Solving multi-objective Modified Distributed Parallel Machine and Assembly Scheduling Problem (MDPMASP) with eligibility constraints using metaheuristics

Ikhlasul Amallynda and Budi Santosa

ABSTRACT
We present a new generalization and metaheuristics for the distributed parallel machine and assembly scheduling problem (DPMASP), namely MDPMASP with eligibility constraints. There is a set of unrelated factories or production lines. Each has a series of non-identical parallel machines with varying processing speeds. Then, it was disposed of as a single assembly machine in a series. Each product is assembled from a set of components (jobs). Each item necessitates multiple unidentical jobs. The objectives are to minimize mean flow time and the number of tardy jobs. We suggest four basic heuristics and three metaheuristics to tackle the problem. The Taguchi approach is used to discuss and calibrate various metaheuristic parameters. Algorithms are compared using the four performance measures. The computational results show that the proposed algorithms can solve moderate-sized problems efficiently and near-optimal solutions in most cases. Moreover, based on the four performance measures, MGA is the best method, followed by MSA, MPSO, SH2, SH4, SH1 and SH3.

1. Introduction

The global economy has undergone radical changes in the last two decades. This is influenced by the growth of global trade and explosive international competition. Globalization has encouraged natural selection, and the strongest will survive. Market success will be obtained by companies that are able to respond quickly to customer needs and desires. One way to respond to these conditions is to increase the variety of products offered. A diversified product portfolio and diversified offering are the company’s main assets.

Providing multiple attributes in a product is an important way to attract customers, but it often increases the complexity of planning and scheduling processes. In the automotive industry sector, where there is variation (variation offered to customers) and internal variation (variation involved in making the product). Forecasting and dealing with internal variations is a challenge for companies to build products to order. The effectiveness of strategies to mitigate negative effects, such as modularity,
mutability, late configuration, and option bundling, depends on the order fulfillment strategy the company follows. Several kinds of manufacturing production strategies such as: make to order or make to assembly can reduce these negative effects.

Suppliers play a very important role at every stage of the product life cycle. From sourcing raw materials to helping increase production, and to finding better raw material options when the market gets saturated, companies need to work closely with their suppliers to get the best out of their products. Automotive suppliers are defined as companies that produce goods that are used in the production process of cars or become parts of cars, so they supply these goods directly or indirectly to car manufacturers, which is identical to Original Equipment Manufacturer (OEM). These items can be individual components, such as screws, or whole assemblies, such as pre-assembled door modules. Therefore, automotive suppliers are part of the automotive supply chain.

Production scheduling has become a must for manufacturing operations looking to take their production facilities to the next level. Production scheduling refers to the allocation of resources, operations, and processes needed to make goods and services. The purpose of a production schedule is to balance customer requirements with available resources while operating in a cost-effective manner. If the production schedule is inaccurate and not feasible with the available resources, we will have problems getting the goods produced and delivered to the customer on time. In the automotive industry sector, production scheduling is crucial and difficult to do due to the high variety of products and components, each component must be produced by a different supplier. However, it is possible that a supplier is able to produce more than one component. Therefore, a scheduling problem arises called the Distributed Parallel Machine and Assembly Scheduling Problem (DPMASP) with eligibility constraints (Sara Hatami, Ruiz, & Romano, 2015).

The DPMASP with eligibility constraints, as first described by Hatami et al. (2015), is to optimize the processing of \( n \) jobs in a series of two manufacturing stages, namely, the production stage and assembly stage. The production stage consists of a set \( F \) of \( f \) identical factories with a set \( M \) of \( m \) unrelated parallel machines at one factory. Besides, the eligibility constraints are considered in this problem. The eligibility constraints indicate some conditions where job \( j \) can only be assigned to an eligible factory. In the second stage, a single assembly machine is assigned to assemble all the final products. All factories are identical and have the same number of machines.

This article discusses an expansion of the DPMASP to an unrelated multi-factory manufacturing with an unidentical parallel machine scenario. A set of products were manufactured through an assembly of several jobs, each fabricated in the PFSP factories of the production stage. A product could be assembled at the single machine assembly stage only after all the jobs in the production stage are completed. Moreover, each factory has several unidentical parallel machines with different specifications and capacities that influence processing speed. Therefore, the assumption of an identical parallel machine in DPMASP is no longer valid. In conclusion, to make the DPMASP more realistic to implement on a more complex system, some assumptions about identical parallel machines must be changed.

The previous study on the distributed assembly scheduling problem only considers the optimization of a single objective, i.e. minimizing the makespan (Duan et al., 2016; S Hatami et al., 2015; Pan, Gao, Xin-Yu et al., 2019; Shao et al., 2019; K. Wang et al., 2017). Thus, the
concept of optimality for a single objective is meaningless since the best solution for one objective may not be the best for another (Glover & Sörensen, 2015). Besides, most studies present MILP and local search metaheuristics to solve the distributed assembly flow shop scheduling problem and its variants (Framinan et al., 2019). Local search metaheuristics iteratively make small changes to a single solution. Constructive metaheuristics construct solutions from their constituting parts. Population-based metaheuristics iteratively combine solutions into new ones. However, these classes are not mutually exclusive, and many metaheuristic algorithms combine ideas from different classes. Such methods are called hybrid metaheuristics (Glover & Sörensen, 2015).

As a result, we analyze the Modified Distributed Parallel Machine and Assembly Flowshop Scheduling Problem, abbreviated as MDPMASP, with eligibility constraints. Two manufacturing stages are considered: production and assembly. There is a set of $F$ distributed unrelated factories with a set $M$ of $m$ unidentical parallel machines in each factory as a shop configuration. In contrast, the assembly stage consists of a single assembly machine referred to as $M_A$. There are four interrelated decisions to be made to solve this problem. These decisions generate an assembly product sequence as the initial solution, complete the factory selection for each job, schedule within each factory, and assign jobs to one of the machines set at the specific factory. Additional details will be described further. In addition, we consider the mean flow time and the number of tardy products as optimal criteria. To our best knowledge, this is the first attempt at resolving such a critical scheduling issue. The main contributions are constructing a new multi-objective scheduling problem and developing several algorithms that were often implemented in simple cases previously. As regards the computational complexity, we may deduce that the Multi-Objective MDPMASP with eligibility constraints is an NP-hard problem because the standard parallel machine problem (even when two identical machines are used, i.e. the P2/Cmax problem) is already NP-Hard according to Lenstra et al. (1977).

In this paper, we proposed four simple heuristic algorithms and three high-performing population-based algorithms such as genetic algorithm (GA), simulated annealing (SA), and particle swarm optimization (PSO) to solve the multi-objective MDPMASP with eligibility constraints. One of the population-based approaches’ advantages was being able to explore multiple regions of the fitness landscape concurrently, thus mitigating search bias in the initial position or sampling process. They also provide a good balance between exploration and exploitation in the search landscape (Prugel-Bennett, 2010). In addition, population-based algorithms allow obtaining better (feasible) solutions that do not dominate each other in a shorter time than local search metaheuristics. Although it has a slightly more complicated procedure. Significant problem-specific knowledge and accelerations are investigated in order to improve the algorithms’ effectiveness. The small size problems have been solved using MILP and total enumeration sampling to validate the proposed algorithms. Four performance metrics, i.e. overall non-dominated vector generation (ONVG), overall number of non-dominated solutions (ONSN), average quality (AQ), and distance metric ($D_{IR}$) proposed by Qian et al. (2009) are used to measure the algorithms’ performance. It was done because the multi-objective problems result in multiple non-dominated solutions.
The rest of our paper is organized as follows: Section 2 is a literature review, and the problem definition is explained in Section 3. In Section 4, we present a Mixed Integer Linear Programming (MILP) model to solve the considered problem. In Section 5, four simple heuristics and three metaheuristic algorithms are proposed to solve the problem. Section 6 presents a comprehensive computational evaluation of the proposed algorithms. Finally, some concluding remarks and future research directions are provided in Section 9, which shows the computational results.

2. Literature review

Due to the present economic trend toward customized manufacturing, assembly production is becoming more prevalent. Typically, an assembly floor is divided into two stages: production and assembly (Hatami et al., 2015). Assembly Scheduling Problems (ASPs) are gaining increasing attention from academics and practitioners as a subset of scheduling decisions on which parts/components/subsets of products or services must be manufactured in parallel and then assembled in a final stage (Framinan & Perez-Gonzalez, 2017). Among ASPs, the Assembly Flowshop Scheduling Problem (AFSP) is a prevalent type that occurs in a wide variety of production systems (S.-Y. Wang & Wang, 2015). The layout for this problem includes a flowshop for part processing and a single assembly machine for product processing. Lee et al. (1993) were the first to solve the AFSP problem, followed by Yokoyama (2001), Yokoyama and Santos (2005), Sung and Juhn (2009), and Sheikh et al. (2018), and Framinan et al. (2019), and numerous others. AFSP is used in a variety of industries, including engine assembly (Lee et al., 1993) and personal computer manufacture (Hariri & Potts, 1997), to mention a few.

Numerous difficulties confront the industrial sector in reconciling manufacturing complexity and customer desires. In an uncertain and volatile global market, several organizations must quickly grow their product portfolio while maintaining a shorter product life cycle (Elmaraghy et al., 2013). To address these difficulties, diverse firms are required to collaborate to assist in manufacturing and supply chain activities. Unlike the AFSP assumption that all goods and components are processed in a single manufacturing facility. Due to the need for distributed manufacturing in today’s fractured and globalized economy, large corporations frequently operate many production centers or factories (Moon et al., 2002). Managers must decide how to distribute jobs to manufacturers and how to properly schedule jobs inside each facility to remain competitive in continually changing markets. Thus, the so-called Distributed Scheduling Problem (DSP), which concerns the assignment and scheduling of jobs across multiple factories, DSP has become a popular study area (Fernandez-Viagas & Framinan, 2015; Bahman Bahman Naderi & Ruiz, 2014). Collaboration with other factories has resulted in numerous benefits, including increased adaptability and resilience, increased product variety and quality, decreased manufacturing costs, and less risk.

For instance, B. Naderi and Ruiz (2010) study scheduling issues in a distributed permutation flowshop (DFSP) composed of a collection of similar factories, each of which is a permutation flowshop. Ruiz et al. (2019) describe an iterated greedy algorithm for the DFSP that greatly improves results. (Pan, Gao, Xin-Yu, et al., 2019) discuss the DFSP with the total flowtime requirement and offer three successful constructive heuristics as well as four meta-heuristics. Li et al. (2019) offer an
improved artificial bee colony algorithm in a prefabricated system for the DFSP with distance coefficient. Hatami et al. (2013) propose a hybrid DFSP-assembly line with a distributed flowshop as the first stage and a single assembly machine as the second stage. Xiong et al. (2014) use distributed manufacturing to solve a two-stage assembly flowshop problem.

To the extent of our knowledge, the DAPFSP was introduced by Hatami et al. (2013) for the first time, taking into the eligibility constraints. In DAPFSP, there are two different stages. The first stage consists of a set of identical factories (all factories are capable of processing all components and each consists of a permutation flowshop). The second stage entails the establishment of an assembly facility. The assembly factory assembles all of the components into finished goods. The idea is to keep the makespan as short as possible. To solve this problem, Hatami et al. (2013) proposed an MILP model, three constructive heuristics, and a VND algorithm. Zhang et al. (2020) construct a genetic algorithm (GA) with an enhanced crossover strategy and three local search schemas that outperform Hatami’s constructive heuristics. Later, an estimation of the distribution algorithm-based memetic algorithm (EDA-MA) with a bi-vector-based representation was investigated by Wang and Wang (2015) to improve the potential solution. Lin and Zhang (2016) present a hybrid biogeography-based optimization (HBBO) algorithm that uses a path-relinking heuristic as an approach for product local search strategy and an insertion-based heuristic to optimize part permutations. Lin et al. (2017) propose a Backtracking Search Hyper-Heuristic (BSHH) consisting of 10 low-level heuristics to perform operations on the solution space and a backtracking search to modify the heuristics at the lowest level. In addition, Hatami et al. (2015) present the DPMASP with eligibility constraints. DPMASP consists of the distributed unrelated factories in the first stage and the single assembly machine in the second stage. In the first stage, there is a set of unrelated factories with identical parallel machines in the production stage. In this case, parallel machines within a factory have the same specifications and production capacity. It is shown by the same job processing times on parallel machines at the specific factory. Jobs have to be assigned to factories and to machines. Finished jobs in the first stage are assembled into final products in the second stage. Due to eligible constraints, machines cannot be idle, and some jobs can be processed only in certain factories. To solve this problem, Hatami et al. (2015) proposed a mathematical model and two high-performing heuristics to minimize the makespan in the assembly stage. There are three assumptions made in DPMASP with eligibility constraints. First, the assembly stage is separated from the assembly operation at each factory. Second, allow the different jobs that compose a product to be produced in different factories. Third, each product might have a number of jobs (components) different from m. The problem studied by Hatami et al. (2015) is different from our proposed problem, i.e. MDPMASP with eligibility constraints. Here, we consider the use of non-identical parallel machines with varying specifications and production capacity. This increases the problem’s complexity and difficulty in solving it.
To the best of our knowledge, no further literature exists on DPMASP with eligibility constraints since 2019, so this is the first effort that considers the non-identical parallel machine in a distributed manufacturing setting. Moreover, we present four simple heuristics and three metaheuristic algorithms to minimize the mean flow time and number of tardy products in MDPMASP with eligibility constraints.

3. Problem definition

The modified distributed parallel machine and assembly scheduling problem (MDPMASP) with eligibility constraints is now described in detail. This problem consists of two manufacturing stages: production and assembly. Both stages are assigned to produce a set $T$ of $t$ different products manufactured through an assembly process of a set $N$ of $n$ jobs in series according to the requested demand during the planning horizon. There is a set $F$ of $f$ distributed non-identical factories with a set $M$ of $m$ unrelated - parallel machines in each one. In the assembly stage, there is a single assembly facility denoted as MA. Furthermore, eligibility constraints are considered and represented by $LF_j$, $LF_j \subseteq F$, which is the subset of factories where job $j$ can be assigned, where $f \geq |LF_j| \geq 1$, $j = 1, \ldots, n$, and job $j$ can only be processed in the eligible factory.

The related job is assembled into a product by the assembly program of each product $N_s, N_s \subseteq N$. Therefore, product $s$ consists of $|N_s|$ jobs. Each job is associated with a single assembly program for a specific product and $\sum |N_s| = n$, where $s = 1, 2, \ldots, t$. A product can be assembled at a single assembly machine after all jobs in $N_s$ have been completed in the first stage. For the first stage, $PT_{jqi}$ denotes the production time of the job $j$ at a machine $i$ of factory $q$. The second stage is to set $AT_s$ to be the assembly time of product $s$. Note that all factories are non-identical and have the same number of machines. All processing times are positive, known integer quantities, and deterministic or random within a certain range.

Each product requires two or more jobs. Jobs are different in kinds and batch sizes. At first, all jobs are processed in the production stage, including several unrelated parallel machines with different specifications, capacity, and speed for processing. All machines in the specific factory can process all jobs according to $LF_j$, and each machine can also process jobs in the specific factory. Still, they would be processed by exactly one machine at a time. Subsequently, all jobs are disposed to the second stage and are processed by the MA machine.

A simple problem is used to illustrate the proposed algorithms. It consists of fifteen products ($t = 15$), six jobs ($n = 6$), and three different factories ($f = 3$) with two unrelated parallel machines ($m = 2$) in each factory. Each machine in the factory runs at a different speed, resulting in differences in the completion time of jobs on a machine. Every product $s$ is composed of several jobs that exist in the related assembly program $N_s$. The assembly program of the six products are $N_1 = [9,10], N_2 = [7,12,13,14], N_3 = [8,15], N_4 = [6], N_5 = [5], \text{and } N_6 = [1,2,3,4,11]$, meaning job 9 and job 10 must be finished in order to assemble product 1.

There are three kinds of sequences: a product sequence ($\pi$), a job to factory sequence ($\pi_{Fq}, \forall q$), and a job to factory machine sequence ($\pi_{FMqi}, \forall q, i$). In this example, there are several data sets of each sequence, $\pi = \{2,6,1,4,5,3\}, \pi_{F1} = \{14,12,13,1,4,8\}; \pi_{F2}$
= \{11,2,6,5,15\}; \pi_{FM31} = \{7,9\}; \pi_{FM32} = \{3,10\}. While all jobs in the assembly program of a product s are completed in the production stage, it can be assembled in the assembly stage. The earliest assembling time of product s is denoted as \(E_s\). Figure 1 shows the problem with three factories, two machines, 15 jobs, and a single assembly machine.

The purpose of MDPMASP with eligibility constraints is to assign jobs to one machine at a specific factory, schedule all assigned jobs in each machine at each factory, and schedule all products in the single assembly machine while minimizing the mean flow time and the number of tardy products simultaneously. Due to the computational complexity of the problem, it is an NP-Hard problem. The MDPMASP with eligibility constraints is an essential generalization of existing problems and has not been studied before to the best of our knowledge,

\[
F = \frac{1}{t} \sum_{t=1}^{t} F_s = \frac{1}{t} \sum_{t=1}^{t} E_s + AT_s
\]

\[
E_s = \max_{j^* \in N_s} \{C_{j^*}\}
\]

where \(F\) is the mean flow time; \(E_s\) is the maximum completion time of job \(j^*\) \((C_{j^*}, j^* \in N_s)\), and it is equal to the starting time of product \(s\) in an assembly stage; \(AT_s\) is the time of product \(s\) at the assembly stage. The number of tardy products \((n_T)\) can be calculated by

\[
n_T = \sum_{s=1}^{t} n_T^s \text{ where } n_T^s = \begin{cases} 1, & \text{if } (c_s - DD_s) > 0 \\ 0, & \text{otherwise} \end{cases}
\]

\(n_T\) denotes the number of tardy products; \(C_s\) denotes the maximum completion time of a product \(s\); \(DD_s\) denotes the product’s due date. Four simple heuristics and three metaheuristic algorithms are proposed to find efficient schedules that minimize the mean flow time and the number of tardy products on the MDPMASP with eligibility constraints system.

**Figure 1.** Representative Solution of MDPMASP with eligibility constraints.
The mathematical model for this problem was developed based on (Sara Hatami et al., 2015). The distinction was that, in this model, the factories are not identical, and the machines in each factory are unrelated. The machines in a factory have different processing times. As a result, we need to add an index $q$ to the production time variable $PT_{jk}$, so that it becomes $PT_{jkq}$. A mixed-integer linear programming model for the Multi-Objective Modified Distributed Parallel Machine and Assembly Scheduling Problem (MDPMASP) with eligibility constraints is explained as follows. This model was developed based on

| Index | Description |
|-------|-------------|
| $i,j$ | Jobs, $i,j = 0, 1, \ldots, n$, where 0 is a dummy job |
| $k$ | Machines, $k = 1, \ldots, m$ |
| $q$ | Factories, $q = 1, \ldots, f$ |
| $l,s$ | Products, $l,s = 0, 1, \ldots, t$, where 0 is a dummy product |
| $M$ | A sufficiently large positive number |

| Parameter | Description |
|-----------|-------------|
| $n$ | Number of jobs |
| $m$ | Number of machines |
| $f$ | Number of factories |
| $t$ | Number of products |
| $PT_{jk}$ | Production time of job $j$ on machine $k$ in factory $q$ |
| $AT_s$ | Assembly time of product $s$ at the assembly stage |
| $DD_s$ | Due date of product $s$ at the assembly stage |
| $G_p$ | Binary parameter equal to 1 if job $j$ belongs to product $s$, and 0 otherwise |
| $LF_j$ | A subset of factories where job $j$ can be assigned |

| Variable | Description |
|----------|-------------|
| $X_{ijkq}$ | Binary variable equal to 1 if job $i$ is an immediate predecessor of job $j$ on machine $k$ in factory $q$, and 0 otherwise, where $j \neq k$ |
| $Y_{sl}$ | Binary variable equal to 1 if product $l$ is an immediate predecessor of product $s$ at the assembly stage, and 0 otherwise, where $s \neq l$ |
| $CP_j$ | Completion time of job $j$ at the production stage |
| $CA_s$ | Completion time of product $s$ at the assembly stage |
| $\bar{F}$ | Mean flow time |
| $n_T$ | Number of tardy product |

The objective function of the model is to minimize the mean flow time and the number of tardy products, respectively,

$$Min \bar{F}$$

$$\bar{F} = \frac{1}{t} \sum_{s=1}^{t} F_s$$

$$F_s = CA_s = \max_{1 \leq j \leq n} \{CP_j, G_p\} + AT_s$$

$$CP_j = CP_i + PT_{ijkq}, \forall i, j \neq i, k, q \in LF_i \cap LF_j$$
Min \( n_T \) \hspace{1cm} (8)

\[
n_T = \sum_{s=1}^{n} \delta(T_s), \delta(x) = 1 \text{ if } x > 0, \text{ and } \delta(x) = 10 \text{ otherwise}
\] \hspace{1cm} (9)

\[
T_s = \max_{1 \leq s \leq l} \{0, L_s\}
\] \hspace{1cm} (10)

\[
L_s = CA_s - DD_s
\] \hspace{1cm} (11)

\[
CA_s = \max_{1 \leq j \leq n} \{CP_j + G_j\} + AT_s
\] \hspace{1cm} (12)

\[
CP_j = CP_i + PT_{j,k} \cdot X_{j,k}, \quad \forall i, j \neq i, q, k \in LF_i \cap LF_j.
\] \hspace{1cm} (13)

The constraints of the model are as follows:

(1) Control and ensure that every job must be processed on exactly on one machine at a certain factory and must have exactly one preceding and succeeding job,

\[
\sum_{i=0, i \neq j}^{n} \sum_{i=1}^{m} \sum_{q=1}^{f} X_{ijqk} = 1 \quad \forall j,
\] \hspace{1cm} (14)

\[
\sum_{j=0, j \neq i}^{n} \sum_{i=1}^{m} \sum_{q=1}^{f} X_{ijqk} = 1 \quad \forall i.
\] \hspace{1cm} (15)

(2) State that each machine at each factory has to have a dummy job 0 as a predecessor and successor, respectively.

\[
\sum_{j=1}^{n} X_{0j,q,k} \forall q, k, \quad q \in LF_j
\] \hspace{1cm} (16)

\[
\sum_{i=1}^{n} X_{i0,q,k} \forall q, k, \quad q \in LF_i
\] \hspace{1cm} (17)

(3) Determine that if a job is sequenced on a machine, then its predecessor and successor must be processed on the same machine.

\[
\sum_{j=1}^{n} 1, j \neq i (X_{ij,q,k} - X_{ij,q,k}) = 0 \quad \forall i, q, k.
\] \hspace{1cm} (18)

(4) Avoid the occurrence of cross-precedences, which means that a job cannot be both a predecessor and a successor of another job at the same time,
\[
\sum_{k=1}^{m} \sum_{q=1}^{f} X_{ijk} + \sum_{k=1}^{m} \sum_{q=1}^{f} X_{ijqk} \leq 1 \quad \forall i \in \{1, \ldots, n - 1\}, j > 1.
\] (19)

(5) Specify that if a job \( j \) is placed immediately after job \( i \), its processing on the machine \( k \) cannot start before the processing of the job \( i \) on the machine \( k \) finishes,

\[
CP_j \geq CP_i + PT_{jqk} + M(1 - X_{ijqk}) \quad \forall i, j, q, k \in LF_i \cap LF_j.
\] (20)

(6) In the assembly stage, each product must have one predecessor and not more than one succeeding product,

\[
\sum_{s=0, s \neq l}^{t} Y_{ls} = 1 \quad \forall s
\] (21)

\[
\sum_{s=0, s \neq l}^{t} Y_{ls} = 1 \quad \forall l.
\] (22)

(7) Prevent a product from being both a predecessor and a successor to another product at the same time,

\[
Y_{ls} + Y_{sl} \leq 1 \quad \forall l \in \{1, \ldots, t - 1\}, s > 1.
\] (23)

(8) Imply that each product cannot begin to assemble before all its jobs in its assembly program are completed in the first stage,

\[
CA_s \geq (C_j, G_{js}) + AT_s \quad \forall j, s.
\] (24)

(9) Determine that if product \( s \) is placed immediately after product \( l \), it cannot start to be assembled on the assembly machine before the assembling of product \( l \) in the assembly machine has finished,

\[
CA_s \geq CA_l + AT_s + M(1 - Y_{ls}) \quad \forall l, s \neq l.
\] (25)

(10) Define the makespan and the domain of decision variables, respectively,

\[
C_{max} \geq CA_s \quad \forall s
\] (26)

\[
X_{ijqk} \in \{0, 1\} \quad \forall i, j \neq i, q, k \in LF_i \cap LF_j
\] (27)

\[
Y_{ls} \in \{0, 1\} \quad \forall l, s \neq l
\] (28)

\[
CP_j \geq 0 \quad \forall j
\] (29)

\[
CA_s \geq 0 \quad \forall s
\] (30)

Note that \( CP_0 = CA_0 = 0 \).
4. Proposed algorithms

MDPMASP with eligibility constraints is an NP-hard problem if \( n > (f \times m) \). To solve large problems, a heuristic and metaheuristic-based solution search technique is required.

4.1. Simple heuristic algorithms

Heuristic methods can be used to find solutions to optimization problems, but they do not guarantee that they always produce optimal solutions. In principle, the heuristic method is used to obtain the best possible solution in a required time span. Heuristic methods can be used to find solutions to optimization problems, but they do not guarantee that they always produce optimal solutions. In principle, the heuristic method is used to obtain the best possible solution in the required time span. We present four simple heuristic algorithms, namely SH-1, SH-2, SH-3, and SH-4. The four algorithms will produce three solution structures, i.e., product sequence (\( \pi \)), job to factory sequences (\( \pi_{F, q} \forall q \)), and job to factory-machine sequence (\( \pi_{F, M, q, k} \forall q, k \)). The goal to be achieved is the simultaneous minimization of the mean flow time and the number of tardy products. In general, a simple heuristic algorithm was developed based on two algorithms developed by Erenay et al. (2010), namely, independent beam search (BS-I) and dependent beam search (BS-D), and the two algorithms presented by B. Naderi and Ruiz (2010), namely, NR1 and NR2. The BS-I and BS-D algorithms are proven to be efficient in minimizing the mean flow time and number of tardy jobs simultaneously on a single machine. Meanwhile, NR1 and NR2 proved efficient in minimizing the makespan in the distributed permutation flow shop scheduling problem (DPFSP). Table 1 shows the structure of the proposed simple heuristic algorithms.

The following is the general procedure of the proposed simple heuristic algorithm: First, the BS-I or BS-D algorithm presented by Erenay et al. (2010) is used to determine the best product sequence (\( \pi \)) based on \( AT_s \) and \( DD_s \). This can reduce the average flow time and the number of late products in a hierarchical machine. After identifying an efficient product sequence, focus should be given to minimizing \( E_s \). The minimum \( E_s \) can be achieved by forming a job to factory sequence and a job to factory-machine sequence with the lowest makespan. Second, weights representing the priority level of each job are required to determine the job to factory sequences (\( \pi_{F, q} \forall q \)) for each product sequence (\( \pi \)). Third, the job to factory-machine sequence (\( \pi_{F, M, q, k} \forall q, k \)) is then built using two job to factory assignment rules described in B. Naderi & Ruiz (2010). The first one, referred to as NR1, assigns job \( j \) to the machine with the lowest current \( C_{max} \), without considering job \( j \). The second rule, NR2, assigns job \( j \) to the factory with the lowest \( C_{max} \) after scheduling job \( j \). Fourth, each product’s earliest assembly time (\( E_s \)), completion time (\( C_s \)), and flow time

| Proposed Simple Heuristic (SH) | Product Sequence (\( \pi \)) | Job to factory sequences (\( \pi_{F, q} \forall q \)) | Job to factory-machine sequences (\( \pi_{F, M, q, k} \forall q, k \)) |
|-------------------------------|-----------------------------|-----------------------------------------------|------------------------------------------------------------|
| SH-1                          | BS-I Algorithm              | Proposed job to the factory assignment        | NR1                                                        |
| SH-2                          | BS-I Algorithm              | procedure                                     | NR2                                                        |
| SH-3                          | BS-D Algorithm              |                                               | NR1                                                        |
| SH-4                          | BS-D Algorithm              |                                               | NR2                                                        |
(Fₚ) are known. Fifth, the mean flow time and the number of tardy products can be computed. The process is applied until all jobs in the assembly program of product s are scheduled. The pseudocode of proposed simple heuristic algorithms is shown in Figure 2. The processing time and weight calculation are listed in the Table 2.

### 4.2. Metaheuristic algorithms

#### 4.2.1. The Modified Genetic Algorithm (MGA)

Genetic algorithm (GA) belongs to the class of evolutionary algorithms based on a population of individuals. GA are numerical optimization algorithms inspired by both natural selection and natural genetics. This method is a general one and capable of being applied to an extremely wide range of problems. GA generally consists of the following three operators: selection, crossover, and mutation (chosen in part by analogy with the natural world) to direct population (over a series of time steps or generations) towards convergence at the global optimum (Santosa & Willy, 2011).

The modified genetic algorithm (MGA) tries to find efficient schedules by minimizing the mean flow time and the number of tardy products simultaneously using Pareto’s dominant concept. After the objective values are found, an efficient schedule is determined using the non-dominated sorting and crowding distance procedure (Deb et al., 2002). The following is the pseudocode of the proposed MGA (see Figure 3).

---

**Simple Heuristic Algorithms**

| Input (s): | Output (s): |
|------------|-------------|
| t → number of products | Obtained non-dominated solution, include: |
| n → number of jobs | - π → product sequence |
| f → number of factories | - πₑₑ,ₑₚ → job to factory sequence |
| m → number of machines | - πₑₑₚₑₑ,ₑₚ → job to factory – machine sequence |
| ATᵢ → assembly time of product s at the assembly stage | - F → mean flow time |
| DDᵢ → due date of product s at the assembly stage | - Nₑₑ → number of tardy product |
| PTᵢₑₑ → production time of job j on machine k in factory q | Begin |
| Nₑₑ → the assembly program of product s | Generate an initial product sequence (π) based on AT, and DD, using BS-I or BS-D algorithm |
| EFₑₑ → a subset of factories where job f can be assigned | For q = 1 : f |
| Obtain the job to factory sequence (πₑₑ,ₑₚ) using the job to factory procedure in each π |
| For t = 1 : n |
| Define the possible job to factory-machine sequence (πₑₑₚₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑطور


**Figure 2.** Pseudocode of the simple heuristic algorithms.

| j | PTₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑᵉodies


| j | PTₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑₑ孛　| Pₑₑₑₑₑₑₑ孛　| w孛　| min{w孛　} |
|---|---|---|---|
| 7 | 7 0 0 0 0 7 6 | 0 | 0 | 6.5000 | 6.5000 |
| 12 | 9 7 0 0 8 9 | 8.0000 | 0 | 8.5000 | 8.0000 |
| 13 | 5 5 8 7 0 0 | 5.0000 | 7.5000 | 0 | 5.0000 |
| 14 | 1 0 9 0 0 0 0 | 9.5000 | 0 | 0 | 9.5000 |

**Table 2.** Weight calculation for the simple example.
Modified Genetic Algorithm (MGA)

Parameter (s):
- \( N \rightarrow \) population size
- \( P_m \rightarrow \) mutation probability
- \( G_{\text{max}} \rightarrow \) number of maximum generation

Input (s):
- \( I \rightarrow \) number of products
- \( n \rightarrow \) number of jobs
- \( m \rightarrow \) number of machines
- \( A_{ij} \rightarrow \) assembly time of product \( i \) at the assembly stage
- \( D_{ds} \rightarrow \) due date of product \( s \) at the assembly stage
- \( P_{ij} \rightarrow \) production time of job \( j \) on machine \( k \) in factory \( g \)
- \( N_j \rightarrow \) the assembly program of product \( s \)
- \( L_j \rightarrow \) a subset of factories where job \( j \) can be assigned

Output (s):
- Obtained non-dominated solution, include:
  - \( \pi \rightarrow \) product sequence
  - \( \pi_{\pi_q}, V_q \rightarrow \) job to factory sequence
  - \( \pi_{\text{NNIK}}, V_k \rightarrow \) job to factory – machine sequence
  - \( F \rightarrow \) mean flow time
  - \( N_j \rightarrow \) number of tardy product

Begin
Generate an initial population \( X^0 = \{X^0_1, X^0_2, X^0_3, \ldots, X^0_N\} \). A population consists of \( N \) chromosomes, and a chromosome consists of a string of length \( t \) product (\( \pi \)).

\( G = 1 \)
While \( G \leq G_{\text{max}} \)
For \( crh = 1 : N \)
  Calculate fitness \( f(X^0) \) and evaluate the current population using Sub-MGA 1 procedure.
End For
Select an efficient solution using non-dominated sorting and crowding distance procedure
Set the efficient solution as an elite chromosome and duplicated on the first position of the current population.
For \( i = 1 : N \)
  For \( r = \text{rand}() \)
    If \( r < P_m \)
      Chromosome \( \pi \) (except elite chromosomes) will be mutated at each locus (position in a chromosome) by a slide, swap, or slide operation
    End If
  End For
End While
Preserved the best chromosome \( \pi, \pi_{\pi_q}, V_q, \) and \( \pi_{\text{NNIK}}, V_k,q \)
End

Sub-MGA 1

Begin
Define the initial job to factory sequence \( (\pi_{\pi_q}, V_q) \) using the job to factory procedure.
For \( q = 1 : f \)
  Define the optimal job to factory sequence \( (\pi_{\pi_q}, V_q) \) using the GA’s operations in Sub-MGA2 with makespan as the single objective function.
End For
Preserved the best chromosome \( \pi_{\pi_q}, V_q \) and \( \pi_{\text{NNIK}}, V_k,q \)
Calculate mean flow time and the number of tardy products which is related to \( \pi \).
End

Sub-MGA 2

Begin
Generate an initial population, It is generated based on the initial job to factory sequence \( (\pi_{\pi_q}) \). A population consists of \( 1 \) chromosome, and a chromosome consists of a string of length \( n \) job. The \( y \)-th gene contains the job index at position \( y \), so a chromosome describes the initial job to factory sequence \( (\pi_{\pi_q}) \) chosen on a single factory.

\( G = 1 \)
While \( G \leq G_{\text{max}} \)
Set the initial job to factory machine sequence \( (\pi_{\pi_q}) \) as the best initial solution.
Set the best initial solution as an elite chromosome duplicate or copy the best solution once, and assign it as A and B chromosome (Elitism). In this case, the A chromosomes are not undergone mutation.
For \( i = 1 : N \)
  For \( r = \text{rand}() \)
    If \( r < P_m \)
      Chromosome \( \pi \) (except elite chromosomes) will be mutated at each locus (position in a chromosome)
      by a slide, swap, or slide operation
    End If
  End For
Define the job to factory-machine sequence \( (\pi_{\text{NNIK}}, W_k) \) using the modified NRs assignment rules.
Calculate the completion time of each job at the production line \( (C_{\pi_q}, V_k) \).
Calculate the makespan \( (G_{\text{max}}) \) of all jobs in factory \( q \).
Set a chromosome A or B with the smallest makespan as the initial job to factory machine sequence \( (\pi_{\pi_q}) \) in the next generation.
End For
\( G = G + 1 \)
End While
Preserved the best solution \( \pi_{\pi_q} \) and \( \pi_{\text{NNIK}}, W_k \)

End

Figure 3. Pseudocode of the Modified Genetic Algorithm (MGA).
4.2.2. The Modified Simulated Annealing (MSA)
Simulated annealing is a generalization of a Monte Carlo method for statistically finding the global optimum between different local optima. Simulated Annealing (SA) is more powerful than simple local search by accepting worse solutions with a certain probability. There are several steps that the simulated annealing algorithm go through while a specified rate decreases the temperature. The annealing process runs through several iterations at each temperature to sample the search space. It generates a new solution at each cycle by applying a random change to the current solution (Santosa & Willy, 2011). In Simulated Annealing, a system is initialized at a temperature $T$ with some configuration whose energy is evaluated to be $E$.

A new configuration is constructed by applying a random chance, and the change in energy $\Delta E$ is computed. The new configuration is unconditionally accepted if it lowers the energy of the system. If the change increases the system’s energy, the new configuration is accepted with some random probability. In the original Metropolis scheme (Santosa & Willy, 2011), the probability is given by the Boltzmann factor $\exp(-\Delta E/kT)$. This process is repeated sufficient times at the current temperature to sample the search space, and then the temperature is decreased. The process is repeated at successively lower temperatures until a frozen state is achieved. This procedure allows the system to move to lower energy states while still jumping out of local minima (especially at higher temperatures) due to the probabilistic acceptance of some upward moves. The following is the pseudocode of the proposed MSA (see Figure 4).

4.2.3. The Modified Particle Swarm Optimization (MPSO)
Particle swarm optimization (PSO) was developed based on the behavior of a swarm of insects, such as a flock of birds or a school of fish. The word particle denotes, for example, a bee in a colony or a bird in a flock. Each individual or particle in a swarm behaves in a distributed way using its own intelligence and the collective or group intelligence of the swarm. As such, if one particle discovers a good path, the rest of the swarm will also be able to follow the good path instantly even if their location is far away in the swarm. In the context of multivariable optimization, the swarm is assumed to be of specified or fixed size with each particle located initially at random locations and velocity in the multidimensional design space. Each particle wanders around in the design space and remembers the best position (in terms of the food source or objective function value) it has discovered. Each particle keeps track of its coordinates in the problem space, which are associated with the best solution (fitness) it has achieved so far. This value is called Lbest. When a particle takes all the population as its topological neighbors, the best value is a global best and is called Gbest. At each time step, changing the velocity of (accelerating) each particle toward its pbest and Lbest locations (local version of PSO). Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward Lbest and Gbest locations. In the past several years, PSO has been successfully applied in many research and application areas. It is demonstrated that PSO gets better results in a faster and cheaper manner compared to other methods. The following is the pseudocode of the proposed MPSO (see Figure 5).
Modified Simulated Annealing (MSA)

Parameter(s):
- $\varepsilon$ → maximum temperature
- $F_0$ → temperature factor
- $N_{\text{max}}$ → maximum number of iterations

Input(s):
- $T$ → number of products
- $N$ → number of jobs
- $M$ → number of factories
- $A_{Ti}$ → assembly time of product $i$ at the assembly stage
- $DD_{is}$ → due date of product $i$ at the assembly stage
- $P_{ijlk}$ → production time of job $j$ on machine $k$ in factory $q$
- $N_i$ → the assembly program of product $i$
- $L_{ij}$ → a subset of factories where job $j$ can be assigned

Output(s):
- Obtained non-dominated solution, include:
  - $\pi$ → product sequence
  - $\pi_{eq}$ → job to factory sequence
  - $\pi_{eq}, L_{ij}$ → job to factory → machine sequence
  - $F$ → mean flow time
  - $N_T$ → number of tardy products

Begin
Generate a product sequence ($\pi$) randomly, then set as an initial solution ($a$). $\pi$ consists of a string of length $T$ products.

While $T > 0$
  For $i = 1: T$
    Generate a candidate solution ($b$) by applying several local search techniques often used in the gene mutation process in the genetic algorithm, i.e., flip, swap, and slide.
    Calculate fitness $f(a)$ and $f(b)$ using Sub-MSA 1 procedure.
    Select an efficient solution using non-dominated sorting and crowding distance procedure.
    If $f(b)$ better than $f(a)$
      $a = b$
    Else
      $r = \text{rand}(0,1)$
      $p = \exp \left( \frac{-f(b) - f(a)}{\varepsilon} \right)$
      If $r < p$
        $a = b$
      Else
        $a = a$
    End If
  End For
  Temperature update $T = T_0 \cdot F_0$
End While

Preserved the best chromosome $\pi$, $\pi_{eq}$, $W_{q}$, and $N_{\text{eq}}$, $W_{k}$, $q$

Sub-MSA 1

Begin
Define the initial job to factory sequence ($\pi_{eq}$) using the job to factory procedure.
For $q = 1: T$
  Define the optimal job to factory sequence ($\pi_{eq}$) using the SA’s operations in Sub-MSA2 with makespan as the single objective function.
End For
Preserved the best chromosome $\pi_{eq}$, $W_{q}$, and $N_{\text{eq}}$, $W_{k}$, $q$
Calculate mean flow time and the number of tardy products which is related to $\pi$.

End

Sub-MSA 2

Begin
Set the initial job to factory sequence ($\pi_{eq}$) as the initial solution ($a$), where $a$ consists of a string of length $N$ jobs in factory $q$.

While $N > 0$
  For $i = 1: N$
    Generate a candidate solution ($b$) by applying several local search techniques often used in the gene mutation process in the genetic algorithm, i.e., flip, swap, and slide.
    Define the job to factory-machine sequence ($\pi_{eq}$) using the modified NR3 assignment rules.
    Calculate the completion time of each job at the production line ($C_{ij}$).
    Calculate the makespan ($\text{Cmax}$) of all jobs in factory $q$
    Calculate the fitness value
  End For
  If $f(b)$ better than $f(a)$
    $a = b$
  Else
    $r = \text{rand}(0,1)$
    $p = \exp \left( \frac{-f(b) - f(a)}{\varepsilon} \right)$
    If $r < p$
      $a = b$
    Else
      $a = a$
    End If
  End If
  Temperature update $T = T_0 \cdot F_0$
End While

Preserved the best solution $\pi_{eq}$ and $N_{\text{eq}}$, $W_{k}$

End

Figure 4. Pseudocode of the Modified Simulated Annealing (MSA).
Modified Particle Swarm Optimization (MPSO)

Parameter(s):
N → population size
ite → number of maximum iteration

Input (s):
I → number of products
n → number of machines
f → number of factories
ATs → assembly time of product s at the assembly stage
DDe → due date of product s at the assembly stage
PTj,k → production time of job j on machine k in factory q
Nps → the assembly program of product s
LFe → a subset of factories where job j can be assigned

Output (s):
 Obtained non-dominated solution, include:
- π → product sequence
- πFeq, Vq → job to factory sequence
- πMFeq, Vqk → job to factory-machine sequence
- t → mean flow time
- NFeq → number of tardy products

Begin
Use Lowest Relative Value (LRV) algorithm to get an initial position of particle (X0 = (X01, X02, X03, ..., X0Np)), and an initial population of particles π = {π1, π2, π3, ..., πn}. A particle consists of a string of length I product (π).
Generate an initial velocity of particles V0 = (V01, V02, V03, ..., V0Np) = 0
i = 1
While i ≤ ite
Calculate the fitness value using Sub-MPSO 1 procedure
Determine Local Best Value (Lbest), where Lbest = {Xs, Xs1, Xs2, Xs3, ..., XsNp}
Determine Global Best Value (Gbest) using non-dominated sorting and crowding distance procedure
Update velocity of particle:
Vb1 = Vb01 + c1*r1*(Lbest - Xb01) + c2*r2*(Gbest - Xb01),
where c1 and c2 are the cognitive (individual) and social (group) learning rates, respectively, and r1 and r2 are uniformly distributed random numbers in the range 0 and 1. he value of c1 and c2 are usually assumed to be 1 so that c1r1 and c2r2 ensure that the particles would overly the target about one time.
Update position of particle:
Xb1 = Xb01 + Vb1
Update population of particle using LRV algorithm
If Xb1 = Xb01
STOP
End If
End While
Preserved the best chromosome π, πFeq, Vq, and πMFeq, Vqk
End

Sub-MPSO 1

Begin
Define the initial job to factory sequence (πFeq, Vq) using the job to factory procedure.
For q = 1 : f
- Define the optimal job to factory sequence (πFeq, Vq) using the SA’s operations in Sub-MPSO2 with makespan as the single objective function.
End For
Preserved the best chromosome πFeq, Vq and πMFeq, Vqk
Calculate mean flow time and the number of tardy products which is related to π.
End

Sub-MPSO 2

Begin
Duplicate job to factory sequence (πFeq) for N times and set as the initial population of particles π = {π1, π2, π3, ..., πn}. Then set a vector of X as the initial position of particles (X0 = (X01, X02, X03, ..., X0Np)). Generate an initial velocity of particles V0 = (V01, V02, V03, ..., V0Np) = 0
i = 1
While i ≤ ite
For i = 1 : N
- Define the job to factory-machine sequence (πMFeq, Vq) using the modified NR assignment rules.
- Calculate the completion time of each job at the production line (Cpij, Vq, j).
- Calculate the makespan (Omax) of all jobs in factory q
- Calculate the fitness value
End For
Determine Local Best Value (Lbest) and Global Best Value (Gbest) based on the fitness value
Update velocity of particle:
Vb1 = Vb01 + c1*r1*(Lbest - Xb01) + c2*r2*(Gbest - Xb01),
where c1 and c2 are the cognitive (individual) and social (group) learning rates, respectively, and r1 and r2 are uniformly distributed random numbers in the range 0 and 1. he value of c1 and c2 are usually assumed to be 1 so that c1r1 and c2r2 ensure that the particles would overly the target about one time.
Update position of particle:
Xb1 = Xb01 + Vb1
Update population of particle using LRV algorithm
If Xb1 = Xb01
STOP
End If
i = i + 1
End While
Preserved the best chromosome π, πFeq, Vq, and πMFeq, Vqk
End

Figure 5. Pseudocode of the Modified Particle Swarm Optimization (MPSO).
4.3. Proposed the job to factory procedure

The job to factory procedure is used to assign each job to one of the eligible factories and to define the job priority scale by considering the job sequence (π) and assembly program of the related product. This procedure worked in the following manner. As the example, for π = {2,6,1,4,5,3}, start from the first product in π (s = 2). First, find j*, where j* ∈ N2. Second, for all j*, determine the weight of each factory (w*j*qi, ∀q, i) by an average value of PT*j*qi, where k = 1 to m. Third, find a factory that has the least weight (except: w*j* = 0), then set it as the chosen factory. The last step is to define a job to factory sequence (πFq, ∀q), so j* has to be sorted by descending order of min(w*j*). Repeat the same steps until all jobs for all products are scheduled. The results are: πF1 = {14,12,13,1,4,8}; πF2 = {11,2,6,5,15}; πF3 = {7,3,10,9}.

5. Result and discussion

In general, this research is conducted in three stages: 1) verifying and validating the proposed algorithm; 2) identifying the optimal parameter configuration for the proposed algorithm; and 3) measuring, comparing, and testing the proposed algorithm’s performance.

All proposed algorithms are coded in MATLAB R2014a and run on a computer with Intel Core i3-6006 U CPU at 2.0 GHz, 8Gb RAM, and 1TB HDD in Windows 7 system. Meanwhile, the solution of the mathematical model using an exact algorithm will be obtained by using LINGO solver. MDPMASP problem is an NP-hard optimization problem. LINGO software is not able to solve large-scale problems in an acceptable time. To get a fair performance comparison between the exact and proposed algorithms, the elapsed time to solve each instance is equal to 60 minutes.

5.1. Verification and validation

A model that is built must be credible and representative. It is used to ensure the reproducibility of measurement results in real conditions. In software testing, there are two terms: verification and validation. Verification is the process of determining whether algorithms and programs run as intended by repeatedly running the program. The verification ensures that the conceptual model (pseudocode and assumptions) is translated correctly into the programming language (Law & Kelton, 2000). Validation is the process of determining whether the stated model accurately represents the system (Hoover & Perry, 1989). Validation is the process of determining whether a conceptual model (as opposed to a computer program) accurately represents the modeled system (Law & Kelton, 2000).

In this study, verification and validation were carried out by performing manual calculations for small cases. Verification is carried out to ensure whether the MDPMASP with eligibility constraints has been represented in the real system properly, whether all mathematical operations and relations are correct, whether there are errors in coding, and whether the structure of the solution search for each algorithm is systematic and logical. While the validation is done by reviewing the structure of the solution and
the proximity of the solution generated by the proposed algorithms to the optimal solution. The optimal solution is obtained by running the MILP model and the total enumeration technique.

For example, MDMASP with eligibility constraints with performance criteria $\bar{F}$ and $n_T$, consists of three products ($t = 3$), eight jobs ($n = 8$) and two unrelated distributed factories ($f = 2$) with two unidentical parallel machines in each factory ($m = 2$). Then by performing the procedure for generating $AT_s$ and $DD_s$ randomly at the intervals in Table 3, the assembly time and lead time for each product are obtained (see, Table 4). Every product $s$ is composed of several jobs that exist in the related assembly program $N_s$. The assembly program of the three products are $N_1 = \{1,4\}$, $N_2 = \{2,6,7\}$, and $N_3 = \{3,5,8\}$ see in Table 5. This means that job 1 and job 4 must be finished in order to assemble product 1 Table 2.

There are 73,728 feasible solutions based on the computational results of the MILP model and total enumeration technique. The best non-dominated solution can be selected from all feasible solutions, which is expressed as the Pareto Front (PF). Non-dominated solution of each problem is given in Table 6.

Two significant findings were drawn as a result of the verification process: 1) the procedure for finding solutions was in accordance with the logic of the flow chart and conceptual model that had been defined previously. There are four interrelated decisions that must be made, namely: determining the feasible and effective factory for each job;

| Table 3. Problem setting for verification and validation process: a small case. |
|---|
| $t$ | $n$ | $f$ | $m$ | $AT_{INT}$ | $DD_{INT}$ |
| Problem 1 | 3 | 8 | 2 | 2 | INT[20,35] |
| Problem 2 | 3 | 8 | 2 | 2 | INT[5,20] |

| Table 4. The production time for each job. |
|---|
| Product (s) | $AT_s$ | $DD_s$ | $AT_s$ | $DD_s$ |
| Problem 1 | Problem 2 |
| 1 | 24 | 13 | 13 | 59 |
| 2 | 28 | 77 | 20 | 44 |
| 3 | 31 | 77 | 6 | 1 |

| Table 5. Eligibility constraints and production time for each job. |
|---|
| Job (j) | Eligibility Constraints | Product Index | Production Time ($PT_{j1k}$) |
| Problem 1 | Problem 2 |
| $P_{T_{j11}}$ | $P_{T_{j12}}$ | $P_{T_{j21}}$ | $P_{T_{j22}}$ | $P_{T_{j1k}}$ | $P_{T_{j2k}}$ |
| 1 | 1 | 1 | 11 | 14 | 0 | 0 | 28 | 26 | 0 | 0 |
| 2 | 1,2 | 2 | 6 | 6 | 7 | 8 | 21 | 22 | 21 | 25 |
| 3 | 1 | 3 | 13 | 17 | 0 | 0 | 23 | 26 | 0 | 0 |
| 4 | 1 | 1 | 14 | 12 | 0 | 0 | 20 | 34 | 0 | 0 |
| 5 | 1 | 3 | 0 | 0 | 5 | 10 | 0 | 0 | 27 | 25 |
| 6 | 1,2 | 2 | 7 | 17 | 9 | 13 | 34 | 25 | 21 | 32 |
| 7 | 1 | 2 | 7 | 14 | 0 | 0 | 26 | 32 | 0 | 0 |
| 8 | 2 | 3 | 0 | 0 | 9 | 15 | 0 | 0 | 26 | 21 |
determining job priorities for selecting machines in a factory; assigning jobs to each machine; and determining the sequence of the job assembling process into the final product at the final stage. Figure 6 and Table 7 illustrate the problem, best feasible solution, and optimal solution. 2) The computer program contains no coding errors, and the solution structures are mathematically correct and logically consistent. Validation is done to ensure that the algorithm and programming code can represent and solve the proposed problem properly.

Table 8 shows the convergence and divergence values of each proposed algorithm. The convergence value represents the proximity of the resulting solution to the optimal solution. The smaller the convergence value, the better the algorithm’s performance. While the value of divergence is used to assess the diversity of the resulting solutions. The smaller the divergence value (closer to zero), the more diverse solutions the algorithm can produce. The computational results show that all the proposed algorithms perform very well in solving multi-objective MDPMSP problems with eligibility constraints, although there are differences in the level of accuracy. So it can be concluded that the proposed model (MILP, algorithm, and program) has been verified, is valid and can represent and solve problems very well. As a result, we can perform further experiments to find out how the algorithm performs on larger problem sizes.

5.2. Parameter tuning

The quality and effectiveness of the algorithm are affected by the parameter configuration. Therefore, to obtain optimal performance, these parameters must be adjusted by using an efficient design of experiments. The most commonly used and intuitive strategy of experimental design is the full factorial design William G and Gertrude M (1992) which loses its efficiency by increasing the number of parameters. Here, we applied the Taguchi approach Ross (1988), where a large number of decision variables would be tuned through a small number of experiments (Karimi & Davoudpour, 2014).

| Mean Flow Time (F) | Number of Tardy Product (nT) |
|--------------------|------------------------------|
| Problem 1          |                              |
| 65,6667            | 1                            |
| 63,3333            | 2                            |
| Problem 2          |                              |
| 55,3333            | 2                            |
| 55,6667            | 3                            |
Table 7. Solution representation of MDPMASP with eligibility constraints.

| Alg | \( \pi \) | \( \pi_{F_1} \) | \( \pi_{F_2} \) | \( \pi_{FM_{11}} \) | \( \pi_{FM_{12}} \) | \( \pi_{FM_{21}} \) | \( \pi_{FM_{22}} \) | Objective Value | Optimal Solution |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| Problem 1 | SH-1 | 1 2 3 4 1 7 2 3 6 8 5 | 4 7 1 2 3 6 5 | 8 | 67.0000 | 2.0000 | 65.6667 | 1.0000 |
| | | 2 3 1 7 2 3 4 1 6 8 5 | 7 4 1 2 3 6 5 | 8 | 65.6667 | 1.0000 | 63.3333 | 2.0000 |
| | SH-2 | 1 2 3 4 1 7 2 3 6 8 5 | 1 7 3 4 2 6 5 | 8 | 65.0000 | 2.0000 |
| | | 2 3 1 7 2 3 4 1 6 8 5 | 7 3 1 2 4 6 5 | 8 | 65.6667 | 1.0000 |
| | SH-3 | 1 2 3 4 1 7 2 3 6 8 5 | 4 7 1 2 3 6 5 | 8 | 67.0000 | 2.0000 |
| | | 2 3 1 7 2 3 4 1 6 8 5 | 7 4 1 2 3 6 5 | 8 | 65.6667 | 1.0000 |
| | SH-4 | 1 2 3 4 1 7 2 3 6 8 5 | 1 7 3 4 2 6 5 | 8 | 65.0000 | 2.0000 |
| | | 2 3 1 7 2 3 4 1 6 8 5 | 7 3 1 2 4 6 5 | 8 | 65.6667 | 1.0000 |
| | MGA | 2 1 3 2 7 4 1 3 6 8 5 | 2 7 1 4 3 6 5 | 8 | 63.3333 | 2.0000 |
| | | 2 3 1 7 2 3 4 1 6 8 5 | 7 3 1 2 4 6 5 | 8 | 65.6667 | 1.0000 |
| | MSA | 2 1 3 2 7 4 1 3 6 8 5 | 2 7 1 4 3 6 5 | 8 | 63.3333 | 2.0000 |
| | | 2 3 1 7 2 3 4 1 6 8 5 | 7 3 1 2 4 6 5 | 8 | 65.6667 | 1.0000 |
| | MPSO | 2 1 3 2 7 4 1 3 6 8 5 | 2 7 1 4 3 6 5 | 8 | 63.3333 | 2.0000 |
| | | 2 3 1 7 2 3 4 1 6 8 5 | 7 3 1 2 4 6 5 | 8 | 65.6667 | 1.0000 |

| Alg | \( \pi \) | \( \pi_{F_1} \) | \( \pi_{F_2} \) | \( \pi_{FM_{11}} \) | \( \pi_{FM_{12}} \) | \( \pi_{FM_{21}} \) | \( \pi_{FM_{22}} \) | Objective Value | Optimal Solution |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| Problem 2 | SH-1 | 1 2 3 1 4 7 2 5 8 6 | 3 4 2 1 7 5 8 6 | 5 | 65.6667 | 2.0000 | 63.3333 | 2.0000 |
| | | 3 1 2 3 1 4 7 2 5 8 6 | 3 2 4 1 7 5 8 6 | 5 | 65.6667 | 2.0000 | 63.3333 | 2.0000 |
| | SH-2 | 1 2 3 1 4 7 2 5 8 6 | 3 4 2 1 7 5 8 6 | 5 | 65.6667 | 2.0000 | 63.3333 | 2.0000 |
| | | 3 1 2 3 1 4 7 2 5 8 6 | 3 2 4 1 7 5 8 6 | 5 | 65.6667 | 2.0000 | 63.3333 | 2.0000 |
| | SH-3 | 1 2 3 1 4 7 2 5 8 6 | 3 4 2 1 7 5 8 6 | 5 | 65.6667 | 2.0000 | 63.3333 | 2.0000 |
| | | 3 1 2 3 1 4 7 2 5 8 6 | 3 2 4 1 7 5 8 6 | 5 | 65.6667 | 2.0000 | 63.3333 | 2.0000 |
| | SH-4 | 1 2 3 1 4 7 2 5 8 6 | 3 4 2 1 7 5 8 6 | 5 | 65.6667 | 2.0000 | 63.3333 | 2.0000 |
| | | 3 1 2 3 1 4 7 2 5 8 6 | 3 2 4 1 7 5 8 6 | 5 | 65.6667 | 2.0000 | 63.3333 | 2.0000 |
| | MGA | 1 2 3 1 4 7 2 5 8 6 | 3 4 2 1 7 5 8 6 | 5 | 65.6667 | 2.0000 | 63.3333 | 2.0000 |
| | | 3 1 2 3 1 4 7 2 5 8 6 | 3 2 4 1 7 5 8 6 | 5 | 65.6667 | 2.0000 | 63.3333 | 2.0000 |
| | MSA | 1 2 3 1 4 7 2 5 8 6 | 3 4 2 1 7 5 8 6 | 5 | 65.6667 | 2.0000 | 63.3333 | 2.0000 |
| | | 3 1 2 3 1 4 7 2 5 8 6 | 3 2 4 1 7 5 8 6 | 5 | 65.6667 | 2.0000 | 63.3333 | 2.0000 |
| | MPSO | 1 2 3 1 4 7 2 5 8 6 | 3 4 2 1 7 5 8 6 | 5 | 65.6667 | 2.0000 | 63.3333 | 2.0000 |

Table 8. Convergency and divergency matrix.

| Alg | Problem 1 | Problem 2 |
|-----|-----|-----|
| SH-1 | 0.8333 | 2.3334 |
| SH-2 | 0.6009 | 2.0000 |
| SH-3 | 0.8333 | 2.3334 |
| SH-4 | 0.6009 | 2.0000 |
| MGA | 0 | 0 |
| MSA | 0 | 0 |
| MPSO | 0 | 0 |

Taguchi’s experimental design method was used to analyze the impacts of the whole set of parameters using a small number of experiments based on an extraordinary orthogonal array design. Numerous control parameters are examined during factor analysis in order to optimize the performance of the proposed metaheuristic algorithm, i.e. population size for MGA (\( N_{MGA} \)), mutation probability
(\(P_m\)), maximum number of generation (Gmax), initial temperature (\(T_0\)), maximum number of iteration for MSA (\(It_{\text{max}}\_\text{MSA}\)), temperature reduction constant (\(F_r\)), minimum temperature (\(\epsilon\)), population size for MPSO (\(N_{\text{MPSO}}\)), maximum number of iteration for MPSO (\(It_{\text{max}}\_\text{MPSO}\)), and initial velocity (\(v_0\)). Each parameter is categorized into three levels, and DoF = 6. An orthogonal array \(L_9(3^4)\) with one dummy factors in MGA and MPSO was chosen as the experimental design and summarized in Table 9. The Taguchi approach results in nine different treatments, which have to be run five times.

The signal-to-noise (S/N) ratio is used to analyze variations of proposed algorithms performance in response to each set of parameters. The signal describes the desired impact of the yield, while the noise represents the undesired impact. Because the S/N ratio comes from the quadratic loss function, three examples are widely used which are (1) the nominal is best, (2) the smaller is better and (3) the bigger is better. Based on the scheduling problem’s features, the S/N ratio is applied for the smaller-the-better of the convergence and divergence metric,

\[
S/N_{\text{ratio}} = -10\log \frac{1}{R} \sum_{r=1}^{R} (\text{conv}_r * \text{div}_r)^2
\]

where \(n\) is the number of replications and \(y\) is the output of the \(i\)th replication.

**Table 9. Parameter configurations.**

| Trial | Algorithm                  | Parameter Value       | Configuration of Parameter Value |
|-------|----------------------------|-----------------------|---------------------------------|
| 1     | Modified Genetic Algorithm | \(N_{\text{MGA}}\) (50) | \(P_m\) (0.3) \(Gmax\) (5)     |
| 2     | \(N_{\text{MGA}}\) (50)   | \(P_m\) (0.5) \(Gmax\) (10) |
| 3     | \(N_{\text{MGA}}\) (50)   | \(P_m\) (0.7) \(Gmax\) (20) |
| 4     | \(N_{\text{MGA}}\) (100)  | \(P_m\) (0.3) \(Gmax\) (10) |
| 5     | \(N_{\text{MGA}}\) (100)  | \(P_m\) (0.5) \(Gmax\) (20) |
| 6     | \(N_{\text{MGA}}\) (100)  | \(P_m\) (0.7) \(Gmax\) (5)  |
| 7     | \(N_{\text{MGA}}\) (200)  | \(P_m\) (0.3) \(Gmax\) (20) |
| 8     | \(N_{\text{MGA}}\) (200)  | \(P_m\) (0.5)          |
| 9     | \(N_{\text{MGA}}\) (200)  | \(P_m\) (0.7) \(Gmax\) (10) |
| 1     | Modified Simulated Annealing | \(T_0\) (10^3) \(It_{\text{max}}\_\text{MSA}\) (10) | \(F_r\) (0.3) \(\epsilon\) (10^{-3}) |
| 2     | \(T_0\) (10^3) \(It_{\text{max}}\_\text{MSA}\) (20) | \(F_r\) (0.5) \(\epsilon\) (10^{-3}) |
| 3     | \(T_0\) (10^3) \(It_{\text{max}}\_\text{MSA}\) (50) | \(F_r\) (0.7) \(\epsilon\) (10^{-3}) |
| 4     | \(T_0\) (10^3) \(It_{\text{max}}\_\text{MSA}\) (10) | \(F_r\) (0.5) \(\epsilon\) (10^{-3}) |
| 5     | \(T_0\) (10^3) \(It_{\text{max}}\_\text{MSA}\) (20) | \(F_r\) (0.7) \(\epsilon\) (10^{-3}) |
| 6     | \(T_0\) (10^3) \(It_{\text{max}}\_\text{MSA}\) (50) | \(F_r\) (0.3) \(\epsilon\) (10^{-3}) |
| 7     | \(T_0\) (10^3) \(It_{\text{max}}\_\text{MSA}\) (10) | \(F_r\) (0.7) \(\epsilon\) (10^{-3}) |
| 8     | \(T_0\) (10^3) \(It_{\text{max}}\_\text{MSA}\) (20) | \(F_r\) (0.3) \(\epsilon\) (10^{-3}) |
| 9     | \(T_0\) (10^3) \(It_{\text{max}}\_\text{MSA}\) (50) | \(F_r\) (0.5) \(\epsilon\) (10^{-3}) |
| 1     | Modified Particle Swarm Optimization | \(N_{\text{MPSO}}\) (50) \(It_{\text{max}}\_\text{MPSO}\) (5) | \(v_0\) (0) - |
| 2     | \(N_{\text{MPSO}}\) (50) \(It_{\text{max}}\_\text{MPSO}\) (10) | \(v_0\) (0) - |
| 3     | \(N_{\text{MPSO}}\) (50) \(It_{\text{max}}\_\text{MPSO}\) (20) | \(v_0\) (0) - |
| 4     | \(N_{\text{MPSO}}\) (100) \(It_{\text{max}}\_\text{MPSO}\) (5) | \(v_0\) (0) - |
| 5     | \(N_{\text{MPSO}}\) (100) \(It_{\text{max}}\_\text{MPSO}\) (10) | \(v_0\) (0) - |
| 6     | \(N_{\text{MPSO}}\) (100) \(It_{\text{max}}\_\text{MPSO}\) (20) | \(v_0\) (0) - |
| 7     | \(N_{\text{MPSO}}\) (200) \(It_{\text{max}}\_\text{MPSO}\) (5) | \(v_0\) (0) - |
| 8     | \(N_{\text{MPSO}}\) (200) \(It_{\text{max}}\_\text{MPSO}\) (10) | \(v_0\) (0) - |
| 9     | \(N_{\text{MPSO}}\) (200) \(It_{\text{max}}\_\text{MPSO}\) (20) | \(v_0\) (0) - |
While conducting each experiment, the obtained objective function, i.e. convergence and divergence metrics, must be converted according to the Taguchi method and proportional to the signal-to-noise ratio, and analysis is provided according to its variations. In the Taguchi method, the S/Ns ratio indicates the variable ratio and the objective function in each experiment which is converted to this ratio in order to make a decision. Moreover, the convergence value will be measured based on the Pareto front (PF) value obtained by combining the entire solution set from each algorithm, then from that set the non-dominated solution set (Deb et al., 2002).

At this stage, the experiment was carried out with as many as 30 replications for each trial with \( t = 3, n = 8, f = 2, m = 2, PT_{\text{INT}} = \text{INT}[10, 30] \) and \( AT_{\text{INT}} = \text{INT}[30, 50] \). All processing time and other information are generated randomly. All of the obtained solutions from the Taguchi design of experiments for adjustment of parameters have been shown as S/Ns ratio in Figure 7 and the best factor levels for the proposed metaheuristic algorithms are provided in Table 10.

5.3. Comparison algorithm and evaluation

Performance measurement is carried out to determine the quality of the proposed algorithm. Because the multi-objective problems result in multiple non-dominated solutions, the performance of several algorithms compared is a complex problem

![Figure 7. Mean S/Ns ratio.](image)

**Table 10. The optimum level of each parameter.**

| Algorithm | Parameter | Optimum level |
|-----------|-----------|---------------|
| MGA       | Population size for MGA, \( N_{\text{MGA}} \) | 200            |
|           | Mutation probability, \( p_m \)       | 0.3            |
|           | The maximum number of generations, \( G_{\text{max}} \) | 20             |
| MSA       | Initial temperature, \( T_0 \)       | \( 10^4 \)     |
|           | The maximum number of iterations or cycles for MSA, \( t_{\text{maxM}} \) | 50             |
|           | Temperature reduction constant, \( F_t \) | 0.7            |
|           | The minimum temperature, \( \epsilon \) | \( 10^{-3} \)  |
| MPSO      | Population size for MPSO, \( N_{\text{MPSO}} \) | 200            |
|           | The maximum number of iterations or cycles for MPSO, \( t_{\text{maxM}} \) | 20             |
|           | Initial velocity, \( v_0 \)          | 0              |
and requires some performance metrics. Through this research, all of the proposed algorithms are measured by four performance metrics, i.e. overall non-dominated vector generation (ONVG), overall number of non-dominated solution (ONSN), average quality (AQ), and distance metric (DIₗ) proposed by (Qian et al., 2009). The experimental parameters are informed in Table 11. It was generated based on the L₈ (2⁸) orthogonal array of the Taguchi experimental design, with five two-level factors and DoF = 5. Thirty runs were generated for each set of problems using Matlab software.

The comparison of proposed algorithms is presented in Figure 7. The results show that all proposed algorithms can obtain satisfactory solutions for given MDPMASP with eligibility constraints. But the statistical performance metrics show that there exist some differences between them.

The ONVG matrix is used to measure the number of non-dominated solution points generated by the proposed algorithm. The non-dominated solution set generated by the MSA algorithm produces a larger average ONVG metric than that generated by the other algorithms. This means that the MSA algorithm is able to produce more non-dominated solutions than the other proposed algorithms even though MSA has a smaller number of feasible solutions than MGA and MPSO. From the results of the Kruskal–Wallis statistical test (see, Table 12), each proposed algorithm has a significant effect on alpha = 0.05 on the ONVG matrix.

Therefore, the Post-Hoc test (Mann-Whitney U test) needs to be done to find out which algorithm gives the best performance. Based on the results of the Mann–Whitney test (Table 13), it can be concluded that among the presented algorithms, the MSA and MPSO algorithms have a statistically significant number of dominant solutions at all instances.

**Table 11. The experimental parameters.**

| Set of Problem Parameters | t | n | f | M | AT_{int} | PT_{int} |
|---------------------------|---|---|---|---|---------|---------|
| SPP 1                     | 5 | 25| 3 | 2 | radint(30 50) | radint(10 30) |
| SPP 2                     | 5 | 25| 3 | 10| radint(10 30) | radint(30 50) |
| SPP 3                     | 5 | 70| 6 | 2 | radint(30 50) | radint(10 30) |
| SPP 4                     | 5 | 70| 6 | 10| radint(10 30) | radint(30 50) |
| SPP 5                     | 7 | 25| 6 | 2 | radint(10 30) | radint(30 50) |
| SPP 6                     | 7 | 25| 6 | 10| radint(30 50) | radint(10 30) |
| SPP 7                     | 7 | 70| 3 | 2 | radint(30 50) | radint(10 30) |
| SPP 8                     | 7 | 70| 3 | 10| radint(10 30) | radint(30 50) |

**Table 12. Kruskal–Wallis test.**

| Instance | Mean Rank | Kruskal–Wallis (Asym.Sig < 0.05) |
|----------|-----------|----------------------------------|
|          | SH-1      | SH-2    | SH-3    | SH-4    | MGA     | MSA     | MPSO    |        |
| SPP1     | 20.950    | 20.950  | 20.950  | 20.950  | 43.700  | 55.500  | 45.800  | Y(0.000) |
| SPP2     | 20.500    | 20.500  | 20.500  | 20.500  | 45.500  | 55.500  | 49.350  | Y(0.000) |
| SPP3     | 22.000    | 22.000  | 22.000  | 22.000  | 40.800  | 54.200  | 42.600  | Y(0.000) |
| SPP4     | 22.900    | 22.900  | 22.900  | 22.900  | 35.900  | 55.500  | 41.200  | Y(0.000) |
| SPP5     | 26.450    | 26.450  | 26.450  | 26.450  | 21.700  | 55.500  | 31.520  | Y(0.000) |
| SPP6     | 24.200    | 24.200  | 24.200  | 24.200  | 30.700  | 55.500  | 39.950  | Y(0.000) |
| SPP7     | 26.800    | 26.800  | 26.800  | 26.800  | 20.300  | 55.500  | 28.450  | Y(0.000) |
| SPP8     | 22.000    | 22.000  | 22.000  | 22.000  | 39.500  | 55.500  | 43.540  | Y(0.000) |

*Y if there is a significant effect on α = 0.05 and N if not.
Table 13. Mann-Whitney U test.

| No | Proposed Algorithms | SPP1 | SPP2 | SPP3 | SPP4 | SPP5 | SPP6 | SPP7 | SPP8 |
|----|---------------------|------|------|------|------|------|------|------|------|
| 1  | SH-1                | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) |
| 2  | SH-2                | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) |
| 3  | SH-3                | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) |
| 4  | SH-4                | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) |
| 5  | MGA                 | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) |
| 6  | MPSO                | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) |
| 7  | SH-2                | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) |
| 8  | SH-3                | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) |
| 9  | MGA                 | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) |
| 10 | MPSO                | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) |
| 11 | SH-2                | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) |
| 12 | SH-3                | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) |
| 13 | MGA                 | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) |
| 14 | MPSO                | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) |
| 15 | SH-2                | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) |
| 16 | SH-3                | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) | N(1.000) |
| 17 | MGA                 | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) |
| 18 | MPSO                | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) |
| 19 | MGA                 | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) |
| 20 | MPSO                | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) |
| 21 | MPSO                | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) | Y(0.000) |

*Y if there is a significant effect on α = 0.05 and N if not.

Meanwhile, based on the overall non-dominated solution number (ONSNN) metrics, it can be seen that the most competitive non-dominated solution can be obtained from the modified genetic algorithm (MGA). From D1R metric, it can be concluded that the modified GA can obtain a solution closer to the Pareto Front than the other. In addition, D1R can point out proximity and diversity, which are two important features for multi-objective optimization, and they may influence each other’s modified GA to some extent. Thus, the AQ metric is used for comprehensive comparison. From Figure 8, it can be seen that the AQ values of the modified MGA are less than those of the other. This means that the obtained solutions of the MGA are closer to the Pareto front than those of the other and the obtained solutions of the MGA distribute more uniformly and cover more area of the Pareto front. So, it is concluded that the MGA is a most effective and efficient multi-objective optimization algorithm compared to the other.

5.4. Practical insights

MDMASP problems with eligibility constraints are often encountered in manufacturing companies with the following characteristics: 1) intermittent manufacturing process; 2) production process with low volume and high product variety; 3) consists of several dedicated fabrication lines (factories) and a single assembly line. Each component can only be produced at a specific factory and assembled when all components of a product are finished processing. Each factory has several non-identical parallel machines that have different capacities and production speeds. From another perspective, the factory can also be interpreted as a supplier that produces semi-finished goods to be assembled later.
Conclusion

In addition, to minimize the mean flow time, we can reduce the inventory of semi-finished products, improve machine utility, and speed up order completion time. Moreover, by minimizing the number of tardy jobs, we can prevent delays in order completion, increase customer satisfaction, and avoid the risk of penalty fees.

6. Conclusion

We proposed a new model of Multi-Objective Modified Distributed Parallel Machine and Assembly Scheduling Problem (MDPMASP) with Eligibility Constraints and four simple heuristic algorithms and two iterative algorithms to solve it. The objective functions are minimizing mean flow time and the number of tardy products simultaneously. The final results of the proposed algorithms were dependent on its initial parameters. Therefore, the Taguchi design of experiment was used to find the best combination of parameters. Then to measure the performance of the proposed algorithms, computational experiments were performed on a set of experimental parameters.

The optimal combination of parameters is $N_{MGA}(200)$, $P_{m}(0.3)$, $G_{\text{max}}(20)$, $T_{d}(10^{4})$, $Itmax_{\text{MSA}}(50)$, $F_{r}(0.7)$, $\epsilon(10^{-3})$, $N_{\text{MPSO}}(200)$, $Itmax_{\text{MPSO}}(20)$, and $v_{0}(0)$. Based on the evaluation of small cases, it can be concluded that the proposed algorithms were able to solve all problems with their complexity, although there were varying levels of performance. The comparison of each algorithm showed that the MGA is the best algorithm, followed by MSA, MPSO, SH-2, SH-4, SH-1 and SH-3. For the future research, there should be a focus on the development of algorithms based on other metaheuristic approaches, such as ant colony optimization (ACO) or other evolutionary methods.

This problem has a lot of potential to be applied and developed, it is possible to add some practical constraints. For example, sequence-dependent setup times, parallel assembly lines or flowshop assembly lines, objective functions related to due date, energy consumption or machine utility levels, as well as multiple objective functions. Moreover, various optimization models and heuristics also have the potential to be developed. This
can significantly improve the quality and speed of the solution search process. We expect this can be a trigger for ideas for more realistic future work.

**Disclosure statement**

No potential conflict of interest was reported by the author(s).

**ORCID**

Ikhlasul Amallynda http://orcid.org/0000-0002-1706-2000

Budi Santosa http://orcid.org/0000-0003-1753-9603

**References**

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2), 182–197. https://doi.org/10.1109/4235.996017

Duan, W., Li, Z., Ji, M., Yang, Y., Wang, S., & Liu, B. (2016). A hybrid estimation of distribution algorithm for distributed permutation flowshop scheduling with flowline eligibility. 2016 IEEE Congress on Evolutionary Computation (CEC), 2581–2587. https://doi.org/10.1109/CEC.2016.7744111

Elmaraghy, H., Schuh, G., Elmaraghy, W., Piller, F., Schönsleben, P., Tseng, M., & Bernard, A. (2013). Product variety management. *CIRP Annals - Manufacturing Technology*, 62(2), 629–652. https://doi.org/10.1016/j.cirp.2013.05.007

Erenay, F. S., Sabuncuoglu, I., Toptal, A., & Tiwari, M. K. (2010). New solution methods for single machine bicriteria scheduling problem: Minimization of average flowtime and number of tardy jobs. *European Journal of Operational Research*, 201(1), 89–98. https://doi.org/10.1016/j.ejor.2009.02.014

Fernandez-Viagas, V., & Framinan, J. M. (2015). A bounded-search iterated greedy algorithm for the distributed permutation flowshop scheduling problem. *International Journal of Production Research*, 53(4), 1111–1123. https://doi.org/10.1080/00207543.2014.948578

Framinan, J. M., & Perez-Gonzalez, P. (2017). The 2-stage assembly flowshop scheduling problem with total completion time: Efficient constructive heuristic and metaheuristic. *Computers & Operations Research*, 88 C , 237–246. https://doi.org/10.1016/j.cor.2017.07.012

Framinan, J. M., Perez-Gonzalez, P., & Fernandez-Viagas, V. (2019). Deterministic assembly scheduling problems: A review and classification of concurrent-type scheduling models and solution procedures. *European Journal of Operational Research*, 273(2), 401–417. https://doi.org/10.1016/j.ejor.2018.04.033

Glover, F., & Sörensen, K. (2015). Metaheuristics. *Scholarpedia*, 10(4), 6532. https://doi.org/10.4249/scholarpedia.6532

Hariri, A. M. A., & Potts, C. N. (1997). A branch and bound algorithm for the two-stage assembly scheduling problem. *European Journal of Operational Research*, 103(3), 547–556. https://doi.org/10.1016/S0377-2217(96)00312-8

Hatami, S., Ruiz, R., & Andrés-Romano, C. (2013). The distributed assembly permutation flowshop scheduling problem. *International Journal of Production Research*, 51(17), 5292–5308. https://doi.org/10.1080/00207543.2013.807955

Hatami, S., Ruiz, R., & Andrés-Romano, C. (2015). Heuristics for a distributed parallel machine assembly scheduling problem with eligibility constraints. *2015 International Conference on Industrial Engineering and Systems Management (IESM)*, IEEE, 145–153. https://doi.org/10.1109/IESM.2015.7380149
Hatami, S., Ruiz, R., & Romano, C. (2015). The distributed assembly parallel machine scheduling problem with eligibility constraints. *International Journal of Production Management and Engineering, 3*(1), 13. https://doi.org/10.4995/ijpme.2015.3345

Hoover, S. V., & Perry, R. (1989). *Simulation: A Problem-Solving Approach*, Pearson College Div. Karimi, N., & Davoudpour, H. (2014). A high performing metaheuristic for multi-objective flowshop scheduling problem. *Computers & Operations Research, 52*(PB), 149–156. https://doi.org/10.1016/j.cor.2014.01.006

Law, A., & Kelton, D. (2000). *Simulation modeling and analysis*, McGraw-Hill Education.

Lee, C.-Y., Cheng, T. C. E., & Lin, B. M. T. (1993). Minimizing the Makespan in the 3-machine assembly-type flowshop scheduling problem. *Management Science, 39*(5), 616–625. https://doi.org/10.1287/mnsc.39.5.616

Lenstra, J. K., Rinooy Kan, A. H. G., & Brucker, P. (1977). Complexity of machine scheduling problems. In P. L. Hammer, E. L. Johnson, B. H. Korte, & G. L. B. T.-A. of D. M. Nemhauser (Eds.), *Studies in Integer Programming* (Vol. 1, pp. 343–362). Elsevier. https://doi.org/10.1016/S0167-5060(08)70743-X

Li, J., Bai, S.-C., Duan, P., Sang, H., Han, -Y.-Y., & Zheng, Z. (2019). An improved artificial bee colony algorithm for addressing distributed flow shop with distance coefficient in a prefabricated system. *International Journal of Production Research, 57*(22), 1–21. https://doi.org/10.1080/00207543.2019.1571687

Lin, J., Wang, Z.-J., & Li, X. (2017). A backtracking search hyper-heuristic for the distributed assembly flow-shop scheduling problem. *Swarm and Evolutionary Computation, 36*(C), 124–135. https://doi.org/10.1016/j.swevo.2017.04.007

Lin, J., & Zhang, S. (2016). An effective hybrid biogeography-based optimization algorithm for the distributed assembly permutation flow-shop scheduling problem. *Computers & Industrial Engineering, 97*(C), 128–136. https://doi.org/10.1016/j.cie.2016.05.005

Moon, C., Kim, J., & Hur, S. (2002). Integrated process planning and scheduling with minimizing total tardiness in multi-plants supply chain. *Computers & Industrial Engineering, 43*(1), 331–349. https://doi.org/10.1016/S0360-8352(02)00078-5

Naderi, B., & Ruiz, R. (2010). The distributed permutation flowshop scheduling problem. *Computers & Operations Research, 37*(4), 754–768. https://doi.org/10.1016/j.cor.2009.06.019

Naderi, B., & Ruiz, R. (2014). A scatter search algorithm for the distributed permutation flowshop scheduling problem. *European Journal of Operational Research, 239*(2), 323–334. https://doi.org/10.1016/j.ejor.2014.05.024

Pan, Q.-K., Gao, L., Wang, L., Liang, J., & Li, X.-Y. (2019). Effective heuristics and metaheuristics to minimize total flowtime for the distributed permutation flowshop problem. *Expert Systems with Applications, 124*(C), 309–324. https://doi.org/10.1016/j.eswa.2019.01.062

Pan, Q.-K., Gao, L., Xin-Yu, L., & Jose, F. M. (2019). Effective constructive heuristics and meta-heuristics for the distributed assembly permutation flowshop scheduling problem. *Applied Soft Computing, 81*(C), 105492. https://doi.org/10.1016/j.asoc.2019.105492

Prugel-Bennett, A. (2010). Benefits of a population: Five mechanisms that advantage population-based algorithms. *Evolutionary Computation, IEEE Transactions On, 14*(4), 500–517. https://doi.org/10.1109/TEVC.2009.2039139

Qian, B., Wang, L., Huang, D., Wang, W., & Wang, X. (2009). An effective hybrid DE-based algorithm for multi-objective flow shop scheduling with limited buffers. *Computers & Operations Research, 39*(1), 209–233. https://doi.org/10.1016/j.cor.2007.08.007

Ross, J. A. (1988). Controlling variables: A meta-analysis of training studies. *Review of Educational Research, 58*(4), 405–437. https://doi.org/10.3102/00346543058004405

Ruiz, R., Pan, Q.-K., & Naderi, B. (2019). Iterated Greedy methods for the distributed permutation flowshop scheduling problem. *Omega, 83*(C), 213–222. https://doi.org/10.1016/j.omega.2018.03.004

Santosa, B., & Willy, P. (2011). *Metoda Metaheuristik: Konsep dan Implementasi* (Surabaya: Guna Widya).
Shao, W., Pi, D., & Shao, Z. (2019). Local search methods for a distributed assembly no-idle flow shop scheduling problem. *IEEE Systems Journal, 13*(2), 1945–1956. https://doi.org/10.1109/JSYST.2018.2825337

Sheikh, S., Komaki, G. M., & Kayvanfar, V. (2018). Multi objective two-stage assembly flow shop with release time. *Computers & Industrial Engineering, 124* 2018, 276–292. https://doi.org/10.1016/j.cie.2018.07.023

Sung, C. S., & Juhn, J. (2009). Makespan minimization for a 2-stage assembly scheduling problem subject to component available time constraint. *International Journal of Production Economics, 119*(2), 392–401. https://doi.org/10.1016/j.ijpe.2009.03.012

Wang, K., Li, Z., Duan, W., Feng, X., & Liu, B. (2017). Variable neighborhood based memetic algorithm for Just-in-Time distributed assembly permutation flowshop scheduling. *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC) (IEEE),* 3700–3704. https://doi.org/10.1109/SMC.2017.8123208

Wang, S.-Y., & Wang, L. (2015). An estimation of distribution algorithm-based memetic algorithm for the distributed assembly permutation flow-shop scheduling problem. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 46*(1), 1. https://doi.org/10.1109/TSMC.2015.2416127

William G, C., & Gertrude M, C. (1992). Experimental Designs (2nd ed.). John Wiley & Sons, Inc.

Xiong, F., Xing, K., Wang, F., Lei, H., & Han, L. (2014). Minimizing the total completion time in a distributed two stage assembly system with setup times. *Computers & Operations Research, 47*(2), 92–105. https://doi.org/10.1016/j.cor.2014.02.005

Yokoyama, M. (2001). Hybrid flow-shop scheduling with assembly operations. *International Journal of Production Economics, 73*(2), 103–116. https://doi.org/10.1016/S0925-5273(00)00169-9

Yokoyama, M., & Santos, D. L. (2005). Three-stage flow-shop scheduling with assembly operations to minimize the weighted sum of product completion times. *European Journal of Operational Research, 161*(3), 754–770. https://doi.org/10.1016/j.ejor.2003.09.016

Zhang, X., Li, X., & Yin, M. (2020). An enhanced genetic algorithm for the distributed assembly permutation flowshop scheduling problem. *International Journal of Bio-Inspired Computation, 15*(2), 113. https://doi.org/10.1504/IJBIC.2020.10028014