this does not apply to the hyper-heuristic algorithm.

Considering the ever-increasing competition in today's changing global market and the fact that the structure of organizations has changed from centralized to distributed in many decision-making areas, the results of this research can lead to the improvement of the organization's competitiveness along with a better understanding of market needs (by getting closer to customers through improvement in geographic distribution and information collection) and the possibility of more timely and stable supply (with the possibility of production in alternative factories in case of failure in one of the members of the production network). It is obvious that, like any other research, there are many limitations, such as considering classical functions, solving a single-objective problem, and not paying attention to the real-world constraints of the scheduling problem in the current research, which their investigation can be interesting for the continuation of this research. Therefore, several research areas can be considered as suggestions for future research. As striking future research, the problem can be practically examined by implementing the suggested approach in real cases. To this end, utilizing some data-driven techniques will not be far from the mind. Adding more constraints, including constraints on access to machines, machines breakdown, and etc., are also interesting issues for further extensions. The use of the model with the assumption of uncertainty can be considered for future studies. Developing other solving algorithms, *e.g.*, the combination of meta-heuristic algorithms and exact methods, is one of the open fields for upcoming research. Besides, the novel objective functions, such as social responsibilities and environmental issues can also be investigated.

Acknowledgements. We thank the anonymous reviewers for their careful reading of our manuscript and their many insightful comments and suggestions.

References

[1] V. Abdollahzadeh, I. Nakhaikamalabadi, S.M. Hajimolana and S.H. Zegordi, A multifactory integrated production and distribution scheduling problem with parallel machines and immediate shipments solved by improved whale optimization algorithm. *Complexity* **2018** (2018). DOI: 10.1155/2018/5120640.

[2] J. Acevedo-Chedid, J. Grice-Reyes, H. Ospina-Mateus, K. Salas-Navarro, A. Santander-Mercado and S.S. Sana, Soft-computing approaches for rescheduling problems in a manufacturing industry. *RAIRO: Oper. Res.* **55** (2021) S2125–S2159.

[3] N. Bagheri Rad and J. Behnamian, Recent trends in distributed production network scheduling problem. *Artif. Intell. Rev.* **55** (2022) 2945–2995.

[4] B. Behdani, Z. Lukszo, A. Adhitya and R. Srinivasan, Decentralized vs. centralized management of abnormal situations in a multi-plant enterprise using an agent-based approach, in Computer Aided Chemical Engineering. Vol. 28. Elsevier (2010) 1219–1224.

[5] J. Behnamian, Decomposition based hybrid VNS–TS algorithm for distributed parallel factories scheduling with virtual corporation. *Comput. Oper. Res.* **52** (2014) 181–191.

[6] J. Behnamian, Multi-objective production network scheduling using sub-population genetic algorithm and elastic method. *J. Ind. Eng. Res. Prod. Syst.* **3** (2016) 133–147.

[7] J. Behnamian and S.M.T. Fatemi Ghomi, The heterogeneous multi-factory production network scheduling with adaptive communication policy and parallel machine. *Inf. Sci.* **219** (2013) 181–196.

[8] J. Behnamian and S.M.T. Fatemi Ghomi, A survey of multi-factory scheduling. *J. Intell. Manuf.* **27** (2016) 231–249.

[9] J. Behnamian and S.M.T. Fatemi Ghomi, Multi-objective multi-factory scheduling. *RAIRO: Oper. Res.* **55** (2021) S1447–S1467.

[10] P. Cowling and E. Soubeiga, Neighborhood structures for personnel scheduling: a summit meeting scheduling problem, in Proceedings of the 3rd International Conference on the Practice and Theory of Automated Timetabling, edited by E.K. Burke and W. Erben (2000) 277.

[11] A. Delgoshaei and C. Gomes, A multi-layer perceptron for scheduling cellular manufacturing systems in the presence of unreliable machines and uncertain cost. *Appl. Soft Comput.* **49** (2016) 27–55.

[12] A. Gharaei and F. Jolai, A multi-agent approach to the integrated production scheduling and distribution problem in multi-factory supply chain. *Appl. Soft Comput.* **65** (2018) 577–589.

[13] S. Giordani, M. Lujak and F. Martinelli, A decentralized scheduling policy for a dynamically reconfigurable production system, in International Conference on Industrial Applications of Holonic and Multi-Agent Systems. Springer (2009) 102–113.

[14] D. Lei and M. Liu, An artificial bee colony with division for distributed unrelated parallel machine scheduling with preventive maintenance. *Comput. Ind. Eng.* **141** (2020) 106320.

[15] D. Lei, Y. Yuan, J. Cai and D. Bai, An imperialist competitive algorithm with memory for distributed unrelated parallel machines scheduling. *Int. J. Prod. Res.* **58** (2020) 597–614.

[16] J. Lin, Backtracking search based hyper-heuristic for the flexible job-shop scheduling problem with fuzzy processing time. *Eng. App. Artif. Intell.* **77** (2019) 186–196.

[17] J. Lin, Z.-J. Wang and X. Li, A backtracking search hyper-heuristic for the distributed assembly flow-shop scheduling problem. *Swarm Evol. Comput.* **36** (2017) 124–135.

[18] C. Lu, Q. Liu, B. Zhang and L. Yin, A pareto-based hybrid iterated greedy algorithm for energy-efficient scheduling of distributed hybrid flowshop. *Expert Syst. App.* **204** (2022) 117555.

[19] Q. Luo, Q. Deng, G. Gong, X. Guo and X. Liu, A distributed flexible job shop scheduling problem considering worker arrangement using an improved memetic algorithm. *Expert Syst. App.* **207** (2022) 117984.

[20] J.-Y. Mao, Q.-K. Pan, Z.-H. Miao, L. Gao and S. Chen, A hash map-based memetic algorithm for the distributed permutation flowshop scheduling problem with preventive maintenance to minimize total flowtime. *Knowl.-Based Syst.* **242** (2022) 108413.

[21] J. Mou, P. Duan, L. Gao, X. Liu and J. Li, An effective hybrid collaborative algorithm for energy-efficient distributed permutation flow-shop inverse scheduling. *Future Gener. Comput. Syst.* **128** (2022) 521–537.

[22] J. Park, Y. Mei, S. Nguyen, G. Chen and M. Zhang, An investigation of ensemble combination schemes for genetic programming based hyper-heuristic approaches to dynamic job shop scheduling. *Appl. Soft Comput.* **63** (2018) 72–86.

[23] A. Saadi, P. Fattahi, D. Rahmanii, S.M. Hassan Hosseini and S.S. Sana, An exact method for job sequencing in a mixed-model assembly line considering feeding lines based on the benders decomposition approach. *Int. J. Manage. Sci. Eng. Manage.* (2022) 1–14.

[24] S.S. Sana, H. Ospina-Mateus, F.G. Arrieta and J.A. Chedid, Application of genetic algorithm to job scheduling under ergonomic constraints in manufacturing industry. *J. Ambient Intell. Humanized Comput.* **10** (2019) 2063–2090.

[25] Y. Sang and J. Tan, Intelligent factory many-objective distributed flexible job shop collaborative scheduling method. *Comput. Ind. Eng.* **164** (2022) 107884.

[26] S. Schulz, M. Schönheit and J.S. Neufeld, Multi-objective carbon-efficient scheduling in distributed permutation flow shops under consideration of transportation efforts. *J. Cleaner Prod.* **365** (2022) 132551.

[27] W. Shao, Z. Shao and D. Pi, Multi-local search-based general variable neighborhood search for distributed flow shop scheduling in heterogeneous multi-factories. *Appl. Soft Comput.* **125** (2022) 109138.

[28] A. Soares, A. Azevedo and J.P. De Sousa, Distributed planning and control systems for the virtual enterprise: organizational requirements and development life-cycle. *J. Intell. Manuf.* **11** (2000) 253–270.

[29] G. Wang, X. Li, L. Gao and P. Li, An effective multi-objective whale swarm algorithm for energy-efficient scheduling of distributed welding flow shop. *Ann. Oper. Res.* **310** (2022) 223–255.

[30] J.F. Williams, Heuristic techniques for simultaneous scheduling of production and distribution in multi-echelon structures: theory and empirical comparisons. *Manage. Sci.* **27** (1981) 336–352.

[31] H. Yahyaoui, S. Krichen, B. Derbel and E.-G. Talbi, A hybrid ILS-VND based hyper-heuristic for permutation flowshop scheduling problem. *Proc. Comput. Sci.* **60** (2015) 632–641.

[32] F. Zhao, X. Hu, L. Wang and Z. Li, A memetic discrete differential evolution algorithm for the distributed permutation flow shop scheduling problem. *Complex Intell. Syst.* **8** (2022) 141–161.

[33] Y. Zhou and J.-J. Yang, Automatic design of scheduling policies for dynamic flexible job shop scheduling by multi-objective genetic programming based hyper-heuristic. *Proc. CIRP* **79** (2019) 439–444.