Journal of Quality Engineering and Production Optimization

Volume 9, Issue 1, Winter & Spring 2024

http://jqepo.shahed.ac.ir

Research Paper

DOI: 10.22070/jqepo.2024.17794.1261

Production-assembly problem with parallel machines in three steps and in distributed factories

Mohsen Torkashvand [1], **Fardin Ahmadizar** [1*]

[1] *Department of Industrial Engineering, University of Kurdistan, Sanandaj, Iran*

*** Corresponding Author:** *Fardin Ahmadizar (Email: f.ahmadizar@uok.ac.ir)*

Abstract *–Distributed factories represent a type of manufacturing system in which production is spread out across multiple geographically dispersed locations. This approach offers several advantages, including reduced transportation costs, improved responsiveness to customer demand, increased flexibility and enhanced supply chain resilience. The production-assembly flow shop problem with three stages is a scheduling problem focused on optimizing the sequence of jobs to be processed on a set of machines. In this problem, the first and third stages involve dedicated parallel machines, meaning that each job is assigned to a specific machine and cannot be processed on any other. The second stage consists of identical parallel machines, where all machines are functionally equivalent and capable of processing any job. A model is presented for minimizing total tardiness times. Since the problem under investigation is NP-hard, solving it exactly is either impossible or highly time-consuming (depending on the processor's capability) for large instances. Consequently, the Hybrid Biogeography-Based Optimization Dominance Rules (HBBO) algorithm is proposed to address the problem in larger instances. This algorithm is an enhanced version of the Biogeography-Based Optimization (BBO) algorithm, incorporating dominance rules. The Taguchi method has been employed to determine appropriate parameter values. The results obtained from the model and algorithm demonstrate the algorithm's acceptable efficiency and the inclusion of dominance rules has further improved the outcomes.*

Keywords*– Biogeography-Based Optimization, Production and assembly, Scheduling, dominance rules, Total tardiness times.*

I. INTRODUCTION

The ability to provide products according to the delivery time, variety, and volume of customer requests is a key competitive advantage for businesses. Satisfied customers are the foundation of any successful business. When a company consistently demonstrates the ability to meet customer needs in flexible and efficient ways, a sense of trust is fostered, reinforcing the positive qualities of the brand. This reliability, in turn, enhances customer loyalty, making them more likely to return for future purchases. Supply chain agility plays a significant role in reducing costs associated with inventory management and warehousing. A detailed investigation into product production can serve as a valuable tool for businesses striving to meet customer demand while reducing costs. Identifying and addressing inefficiencies in production processes is crucial for achieving and maintaining supply chain agility. This enables improvements in efficiency, reduction of waste, and ultimately, enhancement of the company's bottom line.

Manuscript Received: 17-December-2023 & Revised: 21-March-2024 & Accepted: 2-May-2024
ISSN: 2423-3781

Planning and scheduling are critical operations in both service and manufacturing industries. These processes help organizations allocate resources efficiently, optimize workflows, and manage time effectively, leading to increased overall productivity (Zhao et al., 2019). Effective planning offers significant benefits for both service providers and recipients. By ensuring efficient resource allocation, streamlined processes, and effective time management, planning creates a positive ripple effect that benefits both parties involved in the service experience:

- Service providers: Experience cost reduction, increased productivity, and improved customer satisfaction.
- Service recipients: Benefit from shorter wait times, enhanced service quality, better predictability, and potentially lower costs.

In many production environments, a common method for organizing production is serial production, also referred to as flow shop (Rostami & Shad, (2020); Samarghandi & Firouzi Jahantigh, (2019)). When identical parallel machines are used in one of the steps, this system is referred to as a hybrid flow shop (Rastgar et al., 2021). The process of manufacturing products or equipment typically involves producing individual components, which are then assembled to create the final product. As such, the start of a job depends on the completion of its preceding Jobs, a common feature in manufacturing and service industries. To achieve the desired outcome, these jobs are scheduled concurrently using different resources. Production-assembly scheduling problems exemplify scenarios where the desired outcome is achieved by scheduling various jobs (component assembly) using multiple resources (workers, machines) simultaneously. (Framinan & Gonzalez, 2018).

In this article, a three-step production-assembly problem is considered across parallel factories. As shown in Figure 1, in the first step of each factory, production operations are performed by dedicated parallel machines, where different components of a product are produced separately. Once the components are manufactured, they can be assembled using any of the machines in the second step, as this stage employs identical parallel machines. Finally, in the third step, dedicated parallel machines are used, and the remaining operations of a product can be completed by one of these machines.

The concept of dedicated parallel machines has numerous real-world applications, one of which is fire engine assembly. A fire engine is a complex vehicle composed of various parts working together to assist firefighters in their crucial tasks. Key components include the body, chassis, and engine. Each of these components requires specific manufacturing processes, and dedicated parallel machines can be employed to perform these processes efficiently and effectively. The framework, chassis, and engine are produced in the first step, followed by their assembly in the second step (lee et al., 1993). Similary, the process of printing invoice pages can be divided into three main steps: preparation of template, printing of pages, and assembly of the invoices. This system ensures that invoices are accurately generated and delivered to customers on time. The process of printing invoice pages can be considered as an assembly flow shop, where each page represents a component, and the invoice is the final product. The two steps of printing and assembly can be viewed as parallel machines, each responsible for processing a specific set of components. Assembly flow shop problems can also be found in various real-world environments, including the automobile industry, electrical system testing, and airplane maintenance (Wagneur & Sriskandarajah, 1993).

The mathematical model (According to the sequence of jobs processing) and the branch-and-bound (B&B) method are effective techniques for solving assembly flow shop problems, especially in small dimensions. These methods can efficiently identify the optimal solution, which minimizes the overall processing time or other relevant objective functions. A single machine with minimizing the sum of earliness and tardiness is a strongly NP-hard problem (Esmaeili et al., 2021). The production-assembly flow shop scheduling problem with two machines in the first step and one machine in the second step is NP-hard (lee et al., 1993). The expanded problem, which includes more steps, more machines per step, and more factories, is more complicated than the original problem presented by Lee et al. (1993). This additional complexity increases the difficulty of solving the problem, making it more likely to be NP-hard. By exploring alternative approaches, organizations can address the scheduling issues of complex production-assembly processes even when finding the absolute optimal solution becomes impractical or impossible. The focus shifts from finding the absolute best solution to identifying efficient and feasible solutions that meet production goals and resource constraints in a timely manner. When dealing with large-scale NP-hard problems, approximation algorithms become crucial tools for navigating the challenges of finding optimal solutions. A new metaheuristic algorithm called the Hybrid BBO (HBBO) algorithm has been proposed to address the expanded production-assembly flow shop scheduling problem in a large-scale. This algorithm combines the strengths of the BBO algorithm with dominance rules to effectively explore the solution space and find near-optimal or even optimal solutions.

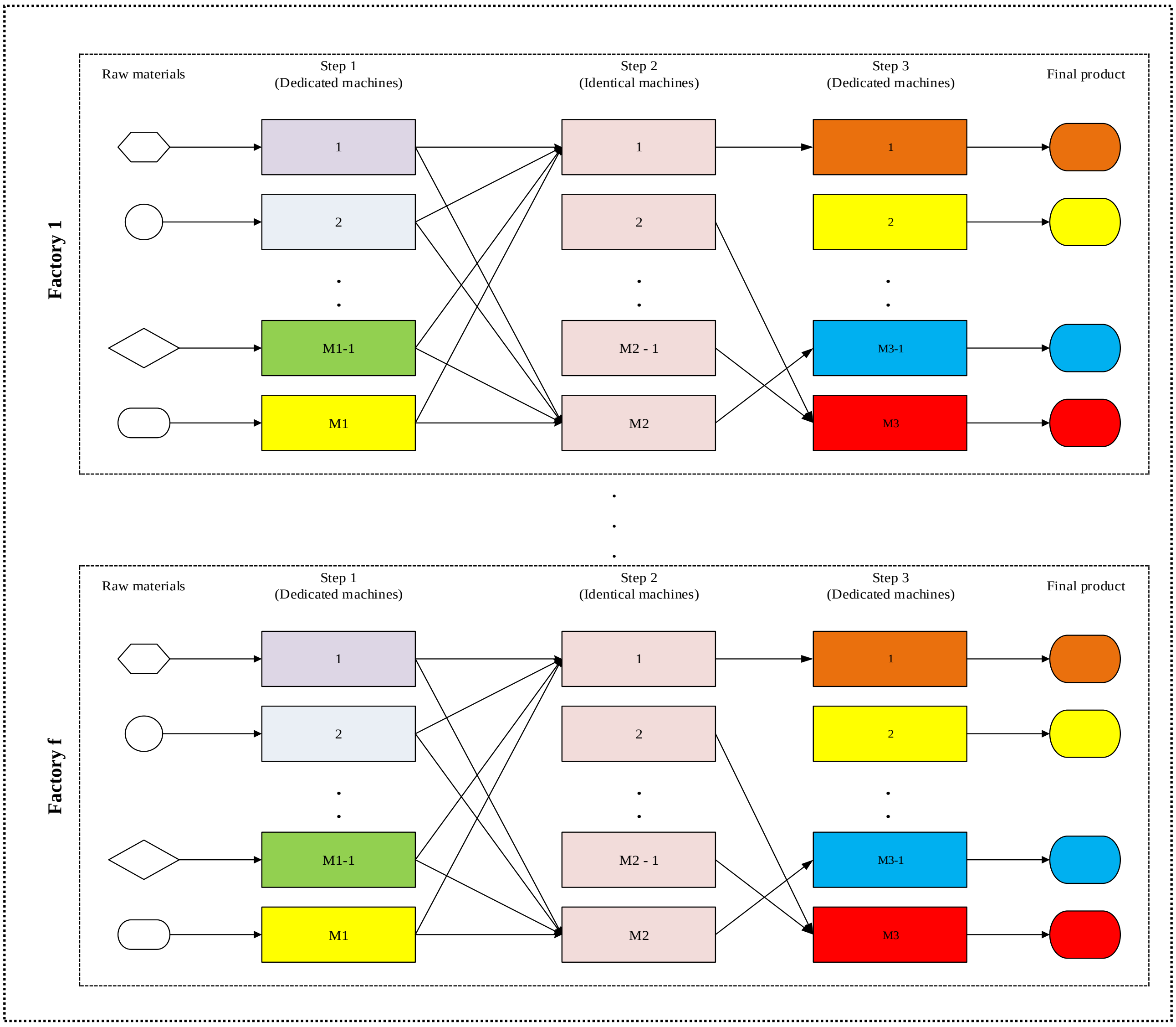

Fig. 1. Arrangement of machines in a three-step production-assembly problem across parallel factories

Based on the aforementioned discussion, the existing gaps in the production-assembly flow shop scheduling problem can be identified, along with the contributions made by the proposed solution. The contributions of the problem will be examined in different parts of the article. The motivation of presenting this article is to cover the existing gaps and to present a problem that is applicable in the real world. To achieve this goal, exact and approximate solution methods have been used in each dimension of the problem.

According to the aforementioned, the innovation of the problem includes the following:

1. Presenting a new mathematical model based on the sequence of jobs processing
2. Considering different parallel machines in the third step of multi-factory production-assembly problem
3. Providing an improved meta-heuristic algorithm to solve the problem in large dimensions
4. The utilization of parallel machines in all three steps of the three-step production-assembly problem in parallel factories

The next sections are presented as follows: the literature review of problem and the mathematical model in the second and third sections, respectively, the solution method in the fourth section, the results of calculation in the fifth section, and the third stage includes the conclusion.

II. LITERATURE REVIEW

The production-assembly problem typically involves two main phases: production and assembly. Various aspects of production-assembly scheduling problems have been continuously explored by researchers and practitioners, with different configurations of machines, numbers of steps, and the distribution of factories considered to address the diverse challenges of real-world manufacturing scenarios. The production-assembly problem was introduced in 1995 by Potts et al. (1995). A scheduling problem consisting of two steps was examined by them. In the first step, the production of different components is performed by dedicated parallel machines, and in the second step, the assembly of the produced components is carried out by a machine. The makespan was considered the objective function. The presented articles in this field are divided into two major categories: single-factory and multi-factory mode. In the following, the single-factory mode is presented first, then the multi-factory mode. For single-factory, the articles are as follows. Wagneur & Sriskandarajah, (1993) considered the scheduling problem with n jobs and m separate machines, such that no prerequisite restriction is imposed for different activities of a job, and they are performed simultaneously by all machines; however, a job limitation is placed on machines, and one machine cannot run more than one activity at a time. They investigated the objective functions, including maximum completion time, total completion time, maximum tardiness time, total tardiness time, and the number of tardy jobs. A heuristic algorithm was presented by Leung et al. (2005) to minimize the weighted sum of tardiness times and the weighted sum of order completion times in a scheduling problem with dedicated parallel machines, where each machine has the ability to process a specific job. An order scheduling problem was investigated by Lee (2013). Each order consists of m parts that are produced by m dedicated parallel workshops. The sum of delay times was minimized as the objective function. Some heuristic methods were presented to improve the lower bound of a BBO algorithm.

Wu et al. (2018) considered the orders scheduling problem, where different orders are processed by multi machines and any order contains several components. Each component is processed by a dedicated machine, and order completion time is defined as when all components of an order are completed.

The flow shop scheduling problem in production-assembly mode was presented by Wu et al. (2019), Zou et al. (2020), and Wu et al. (2020). The first step includes two independent machines, and the second step includes one machine. Two components are produced in the first step and then assembled in the second step. Zou et al., (2020) focused on minimizing the maximum completion time of jobs. Dominance properties, lower bounds, SA, and cloud theory-based simulated annealing algorithms were developed to improve the BBO algorithm. Wu et al. (2019) minimized the total time to complete the jobs. A BBO algorithm and four meta-heuristic algorithms, including cloud theory-based simulated annealing, genetic algorithms, iterated greedy (IG) algorithms, and artificial bee colonies, were presented to solve the problem. Wu et al. (2020) minimized the makespan of jobs. A BBO algorithm and three meta-heuristic algorithms, including simulated annealing (SA), dynamic differential evolution (DDE), and cloud theory-based simulated annealing (CSA), were employed to solve the problem.

Al-Anzi and Allahverdi (2009), Torabzadeh and Zandieh (2010), Tian et al. (2013), Yan et al. (2014), Komaki and Kayvanfar (2015), Allahverdi et al., (2016), Jung et al., (2017), Sheikh et al. (2018), Chung and Chen (2019) and Hosseini et al. (2022) considered two-step production-assembly with independent parallel machines in the first step and one machine in the second step.

Three heuristic algorithms were considered by Al-Anzi and Allahverdi (2009) to minimize the weighted sum of maximum delay and maximum completion time. The CSA algorithm was proposed by Torabzadeh and Zandieh (2010) to minimize the weighted sum of the average completion time and the maximum completion time of jobs. Tian et al. (2013) applied the discrete particle swarm optimization (PSO) algorithm with the objective function of the average completion time and the maximum completion time of the jobs. The hybrid VNS algorithm was employed by Yan et al., (2014) to minimize the weighted sum of the average completion times and the maximum completion time. Komaki and Kayvanfar (2015) introduced Gray Wolf Optimizer (GWO) algorithm to minimize the makespan. Several algorithms, including genetic, simulated annealing, and insertion algorithms, were utilized by Allahverdi et al. (2016) to minimize total delay times. Jung et al. (2017) presented the genetic algorithm to minimize maximum time to complete jobs. Sheikh et al. (2018) considered total time to complete the jobs, the total delay time and the maximum time to complete the jobs as objective function. Non-dominated Sorting Genetic Algorithm (NSGA-III) and Multi-Objective PSO (MOPSO) algorithms were provided to solve the problem. Chung and Chen (2019) presented meta-heuristic algorithm based on Immunoglobulin-based Artificial Immune System in order to solve the problem in large dimensions. The objective function of the problem is minimizing the weighted sum of early and delay job times. Hosseini et al. (2022) presented some heuristic algorithms to solve the problem with the makespan.

Mozdgir et al. (2013), Basir et al. (2018), Mahabadpour et al. (2020), Zhang and Tang (2021) addressed the scheduling problem so that m machines produce m separate parts of a product in the first step and the same parallel machines perform the assembly of the produced parts in the second step. Mozdgir et al. (2013) minimized the weighted sum of the average completion time and the maximum completion time of the jobs using the heuristic VNS algorithm. A genetic algorithm was employed by Basir et al. (2018) to minimize the weighted average of job lateness and total delivery costs. The maximum time to complete the jobs was minimized by Mahabadpour et al. (2020) through a heuristic algorithm. Zhang and Tang (2021) minimized the total completion time of jobs with utilization of heuristic algorithm.

Maleki-darounkolaei et al. (2012), Fattahi et al. (2014), Shoaardebili and Fattahi (2015), and Campos et al. (2017) considered the three-step production-assembly scheduling problem including production, transportation and assembly. In the first step, dedicated parallel machines are responsible for producing different parts of a product. The produced parts are collected by a single machine and transferred to the second step, where they are assembled. In the third step, the assembly is completed.

Maleki-darounkolaei et al. (2012) proposed the weighted average of job completion times. A meta-heuristic algorithm based on SA was used to solve the problem. Fattahi et al. (2014) minimized sum of early and tardy times and the maximum completion time simultaneously by using the Elitist Non-dominated Sorting Genetic Algorithm (NSGA-II). Wang et al. (2016) considered the three-step production-assembly scheduling problem including production, assembly and transportation. Different parallel machines produce different parts of a product in the first step. The produced parts are assembled by a machine in the second step, and different product are transported and delivered to customers in the third step. The objective function is the weighted sum of total delivery costs and average delivery time. A hybrid meta-heuristic algorithm (GA and VNS) was used to solve the problem. Shoaardebili and Fattahi (2015) investigated the weighted sum of completion times as well as the weighted sum of early and late jobs. Meta-heuristic algorithms were used to solve the problem. Campos et al. (2017) presented the idea of minimizing the total delay time of jobs. A heuristic algorithm using VNS was presented to solve the problem. Deng et al. (2021) considered the production-assembly scheduling problem in three steps, so that there are parallel machines in the first and second steps and one machine in the third step. Dedicated parallel machines perform the production operation in the first step, and identical parallel machines perform the transportation operation of the produced parts in the second step. The assembly operation takes place in the third step. The objective function involves minimizing energy consumption, which was achieved using the VNS algorithm.

The reviewed articles were for single-factory state. Articles for multi-factory state are presented in the following.

Lei and Liu (2020) considered the scheduling problem including the allocation of jobs to factories. Separate identical factories were proposed, each containing different parallel machines that perform production operations. To minimize the maximum time to complete a job, the division-based artificial bee colony (DABC) algorithm was used. Hatami et al. (2013) and Liao et al. (2015) considered two steps for production-assembly problem. Identical factories contain different parallel machines that perform production operations in the first step. A machine assembles the produced components in the second step. Heuristic algorithms were applied to minimize the maximum completion time.

Zhang and Xing (2018), Ochi and Driss (2019), Li et al. (2021), Pourhejazy et al. (2021) and Pourhejazy et al. (2023) considered the production-assembly scheduling problem in multi factories such that there are dedicated parallel machines in the first step of each factory and one machine that performs assembly operations in the second step.

Zhang and Xing (2018) used a memetic algorithm (MA) based on developed social spider optimization (SSO) in order to minimize the total time to complete the jobs. Ochi and Driss (2019) presented a bounded-search IG algorithm to minimize the maximum completion time. Li et al. (2021) proposed an imperialist competitive algorithm to minimize the maximum time to complete the jobs. Pourhejazy et al. (2021) presented an IG algorithm to solve the problem with minimizing the maximum time to complete of jobs. Pourhejazy et al. (2023) proposed a extension of the Iterated Greedy algorithm to solve this understudied distributed two-stage production-assembly scheduling problem.

Wang et al. (2023) considered distributed two-stage hybrid flow shop and proposed an improved Non-dominated Sorting Genetic Algorithm-II (INSGA-II) to solve it. Fernandez-Viagas et al. (2018), Chen et al. (2021), Lei & Su, (2023), Zhao et al. (2023), Cui et al. (2023), Li et al. (2023) and Yu et al. (2024) presented the flow shop scheduling for identical parallel factories with m machines in series in each factory. Fernandez-Viagas et al. (2018) propose eighteen constructive heuristics to obtain high-quality solutions in reasonable CPU times and an iterative improvement algorithm to further refine the so-obtained solutions to minimize total flowtime. Chen et al. (2021) considered an IG algorithm to solve the problem and the objective function of the problem is the sum of the times to complete the jobs. Lei & Su, (2023) proposed a multi-class teaching–learning-based optimization (MTLBO) to minimize makespan and maximum tardiness simultaneously. Zhao et al. (2023) presented brain storm optimization algorithm for minimizing the maximum assembly completion time minimizing the total energy consumption (TEC) and achieving resource allocation balanced. Cui et al. (2023) presented a greedy job insertion inter-factory neighborhood structure and a new move evaluation method to ensure the efficiency of neighborhood movement. Also, an improved multi-population genetic algorithm (IMPGA) is proposed to solve the DHHFSP with makespan. Li et al. (2023) suggested that to minimize the maximum completion time, a hyper-heuristic three-dimensional estimation of the distribution algorithm (HH3DEDA) was used to solve it. To solve this problem with total tardiness, Yu et al. (2024) presented a mixed-integer linear programming (MILP) model and a knowledge-based iterated greedy algorithm (KBIG).

Jia et al. (2023) presented a multi-population memetic algorithm (MPMA) with Q-learning (MPMA-QL) to address a distributed assembly hybrid flow shop scheduling problem with flexible preventive maintenance (DAHFSP-FPM). A mixed integer linear programming (MILP) model targeted at the minimal makespan.

Hatami et al. (2013), Li et al. (2015), Lin and Zhang (2016), Ji et al. (2016), Gonzalez-Neira et al. (2017), Yang et al. (2017), Pan et al. (2018), Shao et al. (2018), Pan et al. (2019), Ferone et al. (2020), Huang et al. (2021), Shao et al. (2020), Zhang et al. (2021), Song and Lin. (2021), Zhao et al. (2021) and Zhao et al. (2021) and Ying and Lin (2023) and Huang et al. (2023) considered the two-step production-assembly problem. Separate identical factories contain a flow shop, and perform production operations in the first step. A machine assembles the produced components in the second step.

Hatami et al. (2013), Gonzalez-Neira et al. (2017), and Ferone et al. (2020) aimed to minimize the maximum completion time by employing meta-heuristic algorithms to solve the problem. Li et al. (2015), Lin and Zhang (2016) and Ji et al. (2016) used GA, BBO and (PSO and SA), respectively, to minimize the maximum time to complete the jobs. Yang et al. (2017) used of the scatter search based MA (SS-MA) in order to minimize the total tardiness time of the jobs. Pan et al. (2018) attempted to minimize the total completion time for the two-step production-assembly problem and used the fruit fly optimization (FFO) algorithm to solve the problem. Shao et al. (2018) proposed three constructive heuristics based on a new job assignment rule and suggested two simple meta-heuristics, including iterated local search (ILS) and variable neighborhood search (VNS) to minimize makespan. Pan et al. (2019) presented heuristic methods in order to minimize the maximum completion time of jobs. Huang et al. (2021) used an improved iterative greedy algorithm to solve the two-step assembly problem by minimizing total job completion time. Shao et al. (2020) used a heuristic algorithm in order to minimize the makespan for two-step production-assembly problem. Zhang et al. (2021) state that the makespan is minimized by the utilization of the matrix-cube-based estimation of the distribution algorithm (MCEDA). Song and Lin. (2021) considered an improved genetic algorithm to minimize the makespan for the two-step production-assembly problem. Zhao et al. (2021) presented heuristic algorithms in order to minimize the maximum time to complete the jobs. Zhao et al. (2021) used the water wave optimization algorithm to minimize the maximum time to complete the jobs. Ying and Lin (2023) consider a mixed-integer linear programming (MILP) model and a novel metaheuristic algorithm, called the Reinforcement Learning Iterated Greedy (RLIG) algorithm, to minimize the makespan of this problem. Huang et al. (2023) proposed an effective memetic algorithm (EMA) to minimize total tardiness criterion.

Zheng and Wang (2021) and Wang et al. (2022) considered a three-step production problem so that parallel factories, including flow shop perform production operation in the first step. The produced parts are transported and transferred to next step, in the second step. The produced parts are assembled by one machine in the third step. Zheng and Wang (2021) minimized the makespan by the bat optimization algorithm. Wang et al. (2022) considered total delay time of the jobs as objective function, that is minimized by the Q-learning-based artificial bee colony algorithm (QABC). Zhang et al. (2018) considered the production-assembly problem in identical parallel factories. In each factory, there is a flow shop production line, and finally, parallel machines perform assembly operations of the produced parts in all factories. Heuristic and meta-heuristic methods were applied to minimize makespan.

Yang and Xu (2021) considered three-step production-assembly problem so that parallel factories, including flow shop perform production operations in the first step. The produced parts by first step factories are assembled by parallel factories, including one machine in the second step. Products are grouped separately for each customer in the third step. The objective function is the sum of delay times and delivery cost, that is minimized by seven algorithms.

Torkashvand et al. (2022) studied a production-assembly problem in three steps and in identical parallel factories with the objective function of makespan, so that there are dedicated parallel machines in the first step, one machine in the second step, and identical parallel machines in the third step. An improved GA was developed to solve the problem. Torkashvand and Ahmadizar (2024) addressed a similar problem, focusing on minimizing total completion time. Parallel machines were used in both the first and second steps, with a single machine handling the third step. An improved PSO algorithm was proposed to solve the problem.

The machine positioning for each problem was provided by Framinan et al. (2019) with the formulation (1).

$$\alpha_1 \rightarrow \alpha_2|\beta|\gamma \tag{1}$$

In the notation above, α_1 represents the environment of the machines in the production step. In this environment, all operations are performed before the assembly operation. α_2 demonstrates the environment of the machines in the assembly step. Assembly and subsequent operations take place in this part. Different limitations of the problem are displayed in segment β and finally different types of used objective function in the problem are presented in segment γ. Framinan et al. (2019) discussed the different positions of machines for production and assembly in detail. When additional operations are required after production and assembly, the notation is modified by including α_3 in the formulation, as shown in formulation (2):

$$(\alpha_1 \to \alpha_2) \to \alpha_3|\beta|\gamma \tag{2}$$

If the expression within parentheses represents a factory that combines production and assembly, and $\alpha_3 = 0$, it implies that no further operations are performed after the production and assembly stages. This configuration indicates that the factories operate in parallel, with each producing a final product independently. Based on the literature, the presented articles can be divided into two major categories: one-factory and multi-factory. In the case of one-factory, for most articles, there are dedicated parallel machines in the first step and 0, 1 or m machines in the second step according to Table I. For the three-step mode, the number of machines in the third step is equal to 1. The most complex state was presented by Deng et al (2021) that there are dedicated parallel machines in the first step, identical parallel machines in the second step, and one machine in the third step. In the multi-factory state, the problems are divided into two categories according to Table II. In the first category, the jobs are not processed in parallel factories at least in one step and all of them pass through one machine commonlly, but all the steps exist in one factory and each job is processed only in one factory in the second category.

The most complex state belongs to Yang and Xu (2021) and Torkashvand et al. (2022). The presented article by Yang and Xu (2021) pertains to the first category, so there is a flow shop in all parallel factories in the first step. There are identical parallel machines in the second step, and there is one machine in the third step. The presented article by Torkashvand et al. (2022) pertains to the second category and each factory has three steps: dedicated parallel machines in the first step, one machine in the second step, and identical parallel machines in the third step.

Table I. Summary of literature for single-factory state

Row	Referece	Problem	Obj	Algorithm
1	Wagneur and Sriskandarajah, (1993)	DPm--0	Maximum completion time (C_{max}), total completion time (TCT), maximum lateness (L_{max}), total tardiness time (TT) and number of tardiness (TU)	Heuristic algorithm
2	Leung et al. (2005)	DPm--0	weighted TT (WTT) and TCT	Heuristic algorithm
3	Lee (2013)	DPm--0	Sum of delay times	B&B
4	Wu et al. (2018)	DPm--0	TCT, L_{max}	Hybrid IG algorithm and a PSO algorithm
5	Wu et al. (2019)	DP2--1	TCT	B&B, cloud theory-based SA, GA, IG algorithms and artificial bee colonies
6	zou et al. (2020)	DP2--1	C_{max}	B&B, SA and cloud theory-based SA
7	Wu et al. (2020)	DP2--1	C_{max}	B&B, DDE, SA and CSA
8	Potts et al. (1995)	DPm--1	C_{max}	Heuristic algorithm
9	Al-Anzi and Allahverdi (2009)	DPm--1	weighted sum of L_{max} and C_{max}	Heuristic algorithm
10	Torabzadeh and Zandieh (2010)	DPm--1	Weighted sum of the average completion time (ACT) and C_{max}	CSA
11	Tian et al. (2013)	DPm--1	ACT and C_{max}	Discrete PSO
12	yan et al. (2014)	DPm--1	Weighted sum of ACT and C_{max}	Hybrid VNS
13	komaki and Kayvanfar (2015)	DPm--1	C_{max}	GWO

Continue Table I. Summary of literature for single-factory state

Row	Referece	Problem	Obj	Algorithm
14	Allahverdi et al. (2016)	DPm--1	TT	GA, refrigeration simulation and insertion algorithm
15	Jung et al. (2017)	DPm--1	C_{max}	GA
16	Sheikh et al. (2018)	DPm--1	TT and C_{max}	NSGA-III and MOPSO
17	Chung and Chen (2019)	DPm--1	Weighted sum of early and delay times	Meta-heuristic algorithm based on Immunoglobulin-based Artificial Immune System
18	Hosseini et al. (2022)	DPm--1	C_{max}	Heuristic algorithm
19	Basir et al. (2018)	DPm--Pm	Weighted average of latness and total delivery costs	GA
20	Mahabadpour et al. (2020)	DPm--Pm	C_{max}	Heuristic algorithm
21	Zhang and Tang (2021)	DPm--Pm	TCT	Heuristic algorithm
22	Maleki-darounkolaei et al. (2012)	DPm--F2	Weighted ACT	Meta-heuristic algorithm
23	Fattahi et al. (2014)	DPm--F2	Sum of early and tardy times and the C_{max}	NSGA-II
24	Wang et al. (2016)	DPm--F2	Weighted sum of total delivery costs and average delivery time	Meta-heuristic algorithm (GA and VNS)
25	Shoaardebili and Fattahi (2015)	DPm--F2	Weighted sum of completion times and weighted sum of early and late	Meta-heuristic
26	Campos et al. (2017)	DPm--F2	TT	Heuristic algorithm using VNS
27	Deng et al. (2021)	DPm--HF2(Pm,1)	Energy consumption	VNS

Table II. Summary of literature for multi-factory state

Row	**Referece**	**Problem**	**Obj**	**Algorithm**
1	Lei and Liu (2020)	(DPm--0)--0	C_{max}	DABC
2	Liao et al. (2015)	(DPm--0)--1	C_{max}	Heuristic algorithms
3	Zhang and Xing (2018)	(DPm--1)--0	TCT	MA
4	Ochi and Driss (2019)	(DPm--1)--0	C_{max}	Bounded-search iterated greedy IG
5	Li et al. (2021)	(DPm--1)--0	C_{max}	Imperialist competitive algorithm
6	Pourhejazy et al. (2021)	(DPm--1)--0	C_{max}	IIG
7	Pourhejazy et al. (2023)	(DPm--1)--0	C_{max}	Iterated Greedy algorith
8	Wang et al. (2023)	(F2--0)--0	total weighted earliness/tardiness	INSGA-II
9	Fernandez-Viagas et al. (2018)	(Fm--0)--0	TCT	Constructive heuristics
10	Chen et al. (2021)	(Fm--0)--0	TCT	IG algorithm
11	Zhao et al. (2023)	(Fm--0)--0	C_{max} and TEC	Brain storm optimisation

Continue Table II. Summary of literature for multi-factory state

Row	Referece	Problem	Obj	Algorithm
12	Li et al. (2023)	(Fm--0)--0	C_{max}	
13	Cui et al. (2023)	(Fm--0)--0	C_{max}	IMPGA
14	& Su,(2023) Lei	(Fm--0)--0	C_{max} and L_{max}	MTLBO
15	Yu et al. (2024)	(Fm--0)--0	TT	KBIG
16	Jia et al. (2023)	(F2--0)--1	C_{max}	MPMA-QL
17	Hatami et al. (2013)	(Fm--0)--1	C_{max}	Meta-heuristic algorithm
18	Li et al. (2015)	(Fm--0)--1	C_{max}	Genetic algorithm
19	Lin and Zhang (2016)	(Fm--0)--1	C_{max}	BBO algorithm
20	Ji et al. (2016)	(Fm--0)--1	C_{max}	PSO and SA
21	Gonzalez-Neira et al. (2017)	(Fm--0)--1	C_{max}	Meta-heuristic algorithm
22	Yang et al. (2017)	(Fm--0)--1	TCT	Scatter search based MA
23	Pan et al. (2018)	(Fm--0)--1	TCT	FFO
24	Shao et al. (2018)	(Fm--0)--1	C_{max}	ILS, VNS
25	Pan et al. (2019)	(Fm--0)--1	C_{max}	Heuristic algorithm
26	Ferone et al. (2020)	(Fm--0)--1	C_{max}	Meta-heuristic algorithm
27	Huang et al. (2021)	(Fm--0)--1	TCT	Improved iterative greedy algorithm
28	Shao et al. (2020)	(Fm--0)--1	C_{max}	Heuristic algorithm
29	Zhang et al. (2021)	(Fm--0)--1	C_{max}	Matrix-cube-based estimation of distribution algorithm
30	Song and Lin. (2021)	(Fm--0)--1	C_{max}	Improved GA
31	Zhao et al. (2021)	(Fm--0)--1	C_{max}	Heuristic algorithm
32	Zhao et al. (2021)	(Fm--0)--1	C_{max}	Water wave optimization algorithm
33	Ying and Lin. (2023)	(Fm--0)--1	C_{max}	RLIG
34	Huang et al. (2023)	(Fm--0)--1	TT	EMA
35	Zheng and Wang (2021)	(Fm--0)--1--1	C_{max}	Bat optimization
36	Wang et al. (2022)	(Fm--0)--1--1	TT	QABC
37	Zhang et al. (2018)	(Fm--0)--Pm	C_{max}	Heuristic and meta-heuristic
38	Yang and Xu (2021)	(Fm--0)--Pm--1	Sum of delay times and delivery cost	Heuristic and meta-heuristic
39	Torkashvand et al. (2022)	(DPm--1--Pm)--0	C_{max}	Improved GA
40	Torkashvand and Ahmadizar (2024)	(DPm-- Pm --1)--0	TCT	Improved PSO
This paper		(DPm--Pm--DPm)--0	TT	HGA, HBBO

Based on the presented practical examples in the real world as well as the problems reviewed in the literature, there are some gaps in the presented problems in this field. This article specifically addresses these gaps, which constitutes one of its key contributions. A three-step production-assembly problem in identical parallel factories is presented, aligning with the insights from the literature review. There are dedicated parallel machines in the first step, identical parallel machines in the second step, and dedicated parallel machines in the third step.

The use of parallel machines in all three steps of the production-assembly problem has not, to our knowledge, been previously examined in either single-factory or multi-factory scenarios. In the most complex case, there is only one machine in least one of the steps (Torkashvand et al., 2022 and Torkashvand and Ahmadizar, 2024). Utilization of identical parallel machines in the second step of a production-assembly process can help to prevent interruptions in the jobs processing and reduce the costs of goods in process (WIP) inventory. By having multiple identical machines operating concurrently, jobs no longer have to wait in a queue for a single machine to become available. This parallel processing significantly increases the throughput of the second step in the production line, allowing jobs to flow through the system more efficiently. This not only reduces the overall processing time for each job but also minimizes idle time in the production line, leading to a smoother operation. With identical machines, you can distribute incoming jobs evenly among them. This helps to prevent any single machine from becoming overloaded and creating a bottleneck that slows down the entire production line. This even distribution also minimizes idle time for machines waiting for new jobs, further improving overall efficiency. Utilizing parallel machines in the second step of a production-assembly process can indeed expedite the assembly operation, thereby contributing to timely product delivery to customers. By enabling multiple components or subassemblies to be assembled simultaneously, parallel machines can significantly reduce the overall assembly time. This accelerated assembly process can directly translate into shorter overall production lead times, ensuring that products are delivered to customers within the specified timeframe. Moreover, the reduced assembly time can also help to minimize inventory levels and associated carrying costs. In the third step, dedicated parallel machines facilitate the production of diverse products, accommodate different packing configurations, or support various transportation methods for parts. These specialized machines are designed to handle specific tasks or product variations, enabling manufacturers to adapt their production processes to meet diverse customer requirements. However, the use of such dedicated parallel machines in all three steps within parallel factories has not been investigated in the literature, with identical machines proposed only by Torkashvand et al. (2022) and Torkashvand and Ahmadizar (2024). Distributing production across multiple factories offers strategic benefits by balancing workloads and preventing individual factories from becoming overloaded. This distribution minimizes bottlenecks and slowdowns that could otherwise impact product quality. Additionally, access to a broader pool of skilled labor and expertise across multiple factories can enhance the overall quality of the final product (Naderi & Ruiz, 2010). In summary, utilizing parallel factories provides manufacturers with greater flexibility, operational efficiency, and a more resilient supply chain. By spreading production across multiple facilities, risks are mitigated, costs are lowered, and product quality is improved. Therefore, this study investigates the novel configuration of using parallel machines across all three steps of the flow shop problem in parallel factories, a configuration not previously explored in the literature.

III. MATHEMATICAL MODEL

In this section, one of the key contributions of the research is examined. The use of a mathematical model, GAMS software, Cplex solver, and the B&B algorithm can be a valuable approach for accurately solving the problem in small dimensions. This combination offers a rigorous and systematic method for identifying optimal or near-optimal solutions.

In this model, there are m_1 machines in the first step, m_2 machines in the second step, and m_3 machines in the third step, where each job is processed on only one machine in a continuous manner. In the first step, dedicated machines are employed, with each machine producing a portion of a job. In the second step, there are identical parallel machines, and the components produced in the first step can be assembled by any of these machines. There are dedicated parallel machines, that perform post-assembly operation in the third step, and each job is processed by only one machine continuously in this step. If the jobs have different delivery dates, the production planning and scheduling should be in such a way that the jobs are delivered to the customer the deadline and delay is minimized. Therefore, the objective function of the problem is to minimize the total tardiness of all jobs. This section presents the mathematical model, including the objective function and relevant constraints.

There are two main modes for determining the sequence of processing jobs in a production-assembly scheduling problem: determining the sequence based on the sequence of processing jobs, determining the sequence based on the position of processing jobs. The appropriate way for determining best approach is to experiment with both methods and see which one produces the best results. In order to provide the appropriate model in terms of achieving the optimal solution faster and occupying minimal memory, in this section two models are presented in different modes and the results are compared. Two modes have been studied in order to provide a suitable model for solvng the problem. The first mode of the model is based on the position of jobs processing and the second mode is based on the sequence of jobs processing on machines. Two modes are presented below and their results are compared in order to select the suitable model.

A. Position based model

In this case, the problem model is presented based on the jobs processing position. The problem model includes the objective function and constraints. The parameters, indexes and decision variables are presented initially, then the mathematical model.

Parameters:

n	Total number of jobs
m_1	The machines number in the first step of every factory
m_2	The machines number in the second step of every factory
m_3	The machines number in the third step of every factory
$Pf_{i,k}$	The processing time of job i on machine k in the first step
ps_i	The processing time of job i on the assembly machine in the second step
pt_i	The processing time of job i in the third step
M	A large positive number

Indexes:

i,j	The index of jobs {0, 1, ..., n}
f	Index related to factories {1, 2, ..., q}
k	The index related to the first step machines{1, 2, ..., m_1}
s	The index related to the second step machines{1, 2, ..., m_2}
l	The index related to the third step machines{1, 2, ..., m_3}

Decision variables:

$X_{i,w,f}$	If job i is processed in position w at the first step of factory f, equal 1, otherwise 0
$Z_{i,w,s,f}$	If job i is processed in position w of machine s in the first step of assembly in factory f, equal 1, otherwise 0.
$Y_{i,w,l,f}$	If job i is processed in position w on machine l in the second step of assembly in factory f, equal 1, otherwise 0.
$C_{w.k,f}$	Time to complete job at position w on machine k in the production step of factory f.
$CA_{w,s,f}$	Time to complete job at position w on machine s in the first step of assembly at factory f.
$CT_{w,l,f}$	Time to complete job at position w on machine l in the second step of assembly at factory f.

The mathematical model includes the constraints and the objective function as follows:

$$\text{Minimise } \sum_{i=j}^{n} T_j \tag{3}$$

$$\sum_{f=1}^{F}\sum_{w=1}^{n} X_{i,w,f} = 1 \qquad \forall\, i = 1,2,\dots,n \tag{4}$$

$$\sum_{i=1}^{n} X_{i,w,f} \le 1 \qquad \forall\, f = 1,\dots,F;\; w = 1,\dots,n \tag{5}$$

$$\sum_{i=1}^{n} X_{i,w-1,f} \ge \sum_{j=1}^{n} X_{j,w,f} \qquad \forall\, f = 1,\dots,F;\; w = 2,\dots,n \tag{6}$$

$$\sum_{f=1}^{F}\sum_{w=1}^{n}\sum_{s=1}^{m_2} Z_{i,w,s,f} = 1 \qquad \forall\, i = 1,2,\dots,n \tag{7}$$

$$\sum_{i=1}^{n} Z_{i,w,s,f} \le 1 \qquad \forall\, f = 1,\dots,F;\; w = 1,\dots,n;\; s = 1,2,\dots,m_2 \tag{8}$$

$$\sum_{i=1}^{n} Z_{i,w-1,s,f} \ge \sum_{j=1}^{n} Z_{j,w,s,f} \qquad \forall\, w = 2,\dots,n;\; s = 1,2,\dots,m_2;\; f = 1,\dots,F \tag{9}$$

$$\sum_{r=1}^{n}\sum_{s=1}^{m_2} Z_{i,r,s,f} = \sum_{w=1}^{n} X_{i,w,f} \qquad \forall\, i = 1,2,\dots,n; f = 1,\dots,F \tag{10}$$

$$\sum_{f=1}^{F}\sum_{w=1}^{n}\sum_{l=1}^{m_3} Y_{i,w,l,f} = 1 \qquad \forall\, i = 1,2,\dots,n \tag{11}$$

$$\sum_{i=1}^{n} Y_{i,w,l,f} \le 1 \qquad \forall\, f = 1,\dots,F;\; w = 1,\dots,n;\; l = 1,2,\dots,m_3 \tag{12}$$

$$\sum_{i=1}^{n} Y_{i,w-1,l,f} \geq \sum_{j=1}^{n} Y_{j,w,l,f} \qquad \forall\, w = 2,\dots,n;\ l = 1,2,\dots,m_3;\ f = 1,\dots,F \tag{13}$$

$$\sum_{r=1}^{n}\sum_{l=1}^{m_3} Y_{i,r,l,f} = \sum_{w=1}^{n} X_{i,w,f} \qquad \forall\, i = 1,2,\dots,n; f = 1,\dots,F \tag{14}$$

$$pt_{j,e} \geq \sum_{f=1}^{F}\sum_{r=1}^{n} Y_{j,r,e,f} \qquad \forall\, j = 1,2,\dots,n;\ e = 1,2,\dots,m_3 \tag{15}$$

$$M * \sum_{f=1}^{F}\sum_{r=1}^{n} Y_{j,r,e,f} \geq pt_{j,e} \qquad \forall\, j = 1,2,\dots,n;\ e = 1,2,\dots,m_3 \tag{16}$$

$$C_{w,k,f} \geq C_{w-1,k,f} + p_{i,k} * X_{i,w,f} - M * \left(1 - X_{i,w,f}\right) \qquad \forall\, i = 1,2,\dots,n; w = 2,\dots,n,;\ k = 1,2,\dots,m_1; f = 1,\dots,F \tag{17}$$

$$C_{1,k,f} \geq \sum_{i=1}^{n} p_{i,k} * X_{i,1,f} \qquad \forall\, k = 1,2,\dots,m_1; f = 1,\dots,F \tag{18}$$

$$CA_{w,s,f} \geq CA_{w-1,s,f} + \sum_{i=1}^{n} t_i * Z_{i,w,s,f} - M * \left(1 - \sum_{i=1}^{n} Z_{i,w,s,f}\right) \qquad \forall\, s = 1,2,\dots,m_2; w = 2,\dots,n; f = 1,\dots,F \tag{19}$$

$$CA_{w,s,f} \geq C_{r,K,f} + t_i * Z_{i,w,s,f} - M * \left(2 - Z_{i,w,s,f} - X_{i,r,f}\right) \qquad \forall\, k = 1,2,\dots,m_1; i = 1,2,\dots,n; w,r = 1,2,\dots,n;\ s = 1,2,\dots,m_2;\ f = 1,\dots,F \tag{20}$$

$$CT_{w,l,f} \geq CT_{w-1,l,f} + \sum_{i=1}^{n} pt_{i,l} * Y_{i,w,l,f} - M * \left(1 - \sum_{i=1}^{n} Y_{i,w,l,f}\right) \qquad \forall\, l = 1,2,\dots,m_3;\ w = 2,\dots,n; f = 1,\dots,F \tag{21}$$

$$CT_{w,l,f} \geq CA_{r,s,f} + pt_{i,l} * Y_{i,w,l,f} - M * (2 - Y_{i,w,l,f} - Z_{i,r,s,f}) \qquad \forall\, i = 1,2,\dots,n; w,r = 1,\dots,n;\ k = 1,2,\dots,m_1; f = 1,2,\dots,F;\ s = 1,2,\dots,m_2;\ l = 1,2,\dots,m_3 \tag{22}$$

$$T_j \geq \max\{0, (CT_{w,l,f} - d_j)\} - M * \left(1 - Y_{j,w,l,f}\right) \qquad \forall\, j = 1,2,\dots,n;\ l = 1,2,\dots,m_3;\ w = 2,\dots,n; f = 1,\dots,F \tag{23}$$

$$X_{i,w,f} \in \{0,1\},\ Y_{i,w,l,f} \in \{0,1\} \qquad \forall\, i = 1.\dots.n;\ w = 1,.\dots.n; l = 1,\dots,m_3;\ f = 1,\dots,F \tag{24}$$

$$C_{w,k,f}, CA_{w,f},\ CT_{w,l,f} \geq 0 \qquad \forall\, w = 1,.\dots.n; k = 1,2,\dots,m_1;\ f = 1,\dots,F \tag{25}$$

Equation 3 defines the value of the objective function, which minimizes the total tardiness of jobs by summing the tardiness times of all jobs. Constraint 4 demonstrates that every job in the first step must be assigned to a position. Constraint 5 specifies that maximum one job is assigned to each position in the first step of each factory. Constraint 6 determines that a position is occupied in the first step if its previous position is filled. Constraint 7 shows that each job in the second step is assigned to only one position of a machine. Constraint 8 determines that maximum one job is assigned to each position in each factory in the second step. Constraint 9 specifies that a position in the second step is occupied if its previous position is filled. Constraint 10 shows that if a job is assigned to the first step of a factory, it must also be assigned to the second step. Constraint 1 determines that each job in the third step is assigned to only one position of a machine. Constraint 12 shows that maximum one job is assigned to each position in the third step of each factory. Constraint 13 specifies that a position is occupied in the third step if its previous position is filled. Constraint 14 determines that if a job is assigned to the first step of a factory, it must also be assigned to the third step. Constraints 15 and 16 determine that each job in the third step is only assigned to its dedicated machine. Constraints 17 and 18 show the completion times of the jobs in the first step. Constraints 19 and 20 determine the completion times of the jobs in the second step. Constraints 21 and 22 show the completion times of the jobs in the third step. Constraint 23 determines the tardiness of each job. Constraint 24 shows the binary variables. Constraint 25 determines the range of continuous variables.

B. The sequence based model

In this case, the problem model is presented based on the sequence of jobs processing. The problem model includes the objective function and constraints. The parameters of the problem were defined in the previous section. The indexes and decision variables are presented initially, then the mathematical model.

Indexes:

i,j	The index of jobs {0, 1, ..., n}
f	Index related to factories {1, 2, ..., q}
k	The index related to the first step machines{1, 2, ..., m_1}
l	The index related to the second step machines{1, 2, ..., m_2}
e	The index related to the third step machines{1, 2, ..., m_3}

Decision variables

$X_{i,j,f}$	If job j is processed after job i in the first step of factory f, equal 1, otherwise 0.
$Z_{i,j,l,f}$	If job j is processed after job i on machine l in the second step of factory f, equal 1, otherwise 0.
$Y_{i,j,e,f}$	If job j is processed after job i on machine e in the third step of factory f, equal 1, otherwise 0.
$Cf_{i.k}$	Completion time of job i on machine k in the first step
Cs_i	The completion time of job i in the second step
CT_i	The completion time of job i in the third step

The mathematical model includes the objective function and constraints as follows:

Minimize $\sum_{i=j}^{n} T_j$ (26)

$$\sum_{f=1}^{F} \sum_{i=0.,j\neq i}^{n} X_{i,j,f} = 1 \qquad \forall\, j = 1,2,\dots,n \qquad (27)$$

$$\sum_{f=1}^{F} \sum_{j=1\,.j\neq i}^{n} X_{i,j,f} \leq 1 \qquad \forall\, i = 1,\dots,n \qquad (28)$$

$$\sum_{j=1}^{n} X_{0,j,f} = 1 \qquad \forall\, f = 1,\dots,F \qquad (29)$$

$$\sum_{r=1,r\neq j}^{n} \sum_{q=1,q\neq f}^{F} X_{j,r,q} \leq 1 - \sum_{i=0,i\neq j}^{n} X_{i,j,f} \qquad \forall j = 1,2,\dots,n\,;\ f = 1,\dots,F \qquad (30)$$

$$X_{i,j,f} + X_{j,i,f} \leq 1 \qquad \forall\, i = 1,\dots,n\,; j\ = 1,2,\dots,n, j \neq i\,; f = 1,\dots,F \qquad (31)$$

$$\sum_{f=1}^{F} \sum_{i=0\,.\ i\neq j}^{n} \sum_{l=1}^{m_2} Z_{i,j,l,f} = 1 \qquad \forall\, j = 1,2,\dots,n \qquad (32)$$

$$\sum_{f=1}^{F} \sum_{l=1}^{m_2} \sum_{j=1\,.j\neq i}^{n} Z_{i,j,l,f} \leq 1 \qquad \forall\, i = 1,\dots,n \qquad (33)$$

$$\sum_{j=1}^{n} Z_{0,j,l,f} \leq 1 \qquad \forall\, l = 1,2,\dots,m_2\,; f = 1,\dots,F \qquad (34)$$

$$\sum_{r=0,r\neq j}^{n} \sum_{l=1}^{m_2} Z_{r,j,l,f} \leq 1 - \sum_{q=1,q\neq f}^{F} \sum_{i=1,i\neq j}^{n} \sum_{s=1}^{m_2} Z_{j,i,s,q} \qquad \forall\, j\ = 1,2,\dots,n;\ f = 1,\dots,F \qquad (35)$$

$$\sum_{r=0,r\neq j}^{n} \sum_{f=1}^{F} Z_{r,j,l,f} \leq 1 - \sum_{q=1}^{F} \sum_{i=1,i\neq j}^{n} \sum_{s=1,s\neq l}^{m_2} Z_{j,i,s,q} \qquad \forall\, j\ = 1,2,\dots,n;\ l = 1,2,\dots,m_2 \qquad (36)$$

$$\sum_{l=1}^{m_2} (Z_{i,j,l,f} + Z_{j,i,l,f}) \leq 1 \qquad \forall\, i = 0,\dots,n\,; j\ = 0,2,\dots,n, j \neq i; f = 1,\dots,F \qquad (37)$$

$$\sum_{i=0,i\neq j}^{n} X_{i,j,f} = \sum_{r=0,r\neq j}^{n} \sum_{l=1}^{m_2} Z_{r,j,l,f} \qquad \forall j\ = 1,2,\dots,n; f = 1,\dots,F \qquad (38)$$

$$\sum_{f=1}^{F} \sum_{i=0\,.\ i\neq j}^{n} \sum_{e=1}^{m_3} Y_{i,j,e,f} = 1 \qquad \forall\, j = 1,2,\dots,n \qquad (39)$$

$$\sum_{f=1}^{F} \sum_{e=1}^{m_3} \sum_{j=1\,.j\neq i}^{n} Y_{i,j,e,f} \leq 1 \qquad \forall\, i = 1,\dots,n \qquad (40)$$

$$\sum_{j=1}^{n} Y_{0,j,e,f} \leq 1 \qquad \forall\, e = 1,2,\dots,m_3\,;\, f = 1,\dots,F \tag{41}$$

$$pt_{j,e} \geq \sum_{f=1}^{F} \sum_{i=0\,.\ i\neq j}^{n} Y_{i,j,e,f} \qquad \forall\, j = 1,2,\dots,n;\ e = 1,2,\dots,m_3 \tag{42}$$

$$M * \sum_{f=1}^{F} \sum_{i=0\,.\ i\neq j}^{n} Y_{i,j,e,f} \geq pt_{j,e} \qquad \forall\, j = 1,2,\dots,n;\ e = 1,2,\dots,m_3 \tag{43}$$

$$\sum_{r=0,r\neq j}^{n} \sum_{l=1}^{m_3} Y_{r,j,e,f} \leq 1 - \sum_{q=1,q\neq f}^{F} \sum_{i=1,i\neq j}^{n} \sum_{s=1}^{m_3} Y_{j,i,s,q} \qquad \forall\, j = 1,2,\dots,n;\ f = 1,\dots,F \tag{44}$$

$$\sum_{r=0,r\neq j}^{n} \sum_{f=1}^{F} Y_{r,j,e,f} \leq 1 - \sum_{q=1}^{F} \sum_{i=1,i\neq j}^{n} \sum_{s=1,s\neq l}^{m_3} Y_{j,i,s,q} \qquad \forall\, j = 1,2,\dots,n;\ e = 1,2,\dots,m_3 \tag{45}$$

$$\sum_{e=1}^{m_3} (Y_{i,j,le,f} + Y_{j,i,e,f}) \leq 1 \qquad \forall\, i = 0,\dots,n\,;\, j = 0,2,\dots,n, j \neq i; f = 1,\dots,F \tag{46}$$

$$\sum_{i=0,i\neq j}^{n} X_{i,j,f} = \sum_{r=0,r\neq j}^{n} \sum_{e=1}^{m_3} Y_{r,j,e,f} \qquad \forall j = 1,2,\dots,n; f = 1,\dots,F \tag{47}$$

$$Cf_{j,k} - Cf_{i,k} \geq pf_{j,k} - M * \left(1 - X_{i,j,f}\right) \qquad \forall\, i = 0,1,\dots,n\,;\, j = 1,2,\dots,n, j \neq i;\ k = 1,2,\dots,m_1; f = 1,\dots,F \tag{48}$$

$$C_{0,k} \geq pf_{0,k} \qquad \forall\, k = 1,2,\dots,m_1 \tag{49}$$

$$Cs_j - Cs_i \geq ps_j - M * (1 - \sum_{e=1}^{m_2} Z_{i,j,e,f}) \qquad \forall\, i = 0,1,\dots,n\,;\, j = 1,2,\dots,n, j \neq i; f = 1,\dots,F \tag{50}$$

$$CA_j - Cf_{j,k} \geq ps_j \qquad \forall\, j = 0,1,\dots,n;\ k = 1,2,\dots,\ m_1 \tag{51}$$

$$Ct_j - Ct_i \geq \sum_{e=1}^{m_3} pt_{j,e} - M * (1 - \sum_{e=1}^{m_3} Y_{i,j,e,f}) \qquad \forall\, i = 0,1,\dots,n\,;\, j = 1,2,\dots,n, j \neq i; f = 1,\dots,F;\ e = 1,2,\dots,m_3 \tag{52}$$

$$Ct_j - Cs_j \geq \sum_{e=1}^{m_3} pt_{j,e} \qquad \forall\, j = 0,1,\dots,n \tag{53}$$

$$T_j = \max\{0, Ct_j - d_j\} \qquad \forall\, j = 1,2,\dots,n \tag{54}$$

$$X_{i,j,f} \in \{0,1\}\,,\, Z_{i,j,e,f} \in \{0,1\}, Y_{i,j,e,f} \in \{0,1\} \qquad \forall\, i = 0,1,\dots.n\,;\, j = 1,\dots,n\,;\ e = 1,2,\dots,m_3\,;\ f = 1,2,\dots,F \tag{55}$$

$$Cf_{i,k}, Cs_j\,,\ Ct_j \geq 0 \qquad \forall\, j = 1,\dots,n\,;\ k = 1,2,\dots,m\,;\ f = 1,2,\dots,F \tag{56}$$

Constraint 26 represents the objective function, which is the sum of the tardiness times for all jobs. Constraint 27 specifies that each job must have a prerequisite in the first step. Constraint 28 indicates that each job has maximum one post-requirement in the first step. Constraint 279 illustrates that zero job must have one post-requirement in every factory. Constraint 30 specifies that each job is assigned to only one factory. Constraint 31 determines the precedence and delay of two jobs in the first step; therefore, two jobs cannot be prerequisites and post-requirements for each other in the first step. Constraint 32 shows that every job must have a prerequisite in the second step. Constraint 33 determines that each job has at most one post-requirement in the second step. Constraint 34 displays that zero job on each machine in each factory has at most one post-requirement in the second step. Constraint 35 specifies that each job is assigned to only one factory in the second step. Constraint 36 shows that each job is assigned to only one machine in the second step. Constraint 37 determines the precedence and delay of two jobs in the second step, therefore two jobs cannot be prerequisites and post-requirement of each other. Constraint 38 determines that if a job is assigned to the first step of a factory, it must also be assigned to the second step of the same factory. Constraint 39 specifies that each job must have a prerequisite in the third step. Constraint 40 shows that each job has at most one post-requirement in the third step. Constraint 41 determines that zero job on each machine has at most one post-requirement in the third step of each factory. Constraints 42 and 43 determine that each job is assigned only to its dedicated machine in the third step. Constraint 44 determines that each job is assigned to only one factory in the third step. Constraints 45 specifies that each job is assigned to only one machine in the third step. Constraint 46 presents the precedence and delay of two jobs and shows that two jobs cannot be prerequisites and post-requirement of each other in the third step. Constraint 47 determines that if a job is assigned to the first step of a factory, it must be assigned to the third step of the same factory. Constraints 48 and 49 show the completion times of the jobs in the first step. Constraints 50 and 51 determine the time to complete the jobs in the second step. Constraints 52 and 53 indicate the completion times of the jobs in the third step. Constraint 54 defines the tardiness of each job. Constraint 55 determines the binary variables. Constraint 56 demonstrates the range of continuous variables.

C. Performance comparison of sequence and position based models

To compare the two models and select the most appropriate one, each model was coded in GAMS software. Given a time limitation of 3600 seconds, the calculation results for 40 instances are shown in Table III. For each models, the value of the objective function (obj), the solution time (time) and the deviation from the best solution (GAP) for two models have been calculated in each instance. The value of GAP is obtained through the use of equation (57).

$$GAP_i = \frac{S_i - bst}{bst} \qquad \forall\, i = 1,2 \qquad (57)$$

In equation (57), bst represents the best solution obtained from the two models. S_i is the solution obtained from model i and GAP_i is the relative deviation from the best solution for the model i. According to Table III, the sequence-based model appears to be more effective in finding optimal solutions compared to the position-based model. It achieved the optimal solution in 29 out of 40 instances, while the position-based model reached the optimal solution in only 16 instances. This suggests that the sequence-based model is more robust and can handle a wider range of problem instances. As can be seen, the position-based model has memory errors in 6 instances (indicated by OM in the table), while the sequence-based model has memory errors in two instances. This suggests that the position-based model is more prone to memory errors than the sequence-based model. The average of calculated values is shown in the end of the table. The average of solution time and GAP_i for the sequence-based model equal 1113 and 0, respectively, and for the position-based model equal 2167 and 0.3260. For both parameters, the sequence-based model values are better than the position-based model values. Overall, the results indicate that the sequence-based model demonstrates better efficiency and is more suitable for solving the problem in small dimensions.

Table III. results comparison of sequence and position based models

Instance	n	f	m_1	m_2	m_3	Position based			Sequence based		
						obj	time	GAP	obj	time	GAP
1	3	2	2	2	2	0	2	0	0	2	0
2	3	2	4	2	2	0	2	0	0	2	0
3	3	2	6	2	2	0	2	0	0	1	0
4	4	2	2	2	2	23	3	0	23	2	0
5	4	2	6	2	2	31	4	0	31	2	0
6	4	3	4	2	2	0	3	0	0	2	0
7	5	2	2	2	2	33	16	0	33	2	0
8	5	2	6	2	2	117	86	0	117	2.3	0
9	5	2	4	3	3	104	85	0	104	2	0
10	5	3	2	2	2	0	8	0	0	2	0
11	5	3	4	2	2	9	129	0	9	2	0
12	6	2	2	2	2	86	1184	0	86	3	0
13	6	2	4	3	3	155	2951	0	155	3	0
14	6	3	2	2	2	16	1653	0	16	5	0
15	6	3	4	2	2	25	2754	0	25	2	0
16	7	2	2	2	2	92	3600	0.001	92	4	0
17	7	2	6	3	3	339	3600	0.0059	337	37	0
18	7	3	4	2	3	47	3600	0.3206	35	3	0
19	7	4	2	3	2	0	232	0	0	2	0
20	7	4	4	2	3	12	3600	0	12	4	0
21	8	2	4	2	2	388	3600	0.0517	369	603	0
22	8	2	2	2	3	230	3600	0.0701	215	69	0
23	8	2	6	3	3	585 (OM)	1308	0.3496	433	416	0
24	8	3	2	2	3	93	3600	0.086	85	89	0
25	8	4	4	2	2	83	3600	0.7594	47	11	0
26	10	2	4	3	2	656 (OM)	2410	0.3014	504	3600	0
27	10	3	2	2	3	193	3600	0.3822	139	3600	0
28	10	3	4	2	3	320 (OM)	1995	0.4237	225	3600	0
29	10	4	2	3	2	88	3600	1.1884	40	410	0
30	10	4	6	3	3	489	3600	1.2211	220	3600	0
31	12	2	4	3	2	1325	3600	0.3482	983	3600	0
32	12	3	2	2	3	406	3600	0.1275	360	2218	0
33	12	3	4	2	3	871	3600	0.4915	584	1260	0
34	12	4	2	3	2	516 (OM)	2723	1.4674	209	3600	0
35	12	4	6	3	3	584	3600	0.3297	439	3600	0
36	15	2	4	3	2	3160	3600	0.6079	1965	3543	0
37	15	3	2	2	3	1880 (OM)	1877	1.1717	866 (OM)	1335	0
38	15	3	4	2	3	1596 (OM)	2490	0.3539	1179 (OM)	2105	0
39	15	4	2	3	2	1234	3600	1.1247	581	3600	0
40	15	4	6	3	3	2531	3600	1.8552	886	3600	0
Average						458	2167.93	0.326	285	1113.58	0

IV. SOLUTION METHOD

It is certainly promising that the proposed solution method is a highlight of the research. The pursuit of improved optimization methods is essential for solving complex real-world problems and achieving greater efficiency across various fields. This paper presents the BBO algorithm to address the problem at hand. The BBO algorithm is inspired by nature, specifically the migration patterns of animals and birds between islands. Introduced by Simon (2008), the algorithm draws from biogeography, which examines the behavior of various biological species over time and across different geographical locations. In this context, habitats are considered along with their resident species.. The Habitat Suitability Index (HSI) indicates how suitable a geographical area is for the residence of a species. Suitability Index Variables (SIV) are variables that determine the suitability of a habitat (Such as rainfall, plant diversity, diversity of topographic features, environment and temperature). Resident species in habitats (solutions) where have high HSI, because of crowded habitats migrate to another habitat. In biogeography, this migration is denoted as emigration and μ is the rate of this migration. Therefore, the rate μ is higher in habitats with a high HSI. Habitats with low HSI serve as suitable destinations for species from other habitats due to lower population density. This type of migration is referred to as immigration, with the migration rate denoted by λ. As species migrate to these lower-density habitats, the HSI increases, as the desirability of a habitat is proportional to its density. Figure 2 illustrates the distribution of species within a habitat, along with the emigration rate μ and immigration rate λ, both of which relate to the number of species present in the habitat. The migration curve indicates that the maximum immigration rate for a habitat occurs when there are no species present. As the number of species increases, the habitat becomes more crowded, resulting in a decreased immigration rate. S_{max} is the maximum possible species number that the habitat covers and the immigration rate becomes zero.

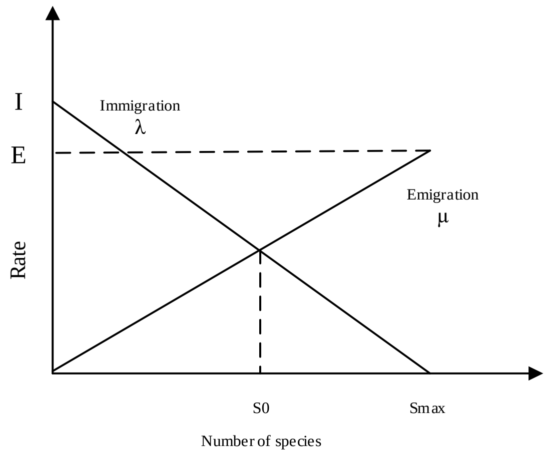

Fig. 2. The distribution of species in a habitat

The emigration rate will be zero if the number of species is zero in a habitat. As the number of species increases, the habitat becomes more crowded, so to discover other possible habitats more species can leave the habitat, and the rate of emigration increases. E is the maximum rate of emigration. E happen when the habitat has the most species number. S_0 shows the species number in the point equilibrium that the rate of immigration and emigration are equal. The values of emigration and immigration rate are calculated based on equations 58 and 59.

$$\lambda_k = I\left(1 - \frac{k}{n}\right) \tag{58}$$

$$\mu_k = \frac{Ek}{n} \tag{59}$$

In these equations, k indicates the rank of habitat k based on its suitability value, where 1 represents the worst rank and n the best. Each habitat has a population s at time t, and P_s denotes the probability of s species existing in the habitat. Over a time interval $\varDelta$t, the number of species can either increase, decrease, or remain constant. Thus, the number of species at time $(\varDelta t + 1)$ will be equal s if one of the following three situations occurs:

1. The number of species would equal s if immigration and emigration don't take place.
2. An immigration takes place, in which the number of species at time t equal $(s - 1)$.
3. An emigration takes place, in which the number of species at time t equal $(s + 1)$.

The probability of the number of species at time $(\varDelta t + 1)$ that equal s is calculated according to equation (60):

$$P_s(\varDelta t + 1) = P_s(t)(1 - \lambda_s \varDelta t - \mu_s \varDelta t) + P_{s-1}\lambda_{s-1}\varDelta t + P_{s+1}\lambda_{s+1}\varDelta t \tag{60}$$

According to equation (60) in steady state, the probability of the species number that equal s is calculated by equation (61):

$$p_i = \frac{v_i}{\sum_{i=1}^{n} v_i} \tag{61}$$

In equation (61), the value of v_i is calculated based on equation (62):

$$v_i = \begin{cases} \dfrac{n!}{(n-1-i)!\,(i-1)!} & (i = 1, \dots, i') \\ v_{n+2-i} & (i = i' + 1, \dots, n+1) \end{cases} \tag{62}$$

```
- Determination of the primary values of problem parameters includes: $m_{max}$ , $E$ , $I$
- The creation of primary population from habitats
- Calculation of HSI values (fitness) for each habitat
  - Beginning of algorithm repetition loop
    - Transmission the part of the best solutions to the next generation
     - Calculation the values of $\lambda_i$ ، $\mu_i$ ، $p_i$ and $m_i$ (mutation rate) for each habitat
        - Doing migration operations between habitats
         - Beginning of the loop to select the habitat $i$
           - Selection of habitat $i$ with probability $\lambda_i$
            - If habitat $i$ is selected
             - Selection of habitat $j$ with probability $\mu_i$ (Roulette wheel mechanism)
               - If habitat j is selected:
                - Doing the two-point cross over operator
               - End if of habitat $j$ selection
             - End loop of habitat $j$ selection
            - End if of habitat $i$ selection
           - End loop of habitat $i$ selection
        - Doing variation or mutation operations in each habitat
           - Selection habitat $i$ with probability $m_i$
            - If habitat $i$ is selected
             - Doing variation and evolution in each habitat through the inversion mutation
               operator
            - End if of habitat $i$ selection
           - End loop of habitat $i$ selection
  - The end loop of the algorithm repetition
  - Control the stop condition
```

Fig. 3. BBO algorithm Steps

The BBO algorithm has operators that change the number of habitat species. Similar to the crossover operator in Genetic Algorithm (GA), the migration operator in BBO aims to enhance the diversity and quality of solutions by combining information from different parts of the search space. In BBO, habitats represent potential solutions, and the movement of species between habitats represents the exchange of information between these solutions. The analogy between migration and crossover is clarified when the specific implementation of the migration operator is considered. In BBO, the migration process involves selecting a pair of habitats and exchanging a subset of species between them. This process effectively combines the characteristics of the two habitats, leading to the creation of new and potentially better solutions. Just as crossover helps GA explore new regions of the search space, migration in BBO enables the exploration and exploitation of different solution possibilities. Both operators promote diversity and search efficiency, contributing to the overall effectiveness of their respective optimization algorithms. Natural disasters such as flood, storm, disease, etc. change the number of habitat species. The steps of BBO algorithm are according to Figure 3.

To implement the algorithm, the required operators are calculated as follows.

A. Solution representation

In order to code the under investigation problem, it is necessary to select a suitable solution representation. According to through the use of algorithm, two types of solution representation have been examined in this section, for solving the problem. In the solution representation 1, the representation is in the single line. For example, for n machines and m factories, $(n + m - 1)$ random numbers are generated in the range of 0 to 1. The cells 1 to n show the jobs and the $(m - 1)$ last number are for separation of the factories. The numbers are arranged in ascending order. An example with 10 jobs and 3 factories is illustrated in Figure 4. Here, $(n + m - 1 = 12)$ random numbers are generated and sorted in ascending order. In this scenario, jobs 6 and 7 are assigned to the first factory, jobs 5 and 9 are assigned to the second factory, and jobs 8, 4, 10, 3, 2, and 1 are assigned to the third factory. All jobs are then processed in the same manner.

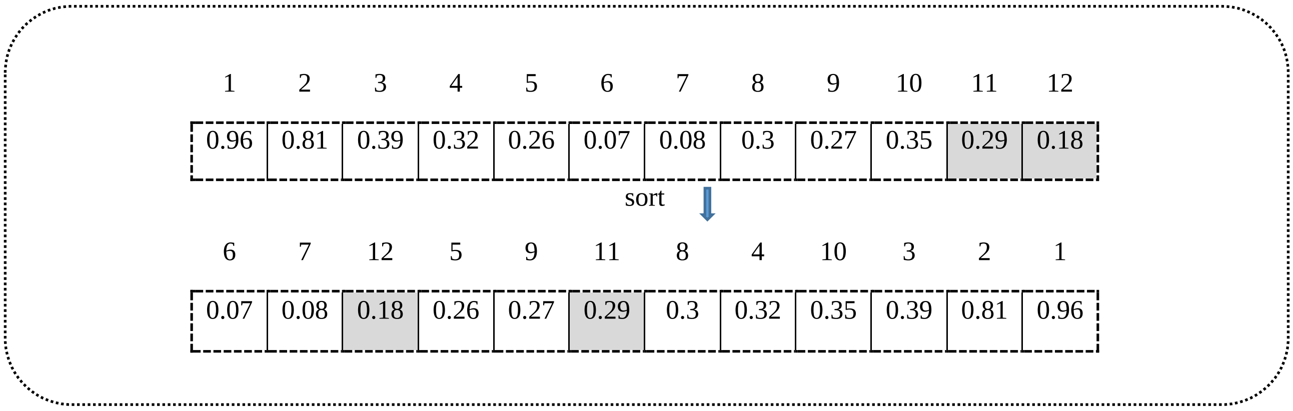

Fig. 4. Solution representation 1 (SR1)

In the solution representation 2, random numbers are generated in the range of $[1, F + 1)$. The integer section of the number indicates the factory number, and the decimal section determines the processing sequence of each factory. In Figure 5, if the numbers are arranged in ascending order, the lowest value is 1.25, which is related to job 4. Considering it is in the range of $[1,2)$, is assigned to the first factory and is placed in the first position. The next number is 1.52, that is related to job 1. Therefore, it is placed in the second position of the first factory. In the same way, the jobs are assigned to the factories and the sequence is determined.

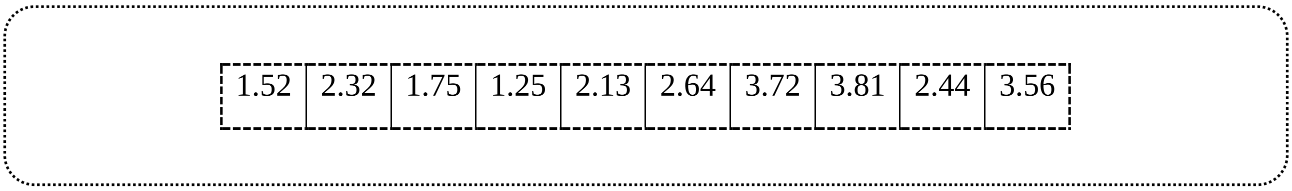

Fig. 5. Solution representation 2 (SR2)

The utilization results of the two solution representations in the BBO algorithm are presented in Table IV. For six samples, each sample was executed five times for each solution representation, and the values for minimum (Min), average (Ave), and maximum (Max) deviation from the best solution were calculated using Equation (55). Based on the results obtained within the same time limitation (1200 seconds) for both solution representations, the best solutions across all five samples correspond to Solution Representation 2 (SR2). The average deviation of all samples equal 0.077 for representation of solution 1 (SR1) and 0.0207 for representation of solution 2 (SR2), that the representation of solution SR2 is less and better. Also, Min and Max values are lower for SR2. Therefore, SR2 solution representation is selected.

Table IV. Comparison of the solution representations 1 and 2

Instance	n	f	m_1	m_2	m_3	**BBO-SR1**			**BBO-SR2**		
						Min	**Ave**	**Max**	**Min**	**Ave**	**Max**
1	15	3	2	2	2	0.0122	0.0224	0.0322	0.0000	0.0073	0.0133
2	25	4	2	2	3	0.0097	0.0292	0.0563	0.0000	0.0057	0.0130
3	35	4	4	3	3	0.0793	0.1079	0.1786	0.0000	0.0434	0.0787
4	45	6	6	2	2	0.0528	0.0915	0.1277	0.0000	0.0145	0.0297
5	55	6	4	3	2	0.0269	0.0955	0.1395	0.0000	0.0289	0.0662
6	65	8	2	3	3	0.0491	0.1157	0.1914	0.0000	0.0245	0.0704
Average						0.0383	0.0770	0.1209	0.0000	0.0207	0.0452

B. Decoding

Considering parallel machines in each job processing step is indeed a significant contribution and adds a whole new layer of complexity to the optimization problem. Considering that there are parallel machines in the second and third stages, after finishing the processing in the first stage, the jobs are processed on a machine in the second and third stages, which are released earlier. The presented sequence by solution representation is only related to the first step and the sequence of jobs processing on the machines is determined by decoding in the second and third steps. The second step in the decoding process often involves assigning jobs to the machine that becomes available first. This is a common approach called "earliest release time" (ERT) scheduling. In the third step, there are dedicated parallel machines, and each job is processed only on one machine in this step, if a processing queue is created, the jobs in the queue are processed in order of Modified Due-Date (MDD) (Baker & Trietsch, 2009). The MDD is according to rule 1.

Rule 1: Suppose at time t, the machine is available and a job must be selected for processing. For jobs in the queue, the modified due-date is defined as equation (63) and the sequence of jobs is arranged in ascending order of the new due-date.

$$d'_j(t) = \max\{d_j, t + p_j\} \tag{63}$$

For example, assume six jobs are assigned to a factory. As shown in Figure 6, the sequence of jobs processing is 6, 3, 5, 1, 2 and 4 in the first step. There are three machines in the second step. jobs 5, 6 and 3 are processed on each of the machines, firstly. Machine 2 processes job 1, that has the fastest completion time. Then machine 3 processes job 2. Machine 1 processes job 4 finally. Jobs 2, 3 and 6 can only be processed on machine 1 and jobs 1, 4 and 5 on machine 2 in the third step. Job 6 is processed on machine 1 firstly. Then jobs 2 and 3 are in the queue. Suppose $d'_2(t) \leq d'_3(t)$, job 2 is completed firstly, then job 3. On machine 2, job 5 is processed firstly, then jobs 1 and 4 are in the queue. Suppose $d'_1(t) \leq d'_4(t)$, job 1 is processed firstly, then job 4.

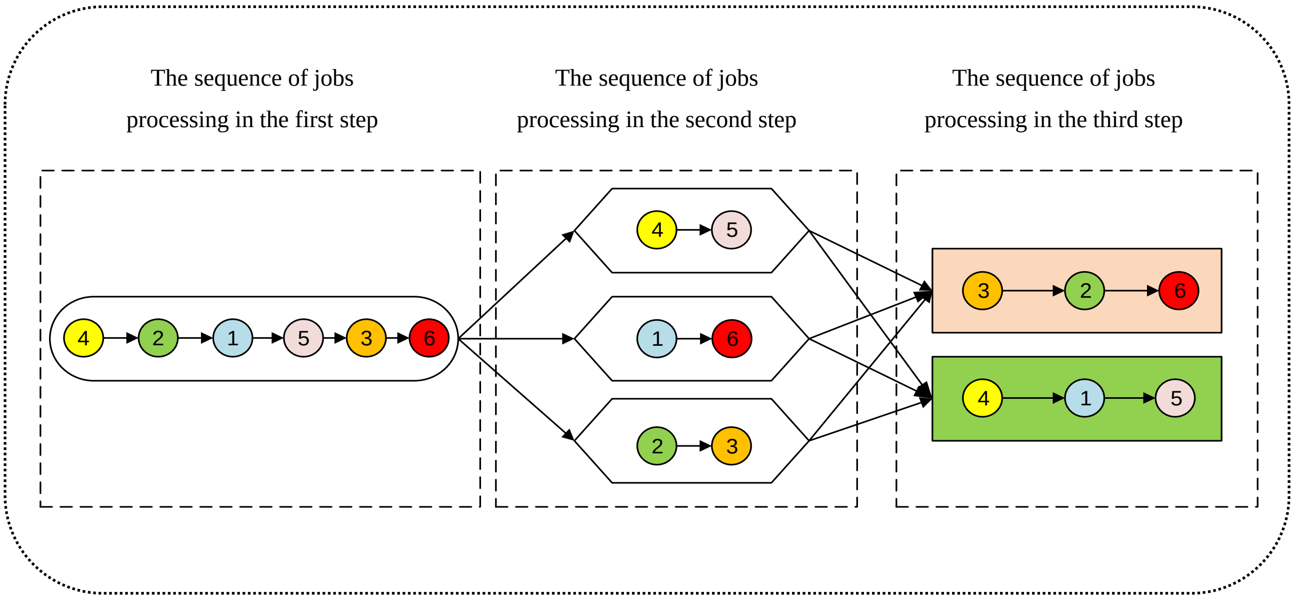
The sequence of jobs processing in the first step
The sequence of jobs processing in the second step
The sequence of jobs processing in the third step
4 2 1 5 3 6
4 5
1 6
2 3
3 2 6
4 1 5

Fig. 6. Sequence of jobs processing in the second and third step

C. Creation of the primary

Generating the initial population randomly is a common starting point for many optimization algorithms, in order to start the implementation of algorithm, the solutions primary population is randomly generated. By creating a diverse initial population through random generation, it ensures that the optimization algorithm explores a broader range of potential solutions. This increases the chances of finding high-quality solutions, even if the initial sequences themselves are not necessarily optimal. The population size is PS. This population is updated in each iteration of the algorithm and a new population of solutions is created.

D. Elite selection

In each iteration, some of the best solutions are transferred to the next generation unchange with the probability of P_{slct}. This selection preserves the high quality solutions and improves them in the combination with other solutions.

E. Migration operation

To create new solutions in each generation of the algorithm, migration occurs between habitats. Emigration takes place in high HSI and SIV habitats, and the entrance probability of a new species is lower than the exit of a species from the habitat. Therefore, the present species in the habitat tend to leave it. Each species in the habitat leaves it with a probability of μ_k. On the other hand, the habitat with low HSI tends to attract species from other habitats and the rate of immigration is higher than the habitat with high HSI. The immigration rate for each habitat is λ_k. In the classic state, the migration diagram of a species in the habitat is linear, while it can be non-linear and change non-linearly due to sudden changes in the habitat. Ma, (2010) explored various states of the migration diagram for a species and identified six different models for it. According on this, the best values of the emigration and the immigration rate are calculated according to relations 64 and 65.

$$\lambda_k = \frac{I}{2}\left(\cos\left(\frac{k\pi}{n}\right) + 1\right) \tag{64}$$

$$\mu_k = \frac{E}{2}\left(-\cos\left(\frac{k\pi}{n}\right) + 1\right) \tag{65}$$

In order to take place the migration, habitat i is selected as the first parent with probability λ_i. In order to select the second parent, habitat j is selected by utilization of the roulette wheel mechanism with probability μ_j. Here, the two-point cross over operator of the genetic algorithm is used to do the migration (soon et al., 2013).

In this method, two points are selected in parents accidentally. Cells between two points are moved to parents and new childs are created. Then the HSI value of each child is calculated and the child with better HSI is selected and passed on to the next generation. An example with 10 jobs is presented in Figure 7. Two points, 5 and 8, are randomly selected on the parent solutions. The numbers between two points are moved among parents.

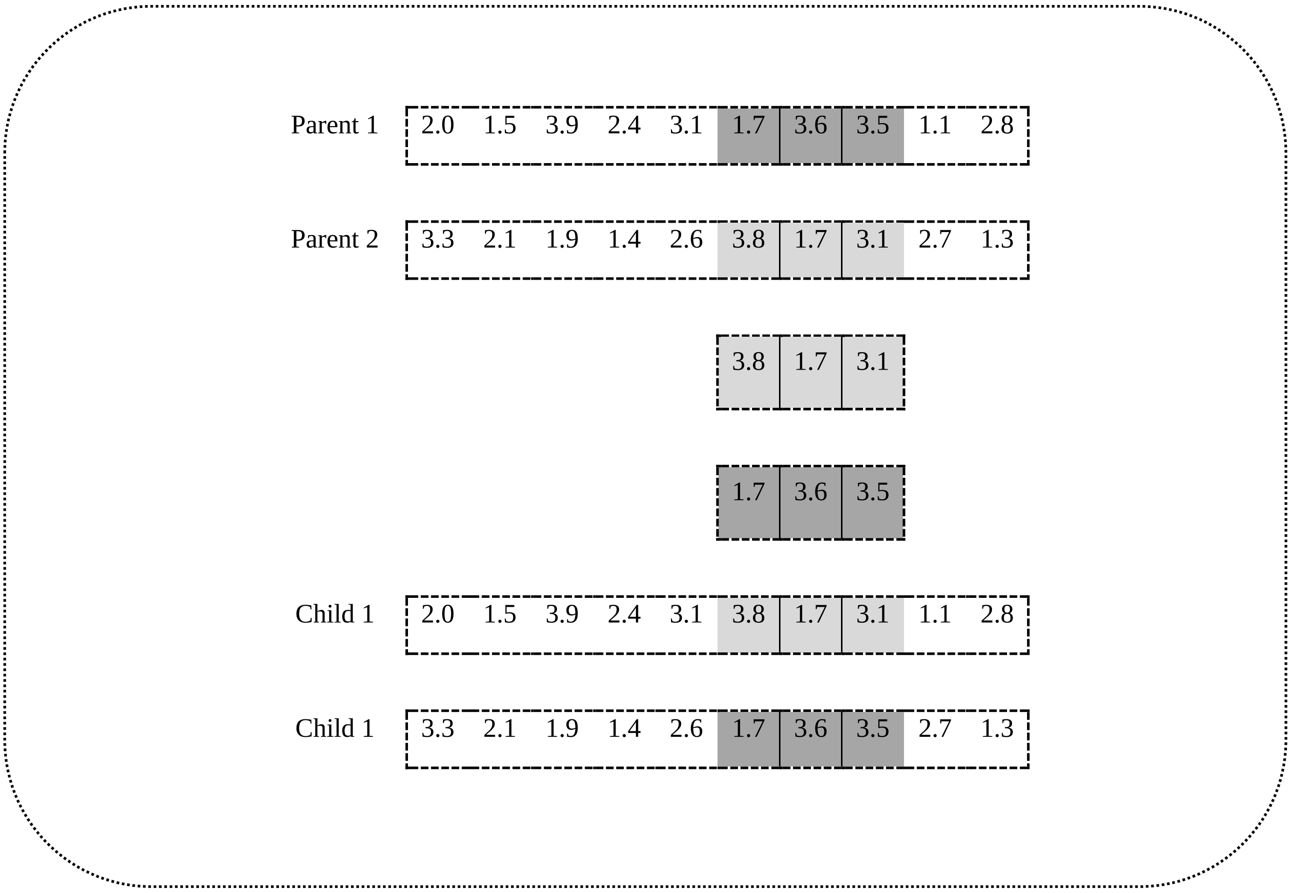

Fig. 7. An example of migration between two habitats

The migration operation is performed with the probability P_{mig} on each solution in every iteration of the algorithm. The pseudo-code of the migration operator between habitats is shown in Figure 8.

```
- Beginning of the loop to select the habitat $i$
  - Selection of habitat $i$ with $\lambda_i$ probability
    - If habitat $i$ is selected
      - The Beginning of the loop to select the habitat $j$
        - If habitat $j$ is selected
          - Selection of habitat $j$ with probability $\mu_i$ using the roulette wheel mechanism
          - Migration between habitats using the two-point cross over operator
        - End if of habitat $j$ selection
      - End loop of habitat $j$ selection
    - End if of habitat $i$ selection
- End loop of habitat $i$ selection
```

Fig. 8. Pseudo-code of migration between two habitats

F. Mutation operation

In the BBO algorithm, natural disasters such as flood, storm or disease changes number of the habitat species. These changes are like mutation in the GA. The obtained solutions of the algorithm execution may not have the necessary diversification, or in order to improve the solutions, mutation is performed on a part of the solutions. Here, inversion is used to perform mutation on the solutions (Lin et al., 2010). In this type of mutation, two points are selected randomly, then the sequence of numbers between these two points is reversed. An example is shown in Figure 9. Two points 2 and 6 are selected randomly, the sequence of numbers between these two points is reversed.

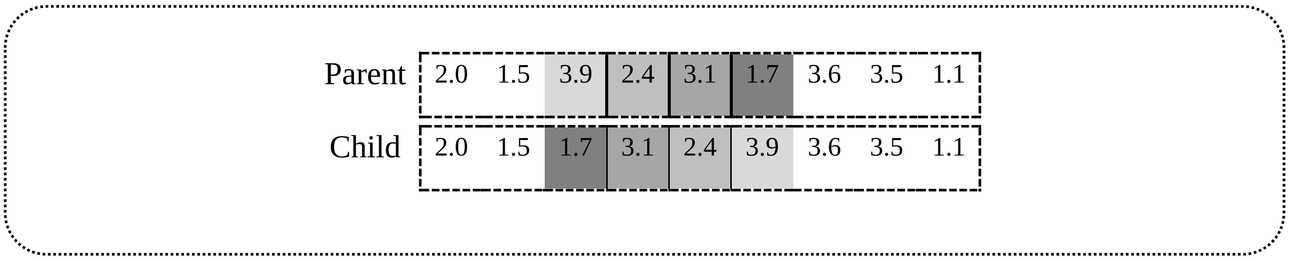

Fig. 9. An example of migration between two habitats

Mutation is applied to each habitat based on the mutation rate m_i. In order to calculate the mutation rate for each habitat, the value of v_i is calculated based on equation (66).

$$v_i = \begin{cases} \dfrac{n!}{(n-1-i)!\,(i-1)!} & (i = 1, \dots, i') \\ v_{n+2-i} & (i = i'+1, \dots, n+1) \end{cases} \tag{66}$$

According to the obtained value of v_i, the steady state value for the probability of species number is determined using equation (67).

$$p_i = \frac{v_i}{\sum_{i=1}^{n} v_i} \tag{67}$$

Based on the obtained value of m_i, the mutation rate is calculated using equation (68).

$$m_i = m_{max}\left(1 - \frac{p_i}{p_{max}}\right) \tag{68}$$

In equation (68), m_{max} is the maximum mutation rate, that is determined by the user. p_{max} is the maximum obtained value of p_i. Mutation is done with probability m_i in habitat i. The mutation operation is performed on each solution in each iteration of the algorithm with the probability P_{mut}. The pseudo-code of the mutation operation is according to Figure 10.

```
- The starting loop to select the habitat $i$
  - selection habitat $i$ with probability $m_i$
    - If habitat $i$ is selected
        - Performing reverse mutation operation on the habitat
    - End if of habitat $i$ selection
 - End loop of habitat $i$ selection
```

Fig. 10. Pseudo-code for the mutation operator

G. Solutions improvment

To enhance the performance of the proposed algorithm, dominance rules are introduced in this section. These rules not only accelerate the attainment of results but also represent a significant advancement in modifying the BBO algorithm from its classical form to an improved version. In the third step of the process, dedicated parallel machines are utilized, with each machine designated to process only one type of job. This arrangement effectively transforms each machine's problem into a single-machine problem, where the objective function is the minimization of the sum of tardiness times. Moreover, the completion time of each job in the second step serves as a release time. To improve solution results, rules relevant to the single-machine mode, which aims to minimize the sum of tardiness times and release times, can be applied to the jobs. Various rules have been examined for applicability to this problem, and the results will be evaluated to identify the most suitable rules (Yin et al., 2014).

Parameters and variables:

r_j	Release date of job j in the third step (Completion time of job j in the second step)
p_j	The processing time of job j in the third step
d_j	Due date of job j
t	Completion time of the last job in the sequence of third step machine

Two schedules, S and S', are considered, with the order of jobs i and j reversed between them. in schedule S, job i is processed before job j, and it is the opposite in S'.

Rule 2: if $p_i \le p_j$, $\max\{r_i, t\} \le \max\{r_j, t\}$ and $d_i \le \max\{\max\{r_j, t\} + p_j, d_j\}$ then dominates.
Rule 3: if $\max\{r_i, t\} \le \max\{r_j, t\} < \max\{r_i, t\} + p_i$, $\max\{r_i, t\} + d_i \le \max\{r_j, t\} + d_j$ and $p_i \le p_j$ then dominates.
Rule 4: if $\max\{r_i, t\} + p_i \ge d_i$, $\max\{r_i, t\} \le \max\{r_j, t\} < \max\{r_i, t\} + p_i$ and $p_i \le p_j$ then dominates .
Rule 5: if $\max\{max\{r_i, t\} - \max\{r_j, t\} + d_i - d_j, \max\{r_i, t\} - \max\{r_j, t\}\} \le 0$ and $\max\{r_j, t\} \le \max\{r_i, t\} < \max\{r_j, t\} + p_j$ then dominates.

To evaluate the utilization percentage of each rule, the algorithm was executed with $PS = 100$ and 10 repetitions. 1000 solutions were checked in total. The related results to the utilization times and percentage are specified according to Table V. Rule 5 has the highest percentage of rule usage compared to other rules with a total average of 24%. Conversely, Rule 3 exhibited the lowest utilization percentage, averaging 20.09%. It is noteworthy that the average utilization percentage for each of the four rules exceeded 20%, indicating their potential for improving the problem. Additionally, an increase in the number of jobs, while keeping other parameters constant, led to a rise in the utilization percentage of each rule. To assess the algorithm's performance before and after applying the rules, three different problem sizes were employed for the instances. For each instance, the algorithm has been executed 5 times before and after applying the rules. The deviation from the best obtained solution is calculated for each execution, as well as the minimum (Min), average (Ave) and maximum (Max) values of the deviations are calculated and all of them are shown in Table VI. For rules 2, 3 and 5, applying the rule has improved the results in all three instances. In rule 4, the first instance of using the rule did not improve the average results, and also the average deviation percentage for the other two instances of this rule is more than 2%, that is more than other rules.

Table V. The utilization percentage of dominance rules in the third step

Rule	size	The utilization times in 1000 soutions					Utilization percentage		
		Run 1	Run 2	Run 3	Run 4	Run 5	Min	Ave	Max
Rule 2	20-3-2-3-2	143	133	169	118	109	10.90%	13.44%	16.90%
	30-3-2-3-2	267	211	258	237	212	21.10%	23.70%	26.70%
	40-3-2-3-2	290	249	223	272	319	22.30%	27.06%	31.90%
	Average						18.10%	21.40%	25.17%
Rule 3	20-3-2-3-2	153	94	59	90	170	5.90%	11.32%	17.00%
	30-3-2-3-2	145	226	175	198	180	14.50%	18.48%	22.60%
	40-3-2-3-2	337	258	305	330	293	25.80%	30.46%	33.70%
	Average						15.40%	20.09%	24.43%
Rule 4	20-3-2-3-2	101	167	150	122	150	10.10%	13.80%	16.70%
	30-3-2-3-2	255	228	207	273	280	20.70%	24.86%	28.00%
	40-3-2-3-2	319	271	296	326	269	26.90%	29.62%	32.60%
	Average						19.23%	22.76%	25.77%
Rule 5	20-3-2-3-2	151	135	124	192	117	11.70%	14.38%	19.20%
	30-3-2-3-2	103	285	253	277	284	10.30%	24.04%	28.50%
	40-3-2-3-2	375	347	334	321	302	30.20%	33.58%	37.50%
	Average						17.40%	24.00%	28.40%

Table VI. Performance evaluation of dominance rules in the third step

Rule	Size	Without/With rule	Deviation percentage from the best solution					Min	Ave	Max
			Run 1	Run 2	Run 3	Run 4	Run 5			
Rule 2	20-3-2-3-2	Without	0.07%	0.28%	1.98%	0.92%	0.57%	0.07%	0.76%	1.98%
		With	0.00%	0.50%	1.27%	1.13%	0.71%	0.00%	0.72%	1.27%
	30-3-2-3-2	Without	3.02%	1.62%	4.36%	1.89%	3.74%	1.62%	2.93%	4.36%
		With	3.54%	0.67%	1.85%	1.51%	0.00%	0.00%	1.51%	3.54%
	40-3-2-3-2	Without	1.60%	4.31%	5.26%	5.72%	2.74%	1.60%	3.93%	5.72%
		With	1.85%	0.00%	5.04%	2.67%	3.43%	0.00%	2.60%	5.04%
Rule 3	20-3-2-3-2	Without	1.00%	1.21%	2.93%	1.86%	1.50%	1.00%	1.70%	2.93%
		With	2.50%	0.00%	4.14%	0.43%	1.14%	0.00%	1.64%	4.14%
	30-3-2-3-2	Without	2.75%	1.35%	4.08%	1.62%	3.46%	1.35%	2.65%	4.08%
		With	1.84%	0.00%	0.31%	0.82%	1.35%	0.00%	0.86%	1.84%
	40-3-2-3-2	Without	1.10%	3.80%	4.74%	5.20%	2.23%	1.10%	3.41%	5.20%
		With	0.00%	1.33%	1.07%	0.56%	0.31%	0.00%	0.65%	1.33%
Rule 4	20-3-2-3-2	Without	0.21%	0.42%	2.12%	1.06%	0.71%	0.21%	0.91%	2.12%
		With	0.00%	0.35%	2.55%	0.78%	1.13%	0.00%	0.96%	2.55%
	30-3-2-3-2	Without	2.75%	1.35%	4.08%	1.62%	3.46%	1.35%	2.65%	4.08%
		With	3.24%	0.00%	1.29%	3.86%	2.53%	0.00%	2.18%	3.86%
	40-3-2-3-2	Without	1.17%	3.87%	4.81%	5.27%	2.31%	1.17%	3.48%	5.27%
		With	3.37%	4.62%	0.26%	0.00%	2.73%	0.00%	2.20%	4.62%
Rule 5	20-3-2-3-2	Without	0.57%	0.78%	2.49%	1.42%	1.07%	0.57%	1.27%	2.49%
		With	0.00%	1.71%	0.92%	2.56%	0.57%	0.00%	1.15%	2.56%
	30-3-2-3-2	Without	2.61%	1.22%	3.94%	1.48%	3.32%	1.22%	2.52%	3.94%
		With	2.35%	3.19%	1.51%	0.55%	0.00%	0.00%	1.52%	3.19%
	40-3-2-3-2	Without	0.42%	3.10%	4.04%	4.50%	1.55%	0.42%	2.72%	4.50%
		With	1.47%	2.32%	0.18%	0.00%	2.97%	0.00%	1.39%	2.97%

According to the obtained results, rules 2, 3 and 5 have improved the results and can be used in the third step.

H. Stop condition

After the implementation of the algorithm and updating the solutions, the stop condition is checked. The stop condition is defined as a time limitation of 1,200 seconds, and the best solution obtained at this point is considered the solution of the algorithm.

V. CALCULATIONS AND NUMERICAL RESULTS METHOD

The mathematical model performance and the provided solution algorithm are examined in this section. The model is coded in GAMS software with win 32 24.1.3 specifications and the solution algorithm is coded in Java software with version 2020.1.2. The system specifications is Intel(R) Core™ i5-3230M CPU @ 2.60GHz. It is possible for solving the problem in a reasonable time using the model in small dimensions. With increasing of the problem dimensions, the time of solution the model is not cost-effective, so the optimal or near-optimal solutions have been calculated by the meta-heuristic algorithm in large dimensions. The ranges of different values for the problem parameters are shown in Table VII.

Table VII. Parameter values to create problem examples

Parameter	Range of parameter values	
	Small size	Large size
f	{2,3,4}	{4,6,8}
n	{3,4,5,6,7,8,10,12}	{25,35,45,55,65,75,85,95,100}
m_1	{2,4,6}	{2,4,6,8}
m_2	{2,3}	{2,3,4,5}
m_3	{2,3}	{2,3,4}
pf	U(1,100)	U(1,100)
ps	U(1,100)	U(1,100)
pt	U(1,100)	U(1,100)

The due date of job j is calculated based on equation (69) (Khare & Agrawal, 2021).

$$d_j = p_j * \left(1 + rnd * \left({}^{1}/_{f}\right) * \alpha\right) \tag{69}$$

In equation (69), the value of p_j is calculated using equation (70). The rnd is a random number in the range of [0,1]. f is the number of factories and α is considered equal to 0.7 according to various tests.

$$p_j = \max_k P_{j,k} + t_j + pt_j \tag{70}$$

In order to use of the BBO algorithm, the algorithm parameters values are set, firstly. Then the results are compared in two cases, small and large size.

A. Parameters setting

Inappropriate parameter values can definitely derail an algorithm from getting the desired results. Therefore, the parameters adjustment of the algorithm can improve its results. The algorithm parameters including population size (PS), unchanged transmission rate to the next generation (P_{slc}), mutation rate (P_{mut}), modification rate (P_{mig}), maximum immigration rate (I), maximum emigration rate (E) and the maximum mutation rate (M_{max}). Three levels are considered for each parameter which are shown in Table VIII.

Table VIII. Parameters Levels

Factor	Level	Value	Factor	Level	Value
PS	1	30	E	1	0.7
	2	50			
	3	80		2	0.9
P_{slc}	1	0.02			
	2	0.04		3	1
	3	0.06			
P_{mut}	1	0.2	I	1	0.7
	2	0.25		2	0.9
	3	0.3		3	1
P_{mig}	1	0.8	M_{max}	1	0.7
	2	0.9		2	0.9
	3	1		3	1

Parameter adjustment has been done by using one-way analysis of variance (ANOVA) to specify the appropriate level for each of the parameters. According to the number of parameters and levels, the orthogonal matrix has created 27 different combinations of parameter levels. For this purpose, an instance of the small size of the problem with $n = 8$, $f = 2$, $m_1 = 2$, $m_2 = 2$, $m_3 = 2$ and an instance of the large size with $n = 45$, $f = 4$, $m_1 = 2$, $m_2 = 4$, $m_3 = 2$ have been used to determine the values of the parameters. Each combination is executed 5 times by the algorithm. For each execution, the percentage of deviation from the best solution is calculated based on equation (71).

$$RE_i = {(solution_i - bestsol)}/{bestsol} \tag{71}$$

In equation (71), $solution_i$ is the obtained value of execution i and $bestsol$ is the best solution of 5 executions. The average of all deviations is calculated using equation (72).

$$ARE = {\sum_i RE_i}/{5} \tag{72}$$

For small-sized problems, the ARE values for 27 combinations are calculated and shown in Table IX.

According to the obtained ARE values from the different levels combination of the parameters, the related graph to the average and dispersion has been drawn to specify the appropriate level of each parameter. According to Figure 11.1, in order to determine the value of PS parameter, first the average deviation increases, then its value decreases at level 3 and its value is lower than level 1. Therefore, the appropriate value of this parameter equal $PS = 80$. The next parameter is P_{slc}, that it is shown in Figure 11.2. In this parameter, the average deviation has increased in levels 2 and 3 by increasing its value, but the dispersion has increased in level 2 and decreased in level 3. Level 1 has the lowest average value and dispersion deviation, so it is selected and the appropriate value of this parameter is $P_{slc} = 0.02$. In order to determine the value of P_{mut} and P_{mig} parameters, their graphs have been drawn in Figures 11.3. and 11.4. According to the figures, the lowest average value and dispersion is related to level 2 of these parameters, it is equal to

$P_{mut} = 0.25$ and $P_{mig} = 0.9$. Figure 11.5. shows the average and standard deviation of parameter E. By increasing the value of this parameter, the average and dispersion of the deviation increases. Therefore, the appropriate value of this parameter is at level 1 and equal $E = 0.7$. Figure 11.6 is used for parameter I. By increasing the value of this parameter, the average and dispersion of the deviation first decrease and then increase at level 3. Therefore, the appropriate value of this parameter is equal to $I = 0.9$. The last parameter is m_{max}, the average and dispersion of its deviation have been shown in Figure 11.7. For this parameter, the average value and dispersion increase in levels 2 and 3 by its increase. Therefore, the appropriate value of this parameter equal $m_{max} = 0.7$. The value of the problem parameters in small size is equal $PS = 80$, $P_{slc} = 0.0$, $P_{mut} = 0.25$. $P_{mig} = 0.$ $E = 0$ $I = 0.9$ and $m_{max} = 0.7$.

Table IX. Average relative deviation values for ANOVA instances in small-size

Experiment number	**Parameters level**							**ARE**
	PS	P_{slc}	P_{mut}	P_{mig}	**E**	**I**	m_{max}	
1	1	1	1	1	1	1	1	0.0204
2	1	1	1	1	2	2	2	0.0219
3	1	1	1	1	3	3	3	0.0210
4	1	2	2	2	1	1	1	0.0120
5	1	2	2	2	2	2	2	0.0320
6	1	2	2	2	3	3	3	0.0259
7	1	3	3	3	1	1	1	0.0384
8	1	3	3	3	2	2	2	0.0256
9	1	3	3	3	3	3	3	0.0521
10	2	1	2	3	1	2	3	0.0466
11	2	1	2	3	2	3	1	0.0241
12	2	1	2	3	3	1	2	0.0256
13	2	2	3	1	1	2	3	0.0136
14	2	2	3	1	2	3	1	0.0167
15	2	2	3	1	3	1	2	0.0759
16	2	3	1	2	1	2	3	0.0370
17	2	3	1	2	2	3	1	0.0556
18	2	3	1	2	3	1	2	0.0247
19	3	1	3	2	1	3	2	0.0037
20	3	1	3	2	2	1	3	0.0127
21	3	1	3	2	3	2	1	0.0287
22	3	2	1	3	1	3	2	0.0213
23	3	2	1	3	2	1	3	0.0519
24	3	2	1	3	3	2	1	0.0055
25	3	3	2	1	1	3	2	0.0296
26	3	3	2	1	2	1	3	0.0259
27	3	3	2	1	3	2	1	0.0389

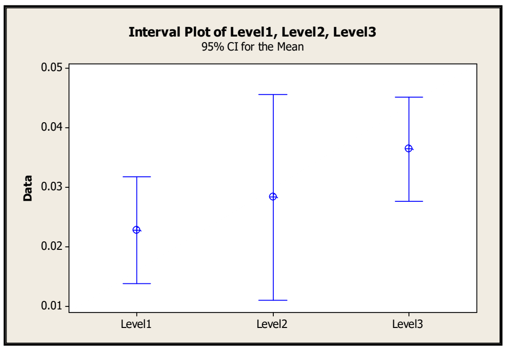

Fig. 11.2. Dispersion and standard deviation of P_{slc}

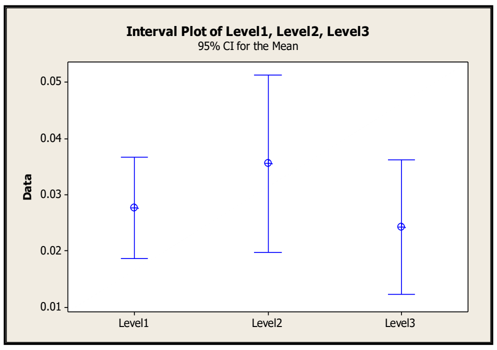

Fig. 11.1. Dispersion and standard deviation of PS

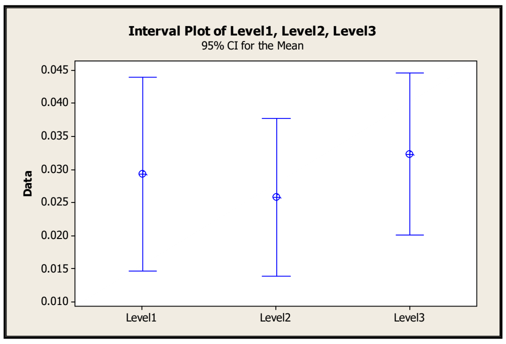

Fig. 11.4. Dispersion and standard deviation of P_{mig}

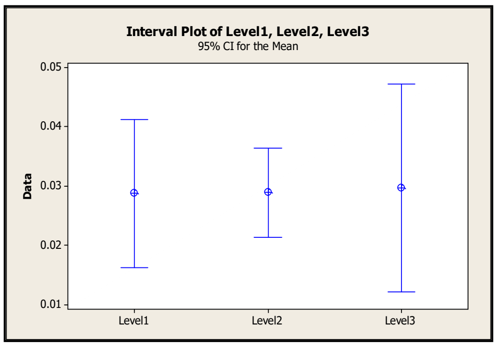

Fig. 11.3. Dispersion and standard deviation of P_{mut}

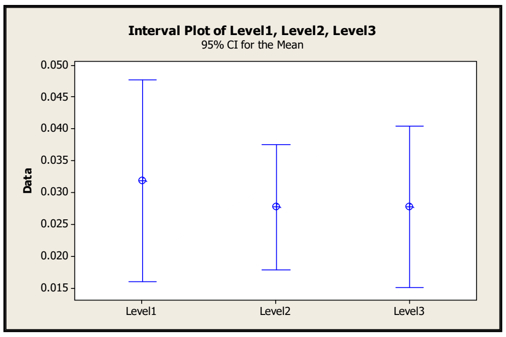

Fig. 11.6. Dispersion and standard deviation of I

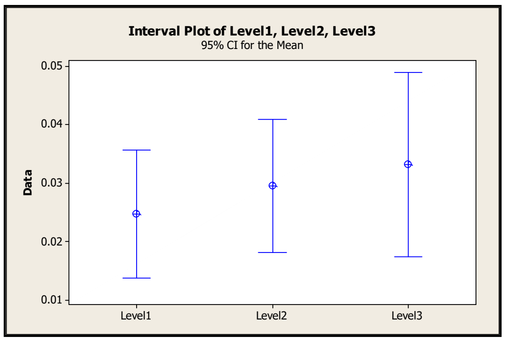

Fig. 11.5. Dispersion and standard deviation of E

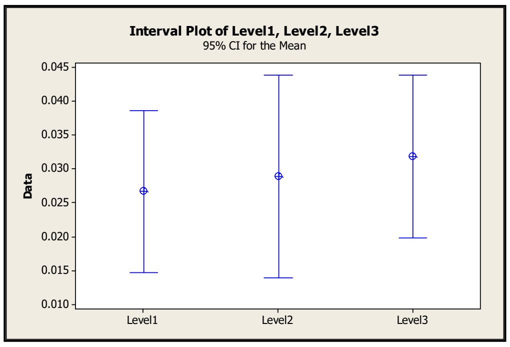

Fig. 11.7. Dispersion and standard deviation of m_{max} parameter

Fig. 11. Dispersion and standard deviation of algorithm parameters for different levels in small size

The ARE values for 27 combinations are calculated and are shown in Table X. for large-size problems.

Table X. Average relative deviation values for ANOVA samples in large size

Experiment number	Parameters level							ARE
	PS	P_{slc}	P_{mut}	P_{mig}	E	I	m_{max}	
1	1	1	1	1	1	1	1	0.0258
2	1	1	1	1	2	2	2	0.0507
3	1	1	1	1	3	3	3	0.0206
4	1	2	2	2	1	1	1	0.0211
5	1	2	2	2	2	2	2	0.0507
6	1	2	2	2	3	3	3	0.0228
7	1	3	3	3	1	1	1	0.0620
8	1	3	3	3	2	2	2	0.0534
9	1	3	3	3	3	3	3	0.0522
10	2	1	2	3	1	2	3	0.0222
11	2	1	2	3	2	3	1	0.0408
12	2	1	2	3	3	1	2	0.0324
13	2	2	3	1	1	2	3	0.0157
14	2	2	3	1	2	3	1	0.0814
15	2	2	3	1	3	1	2	0.0354
16	2	3	1	2	1	2	3	0.0131
17	2	3	1	2	2	3	1	0.0298
18	2	3	1	2	3	1	2	0.0185
19	3	1	3	2	1	3	2	0.0282
20	3	1	3	2	2	1	3	0.0210
21	3	1	3	2	3	2	1	0.0437
22	3	2	1	3	1	3	2	0.0260
23	3	2	1	3	2	1	3	0.0206
24	3	2	1	3	3	2	1	0.0154
25	3	3	2	1	1	3	2	0.0168
26	3	3	2	1	2	1	3	0.0301
27	3	3	2	1	3	2	1	0.0441

According to the obtained ARE values from the combination of different levels of the parameters, the related graph to the average and dispersion has been drawn to determine the appropriate level of each parameter. For the PS parameter, the dispersion and average diagram are according to Figure 12.1. By an increase of population size, the amount of dispersion and average deviation increases. In level 2, the average deviation is lower, but the value of dispersion is higher than level 1. In level 3, the amount of dispersion and average deviation is lower than the other two levels. Therefore, the appropriate value of the PS parameter is at level 3 and equal $PS = 80$. For P_{slc} parameter, by

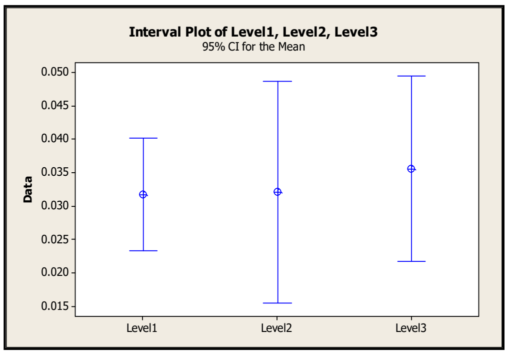

Fig. 12.2. Dispersion and standard deviation of P_{slc}

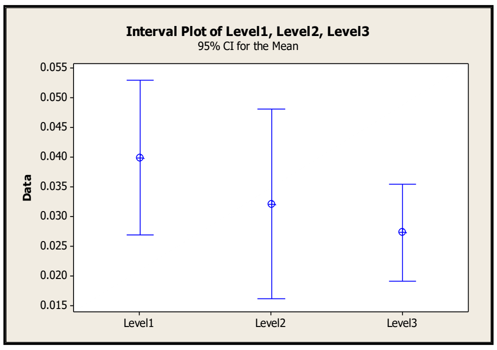

Fig. 12.1. Dispersion and standard deviation of PS

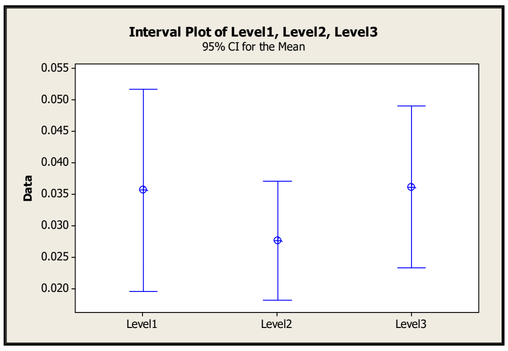

Fig. 12.4. Dispersion and standard deviation of P_{mig}

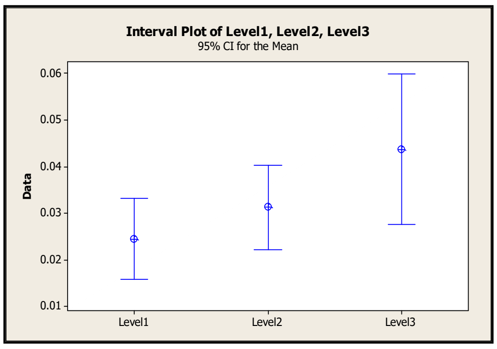

Fig. 12.3. Dispersion and standard deviation of P_{mut}

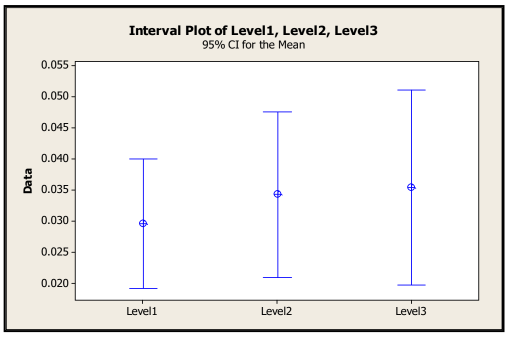

Fig. 12.6. Dispersion and standard deviation of I

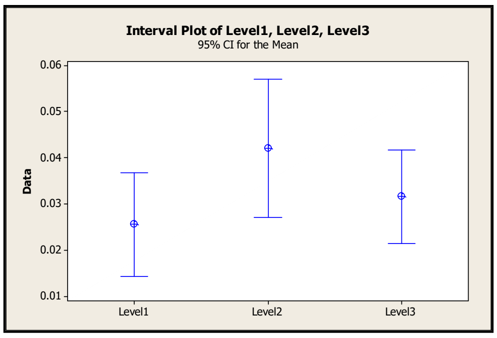

Fig. 12.5. Dispersion and standard deviation of E

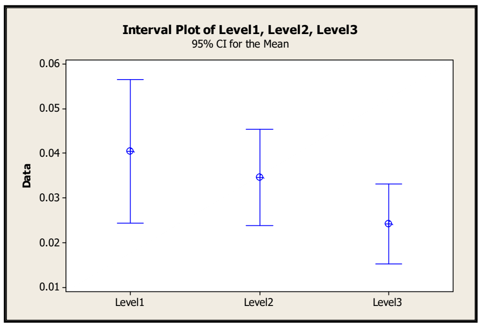

Figure 12.7. Dispersion and standard deviation of m_{max}

Fig. 12. Dispersion and standard deviation of algorithm parameters for different levels in large size

increasing the value of this parameter, the value of dispersion and average deviation increases based on Figure 12.2. In level 2, the average and deviation increases versus level 1. In level 3, the average value increases versus level 2, but the dispersion value decreases. The value of dispersion and average deviation for levels 2 and 3 is higher than level 1. Therefore, level 1 is the best value of this parameter and equal $P_{slc} = 0.02$. The next parameter is P_{mut}. For this parameter, the dispersion and average deviation diagram is according to Figure 12.3. By increasing the value of this parameter, the average and dispersion increase in levels 2 and 3. Therefore, the best value of this parameter is at level 1 and equal $P_{mut} = 0.2$. The dispersion and average deviation of P_{mig} parameter has been shown in Figure 12.4. By increasing its value, the average and dispersion decrease at level 2, but it increase again at level 3. Therefore, the appropriate value of this parameter is at level 2 and equal $P_{mig} = 0.9$. The value of parameter E is specified based on Figure 12.5. By increasing the value of this parameter, the average and dispersion increase at first and then decrease in level 3, but level 1 has the lowest value versus the other two levels. Therefore, the appropriate value of this parameter is $E = 0.7$. The appropriate value of parameter I has been shown in Figure 12.6. By increasing the value of this parameter, the average and dispersion of the parameter increases in levels 2 and 3. Therefore, the best value of this parameter exists on level 1 and equal $I = 0.7$. The value of parameter m_{max} by drawing the dispersion diagram and average deviation has been shown in Figure 12.7. For this parameter, with its increase, the average and dispersion decrease in levels 2 and 3. So, the appropriate value of this parameter equal $m_{max} = 1$. Therefore the value of the problem parameters in large size is equal to $PS = 80$, $P_{slc} = 0.02$, $P_{mut} = 0.2$, $P_{mig} = 0.9$, $E = 0.7$, $I = 0.7$ and $m_{max} = 1$.

Based on the results obtained from parameter settings, different combinations of the problem will be presented to evaluate the performance of the proposed algorithm and to solve the problem in large dimensions, creating both large and small sizes.

B. Analysis of algorithm performance in small size

The exact solution to the problem is obtained by coding the model in small dimensions. The performance of the algorithm is evaluated using the results from solving the model. Consequently, the performance of the improved algorithm (HBBO) is compared with BBO, HGA (as provided by Torkashvand et al., (2022)), and the exact solution across 32 different instances. The mathematical model and algorithms are solved with a time limitation of 3600 seconds and $0.5 * n * f$ seconds respectively. Each instance is executed 5 times by the algorithm, and the percentage of deviation from the best solution is calculated for each execution using equation (71). The minimum (Min), average (Ave) and maximum (Max) deviation values are computed for each instance. with the results summarized in Table XI. Based on the obtained results, the average ARE values for HGA, BBO and HBBO algorithms are 0.0620, 0.0348 and 0.0073, respectively in 32 instances. HBBO algorithm has the lowest average deviation, that indicates its high effectiveness in comparison to other algorithms. As the problem size increases, achieving the optimal solution becomes more challenging for the model due to time limitations. Due to the time limitation, the optimal solution has not been obtained in 4 instances, a memory error has occurred in 3 instances, which is shown with the symbol OM in the table. In 25 instances, the optimal solution has been gotten, and the deviation value is zero and it is shown in bold. The average solution time for the HBBO algorithm is 11 seconds, and for CPLEX is 656 seconds, proving the algorithm's efficiency according to the average deviation from the best solution in the HBBO algorithm.

At the end of Table XI, the average values of Min, Ave and Max are calculated for 32 instances. In order to compare the results of the algorithms, the average values of deviation from the best solutions are shown in Figure 13 for small size. According to the Figure, for all three criteria—Min, Ave, and Max—the HBBO algorithm exhibits the lowest values compared to the other algorithms. For the Min values, the algorithms show little difference, with results close to each other. However, for the Ave and Max values, the gap between the HBBO algorithm and the other algorithms widens, indicating its superior performance.

Table XI. Evaluation of the presented algorithm performance versus the exact solution in small size

Instance	n	f	m1	m2	m3	CPLEX		HGA			BBO			HBBO			Time
						ARE	Time	Min	ARE	Max	Min	ARE	Max	Min	ARE	Max	
1	3	2	2	2	2	0.000	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	3
2	3	2	4	2	2	0.000	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	3
3	3	2	6	2	2	0.000	1	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	3
4	4	2	2	2	2	0.000	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	4
5	4	2	6	2	2	0.000	2	0.000	0.037	0.111	0.000	0.030	0.148	0.000	0.000	0.000	4
6	4	3	4	2	2	0.000	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	6
7	5	2	2	2	2	0.000	2	0.000	0.048	0.242	0.000	0.000	0.000	0.000	0.000	0.000	5
8	5	2	6	2	2	0.000	3	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	5
9	5	2	4	3	3	0.000	2	0.000	0.043	0.214	0.000	0.033	0.119	0.000	0.000	0.000	5
10	5	3	2	2	2	0.000	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	8
11	5	3	4	2	2	0.000	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	8
12	6	2	2	2	2	0.000	3	0.000	0.014	0.035	0.000	0.028	0.081	0.000	0.000	0.000	6
13	6	3	2	2	2	0.000	5	0.000	0.038	0.125	0.000	0.025	0.063	0.000	0.000	0.000	9
14	6	4	4	3	3	0.000	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	12
15	6	4	3	2	2	0.000	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	12
16	7	2	2	2	2	0.000	4	0.000	0.087	0.207	0.000	0.039	0.109	0.000	0.004	0.022	7
17	7	3	2	2	3	0.000	4	0.000	0.100	0.188	0.000	0.038	0.188	0.000	0.000	0.000	11
18	7	4	2	3	2	0.000	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	14
19	7	4	4	2	3	0.000	5	0.000	0.100	0.250	0.000	0.083	0.167	0.000	0.017	0.083	14
20	7	4	6	3	3	0.000	6	0.000	0.063	0.158	0.000	0.000	0.000	0.000	0.000	0.000	14
21	8	2	2	2	3	0.000	69	0.000	0.067	0.140	0.000	0.026	0.042	0.005	0.007	0.009	8
22	8	3	2	2	3	0.000	89	0.000	0.089	0.188	0.024	0.068	0.094	0.000	0.007	0.024	12
23	8	3	6	3	3	0.000	331	0.035	0.119	0.163	0.050	0.082	0.121	0.000	0.006	0.014	12
24	8	4	2	2	2	0.000	7	0.000	0.062	0.138	0.000	0.028	0.069	0.000	0.007	0.034	16
25	10	2	4	2	3	0.012	3600	0.042	0.148	0.192	0.037	0.061	0.089	0.000	0.012	0.023	10
26	10	3	2	2	3	0.000	3600	0.042	0.123	0.218	0.035	0.079	0.155	0.000	0.008	0.014	15
27	10	4	3	3	2	0.000	550	0.061	0.124	0.220	0.000	0.056	0.098	0.000	0.019	0.049	20
28	10	4	6	3	3	0.062	3600	0.062	0.153	0.276	0.000	0.077	0.138	0.069	0.079	0.097	20
29	12	3	2	2	3	OM	2248	0.037	0.154	0.352	0.022	0.085	0.142	0.000	0.007	0.017	18
30	12	4	4	2	3	OM	1828	0.000	0.104	0.220	0.016	0.098	0.299	0.010	0.020	0.035	24
31	12	4	2	3	2	0.039	3600	0.065	0.147	0.326	0.035	0.094	0.152	0.000	0.016	0.035	24
32	12	4	6	3	3	OM	1422	0.068	0.163	0.227	0.026	0.083	0.152	0.000	0.024	0.047	24
Average							656	0.0129	0.0620	0.1309	0.0076	0.0348	0.0758	0.0026	0.0073	0.0157	11

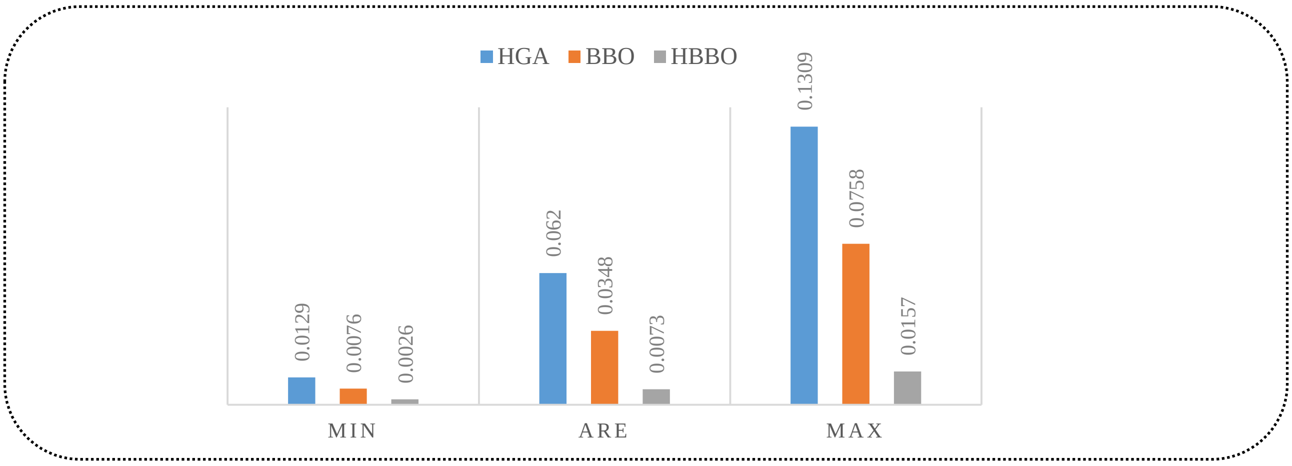

Fig. 13. Comparison of the Min, Ave and Max average values for the algorithms in small size

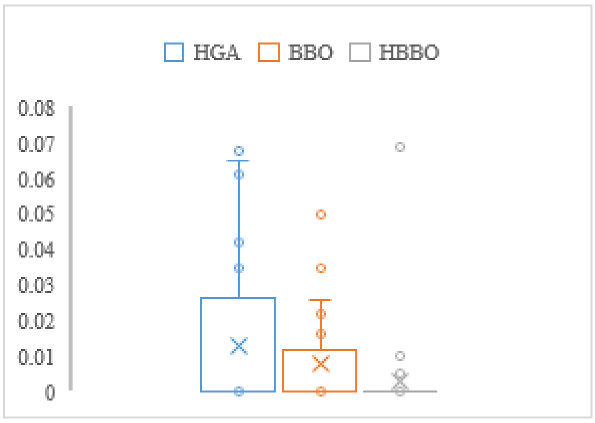

Fig. 14.1. Box graph for Min values

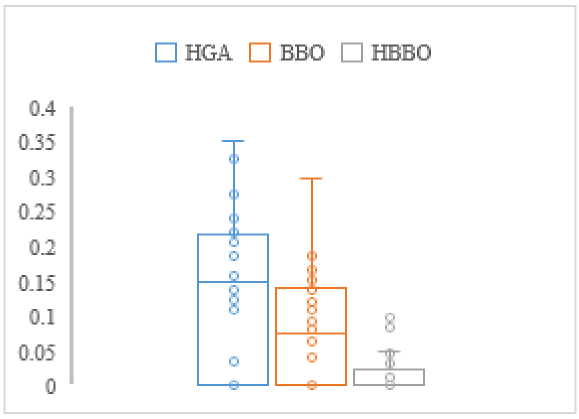

Fig. 14.2. Box graph for Max values

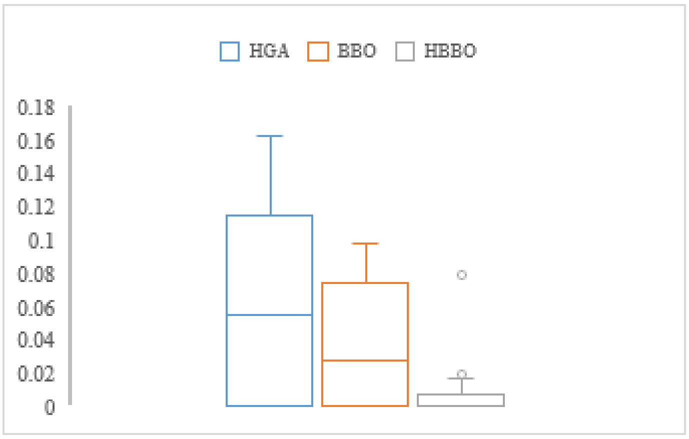

Fig. 14.3. Box graph for ARE values

Fig. 14. Box graph for Min, ARE and Max deviation values of each algorithm in 35 large size instances

To evaluate the performance of the algorithms in the small size, a Box and Whisker graph for the Min, Ave, and Max criteria is presented in Figure 14, based on the results from the 32 instances in Table XI. For the Min values, HBBO algorithm has the least average value based on Figure 14.1. Following this, the BBO and HGA algorithms exhibit relatively low average values as well. The Max values are illustrated in Figure 14.2, where the HBBO algorithm again shows the least average value and dispersion. The BBO and HGA algorithms follow in performance. After that, there are BBO and HGA algorithms. The algorithm's performance is compared with each other based on Ave values, in Figure 14.3. The HBBO algorithm has the least value compared to the other algorithms, and the instance with the highest deviation value is in the HGA algorithm.

C. Analysis of algorithm performance in large size

Following the assessment of efficiency and effectiveness for the solution algorithm in small sizes, the algorithm is applied to solve the problem in larger sizes. 35 instances of different parameter combinations have been created. Each instance is run 5 times with 1200-second time limitation and the best obtained solution is recorded in each execution. Like small-size instances, the percentage of deviation from the best solution h was calculated for each solution. The minimum (Min), average (Ave) and maximum (Max) deviation values were determined for each instance. The results of the calculations are shown in Table XII. The average values of ARE in 35 instances are 0.0852, 0.0477 and 0.0285 for HGA, BBO and HBBO algorithms, respectively. HBBO algorithm has the least average deviation.

Table XII. Comparison of solution algorithm results for large size

Instance	n	f	m_1	m_2	m_3	HGA			BBO			HBBO		
						Min	ARE	Max	Min	ARE	Max	Min	ARE	Max
1	25	4	2	2	2	0.014	0.041	0.105	0.001	0.027	0.067	0.000	0.013	0.028
2	25	4	4	3	3	0.072	0.095	0.120	0.008	0.025	0.048	0.000	0.007	0.017
3	25	6	6	3	2	0.029	0.061	0.079	0.000	0.022	0.033	0.000	0.012	0.028
4	25	8	8	2	3	0.000	0.044	0.092	0.007	0.036	0.082	0.003	0.016	0.024
5	35	4	2	2	2	0.146	0.158	0.170	0.027	0.037	0.044	0.000	0.021	0.045
6	35	4	4	3	3	0.000	0.042	0.077	0.055	0.078	0.097	0.002	0.046	0.073
7	35	6	6	2	2	0.026	0.042	0.058	0.027	0.032	0.037	0.000	0.018	0.037
8	35	8	8	3	3	0.093	0.119	0.147	0.044	0.052	0.067	0.046	0.059	0.083
9	45	4	2	4	2	0.000	0.045	0.110	0.000	0.031	0.052	0.003	0.015	0.033
10	45	6	4	2	3	0.047	0.068	0.091	0.045	0.054	0.063	0.000	0.013	0.030
11	45	6	2	3	4	0.111	0.141	0.174	0.004	0.024	0.046	0.000	0.016	0.028
12	45	8	4	2	2	0.046	0.084	0.174	0.000	0.015	0.048	0.006	0.022	0.031
13	55	8	8	4	4	0.073	0.093	0.102	0.018	0.027	0.045	0.000	0.014	0.036
14	55	4	4	5	3	0.000	0.046	0.102	0.039	0.072	0.096	0.030	0.056	0.092
15	55	6	2	3	3	0.023	0.050	0.089	0.000	0.027	0.066	0.011	0.022	0.028
16	55	8	2	2	2	0.010	0.046	0.080	0.015	0.036	0.085	0.000	0.012	0.031
17	65	8	8	4	2	0.000	0.041	0.072	0.030	0.046	0.055	0.011	0.035	0.050
18	65	4	4	5	3	0.007	0.020	0.031	0.008	0.019	0.029	0.000	0.011	0.019
19	65	6	4	3	4	0.083	0.116	0.142	0.030	0.045	0.068	0.035	0.041	0.044
20	65	8	2	2	3	0.031	0.043	0.068	0.017	0.044	0.066	0.000	0.015	0.036

Continue Table XII. Comparison of solution algorithm results for large size

Instance	n	f	m_1	m_2	m_3	HGA			BBO			HBBO		
						Min	ARE	Max	Min	ARE	Max	Min	ARE	Max
21	75	4	2	4	4	0.003	0.067	0.125	0.028	0.060	0.088	0.000	0.031	0.069
22	75	4	4	5	4	0.000	0.082	0.156	0.085	0.102	0.123	0.028	0.068	0.116
23	75	6	6	3	3	0.027	0.042	0.058	0.025	0.045	0.067	0.000	0.022	0.063
24	75	6	4	2	2	0.010	0.019	0.028	0.014	0.033	0.059	0.000	0.024	0.040
25	85	4	2	4	4	0.302	0.364	0.401	0.061	0.145	0.238	0.027	0.074	0.139
26	85	4	4	5	4	0.049	0.127	0.248	0.000	0.059	0.087	0.027	0.052	0.089
27	85	6	6	3	3	0.000	0.025	0.063	0.021	0.038	0.065	0.000	0.019	0.039
28	85	8	4	2	3	0.015	0.027	0.042	0.016	0.036	0.057	0.005	0.017	0.040
29	95	4	2	4	4	0.002	0.070	0.193	0.001	0.046	0.099	0.000	0.020	0.047
30	95	4	4	5	3	0.149	0.192	0.280	0.019	0.049	0.062	0.000	0.029	0.064
31	95	6	4	3	2	0.003	0.042	0.110	0.027	0.063	0.088	0.000	0.036	0.053
32	95	6	2	2	2	0.123	0.182	0.228	0.036	0.105	0.299	0.008	0.065	0.147
33	100	4	2	4	4	0.011	0.102	0.217	0.022	0.035	0.056	0.000	0.025	0.046
34	100	6	2	3	3	0.022	0.033	0.040	0.024	0.053	0.077	0.000	0.025	0.036
35	100	4	4	5	2	0.085	0.213	0.367	0.016	0.051	0.080	0.000	0.025	0.044
Average						0.0460	0.0852	0.1326	0.0220	0.0477	0.0782	0.0069	0.0285	0.0521

The average of Min, Ave and Max obtained results from solving the problem in 35 instances are calculated at the end of Table XII. For comparison the results of the algorithms, the average values are shown in Figure 15. In all three cases, the HBBO algorithm has the least value compared to other algorithms. For Ave, the value of BBO algorithm is almost half of HGA. Therefore, the HBBO algorithm performs better than other algorithms.

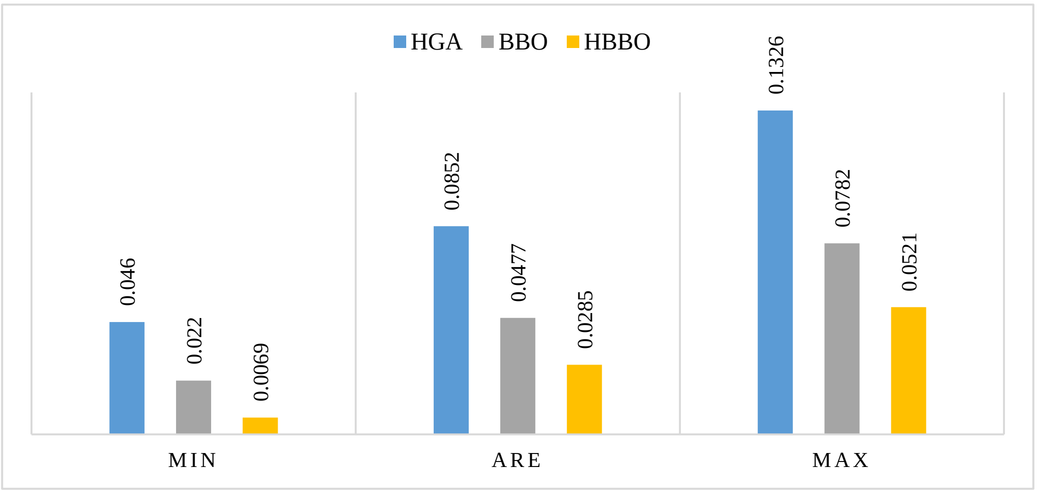

Fig. 15. Comparison of the Min, Ave and Max average values for the algorithms in large size

Based on the results obtained in Table XII, the performance of the algorithms under review has been analyzed by drawing the related graphs for the average values of the minimum, average, and maximum deviations in Figure 16. For Min values, the average line of HBBO algorithm is lower than other algorithms and its value equal 0.0069 in Figure 16.1. Also, the dispersion of result in this algorithm is less than other algorithms and BBO and HGA algorithms are in the next ranks, respectively. For Max values, the average and dispersion of the HBBO algorithm are less than the other algorithms according to Figure 16.2. After that, BBO and HGA algorithms have the least values, respectively.

For Ave values, like the Max graph, HBBO algorithm its value is 0.0285, has the least value versus the other algorithms according to Figure 16.3. For three all criteria Min, Ave and Max, the HBBO algorithm is superior to other algorithms, and it has achieved better results according to Figure 16.

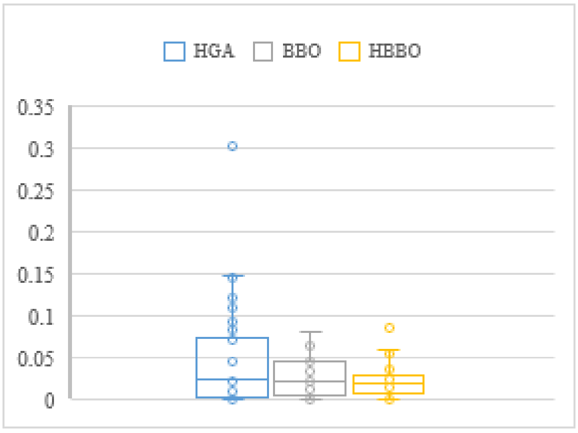

Fig. 16.1. Box graph for Min values

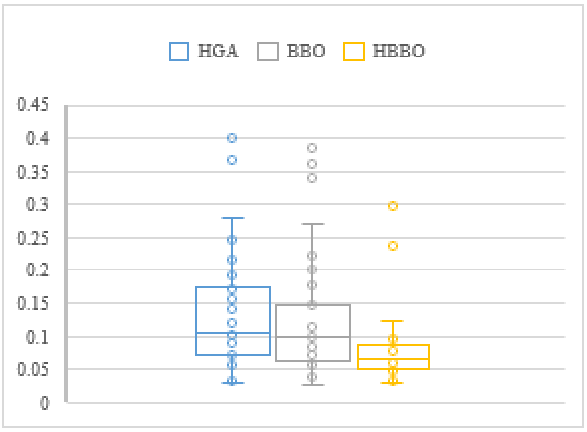

Fig. 16.2. Box graph for Max values

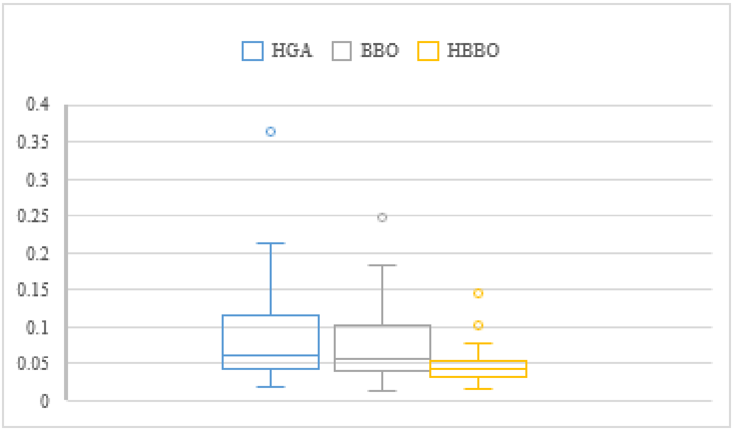

Figure 16.3. Box plot for ARE values

Fig. 16. Box graph for Min, ARE and Max deviation values of each algorithm in 35 large-size instances

VI. SENSITIVITY ANALYSIS

According to managerial insight regarding the provision of services and production of products, considering the appropriate values of parameters before building a production space or providing services can have a great impact on increasing income and customer satisfaction. For this purpose, conducting appropriate analyses is one of the necessities of starting a business. Therefore, in this section, the sensitivity analysis on the parameters of the problem is discussed.

Different objective functions can be considered according to management decisions. After determining the objective, it should be optimized by either minimizing the required capital or maximizing exploitation with a fixed asset. Therefore, the effect of different values of the parameters on the objective function should be checked. The influential parameters in the problem are n, f, m_1, m_2 and m_3. In order to check the effect of each parameter, 5 modes are presented in Table XIII, that their optimal solution is obtained in a small size.

Table XIII. The value of objective function for different parameters of problem

Parameter	n	f	m_1	m_2	m_3	SumTj
n	5	3	6	2	2	18.45
	6	3	6	2	2	37.96
	7	3	6	2	2	83.33
	8	3	6	2	2	141
	9	3	6	2	2	190
f	7	1	3	2	2	410
	7	2	3	2	2	133
	7	3	3	2	2	35
	7	4	3	2	2	1
	7	5	3	2	2	0
m_1	8	3	1	3	2	85
	8	3	2	3	2	54
	8	3	3	3	2	116
	8	3	4	3	2	126
	8	3	6	3	2	141
m_2	7	3	3	1	2	48
	7	3	3	2	2	35
	7	3	3	3	2	35
	7	3	3	4	2	35
	7	3	3	5	2	35
m_3	8	3	3	2	1	0
	8	3	3	2	2	116
	8	3	3	2	3	122
	8	3	3	2	4	116

The appropriate value of each parameter can be determined from the obtained results for each of problem parameters. Supposing that the other parameters are constant, as the value of n increases, the objective function value increases according to Figure 17.1. Therefore, the appropriate value of parameter n can be determined through the objective function value. For parameter f by increasing the value of this parameter, the value of the objective function decreases according to Figure 17.2. Therefore, the appropriate number of factories is determined according to the available capital and the objective function value. For parameter m_1, as this parameter increases, the objective function value decreases at first and then increases according to Figure 17.3. Therefore, the appropriate number of factories is determined according to the requirements, available capital and the value of the objective function. With increase value of parameter m_2, the value of the objective function decreases then remains constant at value of 35 according to Figure 17.4. Therefore, to reduce costs, its value can be considered equal 2. With an increase the value of parameter m_3, the objective function value increases according to Figure 17.5. For $m_3 = 2$ and $m_3 = 4$, the objective function value equals 116, and for the high diversity of products, the value 4 can be chosen.

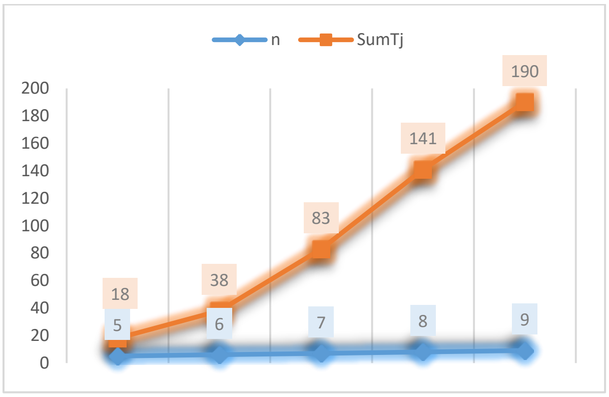

Fig. 17.1.The objective function variations for different values of n

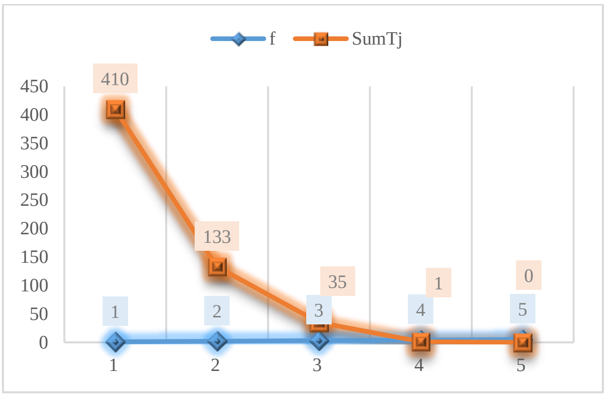

Fig. 17.2. The objective function variations for different values of f

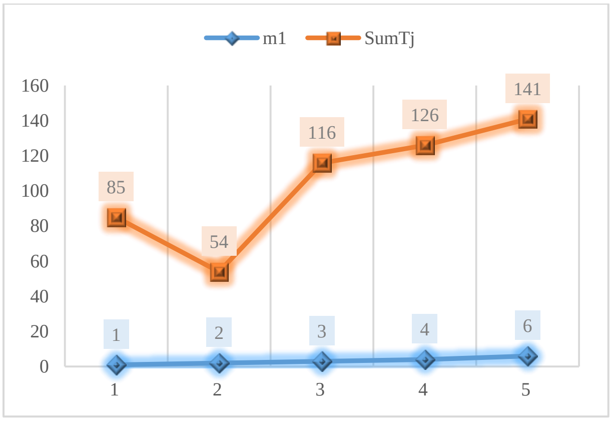

Fig. 17.3. The objective function variations for different values of m_1

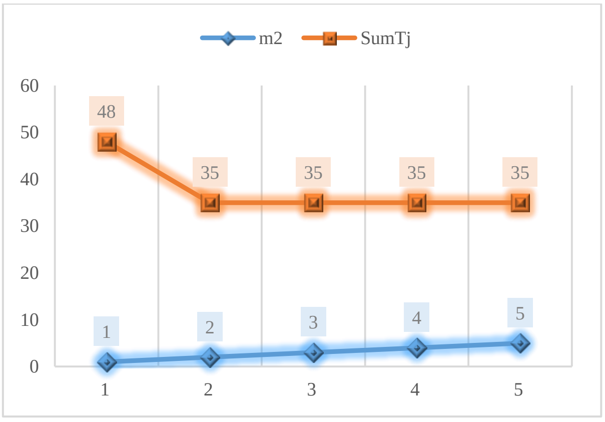

Fig. 17.4. The objective function variations for different values of m_2

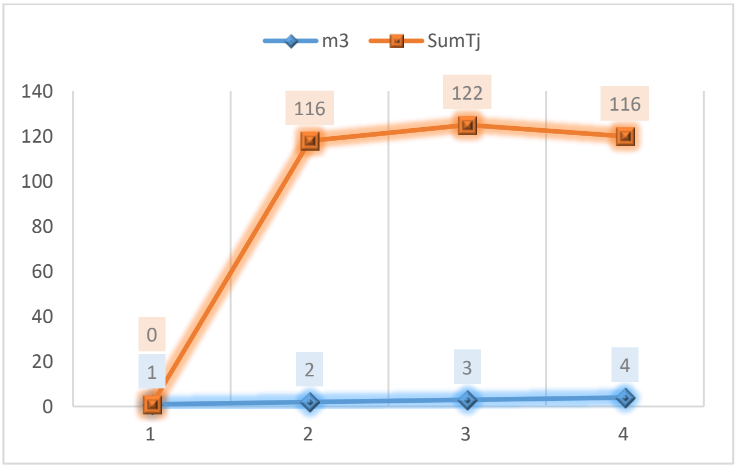

Fig. 17.5. The objective function variations for different values of m_3

Fig. 17. The objective function variations for different values of parameter n

To examine the effect of different values of n, f, m_1, m_2 and m_3 on the algorithms performance, the average Ave for different values of the parameters are caculated based on Table XII for large sizes of problem.

The graph of the deviation from the best solution for different values of the parameters is drawn in Figure 18. In these graphs, the average value of Ave has been calculated for the instances including each parameter. The performance of the algorithms is investigated for different values of n in Figure 18.1. According to this, the HBBO algorithm has the least value versus the other algorithms for all n values. By increasing the value of n, the dispersion has increased for the values of 35, 55, 65, 75 and 85. Different values of factories number (f) are investigated in Figure 18.2. By increasing f, the average deviation from the best solution has decreased for all algorithms. Different values of m_1 have been investigated based on Figure 18.3. In this graph, the HBBO algorithm has the least values versus the other algorithms, and the value of dispersion has increased by increase the value of this parameter. To check the effect of machines number in the second step (m_2), the graph is drawn in Figure 18.4. The HBBO algorithm has the least value compared to the other algorithms in this graph. For all algorithms, the amount of deviation increases for value 5 than value 4. Different values of parameter m_3 are checked in Figure 18.5. For the HBBO and BBO algorithms, the dispersion value increases by the increase of the parameter value, but for the HGA algorithm, it decreases in the value 3 and then increases in the value 4. For all graphs, the least value belongs to the HBBO algorithm.

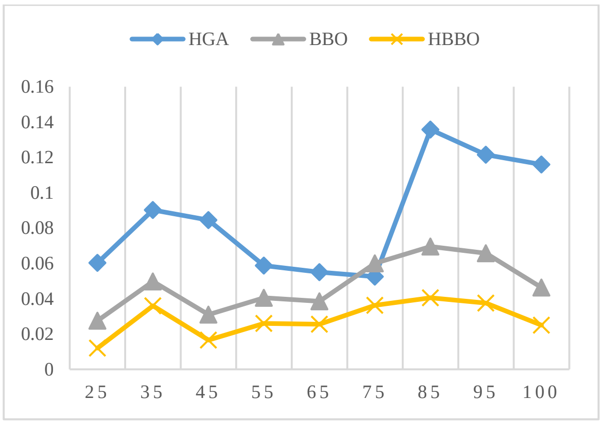

Fig. 18.1. The average deviation for different values of n

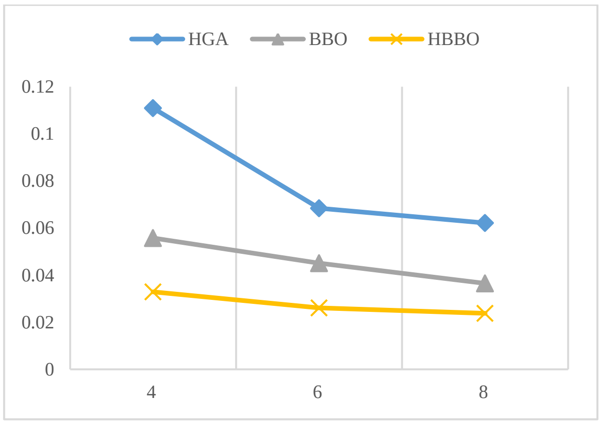

Fig. 18.2. The average deviation for different values of f

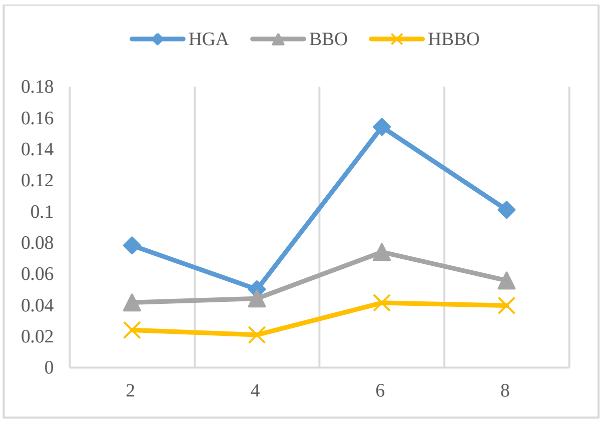

Fig. 18.3. The average deviation for different values of m_1

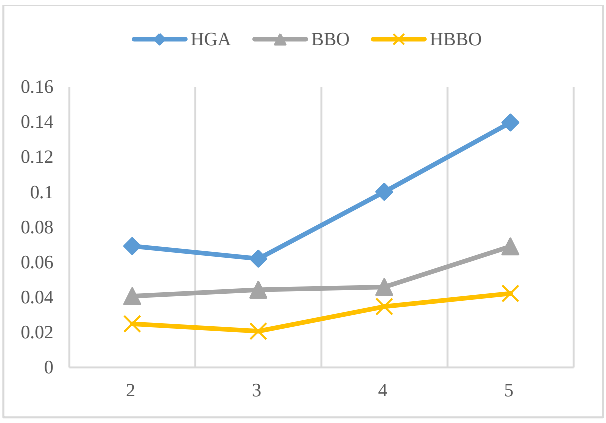

Fig. 18.4. The average deviation for different values of m_2

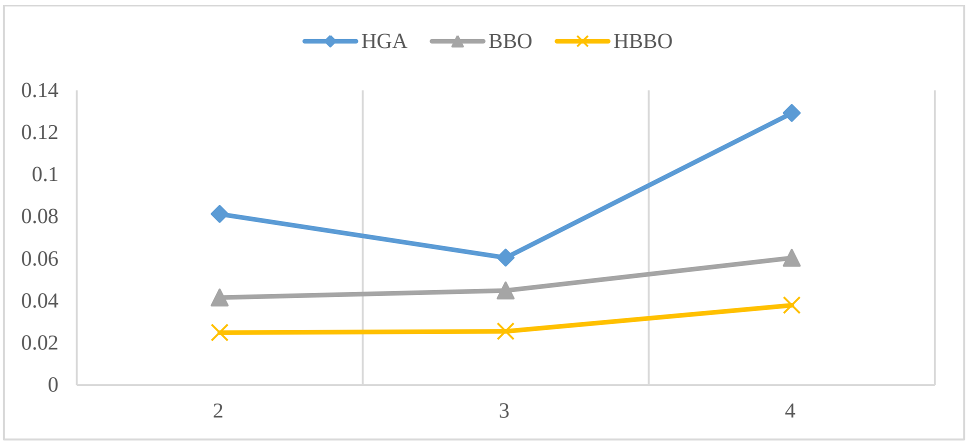

Fig. 18.5. The average deviation for different values of m_3

Fig. 18. The average deviation for different values of problem parameters

VII. CONCLUSION

Effective planning and scheduling of jobs is fundamental to optimizing production and service complexes for both profit and customer satisfaction and scheduling of jobs plays a crucial role in achieving this balance. So in this article, a flow shop scheduling problem was presented in the production-assembly field in parallel factories. This problem is used for the production of various products. Each factory consists of three steps: The first step includes dedicated parallel machines that produce multiple components of a product. The second step has identical parallel machines that assemble the produced parts of the previous step. In the third step, there are dedicated parallel machines that perform post-assembly operations, and each product can only be processed by one them.

There are many applications in real world such as production of personal computers, manufacture of car, printing of invoices, etc. According to the managerial insight, different objective functions can be considered. In this article, two models were presented to minimize the total tardiness times based on the sequence and position of jobs in small sizes. Because of the time limitation and processor power, achieving the optimal solution is difficult in large sizes and the problem is NP-hard, so a developed mode of Biogeography Based Optimization (BBO) meta-heuristic algorithm was presented from a combination with the dominance rules that is called Hybrid BBO (HBBO). Also, to check the performance of the presented algorithm, its results were compared with the Hybrid Genetic Algorithm (HGA) algorithm used, which is close to the problem under investigation.

For solving the problem, In order to reach the better results and increase the efficiency of the presented algorithm, the parameters of the algorithm were adjusted using the one-way analysis of variance (ANOVA) method. The obtained results from solving the problem show the proper performance of the presented algorithm versus the other algorithms. Managers must make the right decisions to achieve their goals in different situations. In order to choose the right parameters of the problem the sensitivity analysis has been performed on the parameters of the problem at the end of the research, so that an appropriate decision can be made at the appropriate time. Some future studies, can be mentioned such as: 1- representation of new meta-heuristic algorithms, 2- considering the same parallel machines for each type of dedicated parallel machines in the third step, 3- considering other objective functions according to the conditions, 4- increase the number of steps by considering cases like transport between the steps or packaging of products.

REFERENCES

Al-Anzi, F. S., & Allahverdi, A. (2009). Heuristics for a two-stage assembly flowshop with bicriteria of maximum lateness and makespan. *Computers & Operations Research*, *36*(9), 2682-2689.

Allahverdi, A., Aydilek, H., & Aydilek, A. (2016). Two-stage assembly scheduling problem for minimizing total tardiness with setup times. *Applied Mathematical Modelling*, *40*(17-18), 7796-7815.

Baker, K. R., & Trietsch, D. (2009). *Principles of sequencing and scheduling*. John Wiley & Sons.

Basir, S. A., Mazdeh, M. M., & Namakshenas, M. (2018). Bi-level genetic algorithms for a two-stage assembly flow-shop scheduling problem with batch delivery system. *Computers & Industrial Engineering*, *126*, 217-231.

Campos, S. C., Arroyo, J. E. C., & Tavares, R. G. (2017). A general VNS heuristic for a three-stage assembly flow shop scheduling problem. In *Intelligent Systems Design and Applications: 16th International Conference on Intelligent Systems Design and Applications (ISDA 2016) held in Porto, Portugal, December 16-18, 2016* (pp. 955-964). Springer International Publishing.

Chen, S., Pan, Q. K., Gao, L., & Sang, H. Y. (2021). A population-based iterated greedy algorithm to minimize total flowtime for the distributed blocking flowshop scheduling problem. *Engineering Applications of Artificial Intelligence*, *104*, 104375.

Chung, T. P., & Chen, F. (2019). A complete immunoglobulin-based artificial immune system algorithm for two-stage assembly flowshop scheduling problem with part splitting and distinct due windows. *International Journal of Production Research*, *57*(10), 3219-3237.

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

Deng, C., Hu, R., Qian, B., & Jin, H. P. (2021). Hybrid estimation of distribution algorithm for solving three-stage multiobjective integrated scheduling problem. *Complexity*, *2021*, 1-18.

Esmaeili, M., Ahmadizar, F., & Sadeghi, H. (2021). Minimizing the sum of earliness and tardiness in single-machine scheduling. *Journal of Quality Engineering and Production Optimization*, *6*(2), 59-78.

Fattahi, P., Hosseini, S. M. H., Jolai, F., & SAFI, S. A. (2014). Multi-objective scheduling problem in a three-stage production system. *Intternattiionall Journall off Industtriiall Engiineeriing & Producttiion* Research, *25*(1), 1-12.

Fernandez-Viagas, V., Perez-Gonzalez, P., & Framinan, J. M. (2018). The distributed permutation flow shop to minimise the total flowtime. *Computers & Industrial Engineering*, *118*, 464-477.

Ferone, D., Hatami, S., González-Neira, E. M., Juan, A. A., & Festa, P. (2020). A biased-randomized iterated local search for the distributed assembly permutation flow-shop problem. *International Transactions in Operational Research*, *27*(3), 1368-1391.

Framinan, J. M., & Perez-Gonzalez, P. (2018). Order scheduling with tardiness objective: Improved approximate solutions. *European Journal of Operational Research*, *266*(3), 840-850.

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.

Gonzalez-Neira, E. M., Ferone, D., Hatami, S., & Juan, A. A. (2017). A biased-randomized simheuristic for the distributed assembly permutation flowshop problem with stochastic processing times. *Simulation Modelling Practice and Theory*, *79*, 23-36.

Hatami, S., Ruiz, R., & Andres-Romano, C. (2013). The distributed assembly permutation flowshop scheduling problem. *International Journal of Production Research*, *51*(17), 5292-5308.

Hosseini, S. M. H., Sana, S. S., & Rostami, M. (2022). Assembly flow shop scheduling problem considering machine eligibility restrictions and auxiliary resource constraints. *International Journal of Systems Science: Operations & Logistics*, *9*(4), 512-528.

Huang, Y. Y., Pan, Q. K., & Gao, L. (2023). An effective memetic algorithm for the distributed flowshop scheduling problem with an assemble machine. *International Journal of Production Research*, *61*(6), 1755-1770.

Huang, Y. Y., Pan, Q. K., Huang, J. P., Suganthan, P. N., & Gao, L. (2021). An improved iterated greedy algorithm for the distributed assembly permutation flowshop scheduling problem. *Computers & Industrial Engineering*, *152*, 107021.

Jia, Y., Yan, Q., & Wang, H. (2023). Q-learning driven multi-population memetic algorithm for distributed three-stage assembly hybrid flow shop scheduling with flexible preventive maintenance. *Expert Systems with Applications*, 120837.

Ji, M., Yang, Y., Duan, W., Wang, S., & Liu, B. (2016, July). Scheduling of no-wait stochastic distributed assembly flowshop by hybrid PSO. In *2016 IEEE congress on evolutionary computation (CEC)* (pp. 2649-2654). IEEE.

Jung, S., Woo, Y. B., & Kim, B. S. (2017). Two-stage assembly scheduling problem for processing products with dynamic component-sizes and a setup time. *Computers & Industrial Engineering*, *104*, 98-113.

Khare, A., & Agrawal, S. (2021). Effective heuristics and metaheuristics to minimise total tardiness for the distributed permutation flowshop scheduling problem. *International Journal of Production Research*, *59*(23), 7266-7282.

Komaki, G. M., & Kayvanfar, V. (2015). Grey Wolf Optimizer algorithm for the two-stage assembly flow shop scheduling problem with release time. *Journal of Computational Science*, *8*, 109-120.

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.

Lee, I. S. (2013). Minimizing total tardiness for the order scheduling problem. *International Journal of Production Economics*, *144*(1), 128-134.

Lei, D., & Liu, M. (2020). An artificial bee colony with division for distributed unrelated parallel machine scheduling with preventive maintenance. *Computers & Industrial Engineering*, *141*, 106320.

Lei, D., & Su, B. (2023). A multi-class teaching–learning-based optimization for multi-objective distributed hybrid flow shop scheduling. *Knowledge-Based Systems*, *263*, 110252.

Leung, J. Y. T., Li, H., & Pinedo, M. (2005). Order scheduling in an environment with dedicated resources in parallel. *Journal of Scheduling*, *8*, 355-386.

Li, M., Su, B., & Lei, D. (2021). A novel imperialist competitive algorithm for fuzzy distributed assembly flow shop scheduling. *Journal of Intelligent & Fuzzy Systems*, *40*(3), 4545-4561.

Li, X., Zhang, X., Yin, M., & Wang, J. (2015, May). A genetic algorithm for the distributed assembly permutation flowshop scheduling problem. In *2015 IEEE Congress on Evolutionary Computation (CEC)* (pp. 3096-3101). IEEE.

Li, X., Zhang, Z. Q., Hu, R., Qian, B., & Li, K. (2023, July). Hyper-heuristic Three-Dimensional Estimation of Distribution Algorithm for Distributed Assembly Permutation Flowshop Scheduling Problem. In *International Conference on Intelligent Computing* (pp. 386-396). Singapore: Springer Nature Singapore.

Liao, C. J., Lee, C. H., & Lee, H. C. (2015). An efficient heuristic for a two-stage assembly scheduling problem with batch setup times to minimize makespan. *Computers & Industrial Engineering*, *88*, 317-325.

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*, 128-136.

Lin, T. L., Horng, S. J., Kao, T. W., Chen, Y. H., Run, R. S., Chen, R. J., ... & Kuo, I. H. (2010). An efficient job-shop scheduling algorithm based on particle swarm optimization. *Expert Systems with Applications*, *37*(3), 2629-2636.

Ma, H. (2010). An analysis of the equilibrium of migration models for biogeography-based optimization. *Information Sciences*, *180*(18), 3444-3464.

Mahabadpour, M., Naderi, B., & Mohammadi, M. (2020). An effective model and algorithm for two-stage assembly flow shop problems. *International Journal of Services and Operations Management*, *35*(1), 100-114.

Maleki-Darounkolaei, A., Modiri, M., Tavakkoli-Moghaddam, R., & Seyyedi, I. (2012). A three-stage assembly flow shop scheduling problem with blocking and sequence-dependent set up times. *Journal of Industrial Engineering International*, *8*, 1-7.

Mozdgir, A., Fatemi Ghomi, S. M. T., Jolai, F., & Navaei, J. (2013). Two-stage assembly flow-shop scheduling problem with non-identical assembly machines considering setup times. *International Journal of Production Research*, *51*(12), 3625-3642.

Naderi, B., & Ruiz, R. (2010). The distributed permutation flowshop scheduling problem. *Computers & operations research*, *37*(4), 754-768.

Ochi, H., & Driss, O. B. (2019). Scheduling the distributed assembly flowshop problem to minimize the makespan. *Procedia Computer Science*, *164*, 471-477.

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*, 105492.

Pan, Y. R., Chen, Q. D., & Pan, Q. K. (2018, July). An Effective Fruit Fly Optimization for the Distributed Assembly Flowshop Scheduling Problem. In *2018 37th Chinese Control Conference (CCC)* (pp. 8374-8378). IEEE.

Potts, C. N., Sevast'Janov, S. V., Strusevich, V. A., Van Wassenhove, L. N., & Zwaneveld, C. M. (1995). The two-stage assembly scheduling problem: complexity and approximation. *Operations Research*, *43*(2), 346-355.

Pourhejazy, P., Cheng, C. Y., Ying, K. C., & Lin, S. Y. (2021). Supply chain-oriented two-stage assembly flowshops with sequence-dependent setup times. *Journal of Manufacturing Systems*, *61*, 139-154.

Pourhejazy, P., Cheng, C. Y., Ying, K. C., & Nam, N. H. (2023). Meta-Lamarckian-based iterated greedy for optimizing distributed two-stage assembly flowshops with mixed setups. *Annals of Operations Research*, *322*(1), 125-146.

Rastgar, I., Rezaean, J., Mahdavi, I., & Fattahi, P. (2021). Opportunistic maintenance management for a hybrid flow shop scheduling problem. *Journal of Quality Engineering and Production Optimization*, *6*(2), 17-30.

Rostami, M., & Shad, S. (2020). A Hybrid Bee Algorithm for Two-Machine Flow-Shop Scheduling Problems with Batch Delivery. *Journal of Quality Engineering and Production Optimization*, *5*(1), 137-164.

Samarghandi, H., & Firouzi Jahantigh, F. (2019). Comparing Mixed-Integer and Constraint Programming for the No-Wait Flow Shop Problem with Due Date Constraints. *Journal of Quality Engineering and Production Optimization*, *4*(1), 17-24.

Shoaardebili, N., & Fattahi, P. (2015). Multi-objective meta-heuristics to solve three-stage assembly flow shop scheduling problem with machine availability constraints. *International Journal of Production Research*, *53*(3), 944-968.

Shao, W., Pi, D., & Shao, Z. (2018). Local search methods for a distributed assembly no-idle flow shop scheduling problem. *IEEE Systems Journal*, *13*(2), 1945-1956.

Shao, Z., Shao, W., & Pi, D. (2020). Effective constructive heuristic and metaheuristic for the distributed assembly blocking flow-shop scheduling problem. *Applied Intelligence*, *50*, 4647-4669.

Sheikh, S., Komaki, G. M., & Kayvanfar, V. (2018). Multi objective two-stage assembly flow shop with release time. *Computers & Industrial Engineering*, *124*, 276-292.

Simon, D. (2008). Biogeography-based optimization. *IEEE transactions on evolutionary computation*, *12*(6), 702-713.

Song, H. B., & Lin, J. (2021). A genetic programming hyper-heuristic for the distributed assembly permutation flow-shop scheduling problem with sequence dependent setup times. *Swarm and Evolutionary Computation*, *60*, 100807.

Soon, G. K., Guan, T. T., On, C. K., Alfred, R., & Anthony, P. (2013, November). A comparison on the performance of crossover techniques in video game. In *2013 IEEE international conference on control system, computing and engineering* (pp. 493-498). IEEE.

Tian, Y., Liu, D., Yuan, D., & Wang, K. (2013). A discrete PSO for two-stage assembly scheduling problem. *The International Journal of Advanced Manufacturing Technology*, *66*, 481-499.

Torabzadeh, E., & Zandieh, M. (2010). Cloud theory-based simulated annealing approach for scheduling in the two-stage assembly flowshop. *Advances in Engineering Software*, *41*(10-11), 1238-1243.

Torkashvand, M., Ahmadizar, F., & Farughi, H. (2022). Distributed Production Assembly Scheduling with Hybrid Flowshop in Assembly Stage. *International Journal of Engineering*, *35*(5), 1037-1055.

Torkashvand, M., & Ahmadizar, F. (2024). One-step production and two-step assembly scheduling in identical factories. *Engineering Optimization*, *56*(8), 1255-1277.

Wagneur, E., & Sriskandarajah, C. (1993). Openshops with jobs overlap. *European Journal of Operational Research, 71*(3), 366–378.

Wang, J., Lei, D., & Li, M. (2022). A Q-Learning-Based Artificial Bee Colony Algorithm for Distributed Three-Stage Assembly Scheduling with Factory Eligibility and Setup Times. *Machines*, *10*(8), 661.

Wang, K., Ma, W. Q., Luo, H., & Qin, H. (2016). Coordinated scheduling of production and transportation in a two-stage assembly flowshop. *International journal of production research*, *54*(22), 6891-6911.

Wang, Z., Deng, Q., Zhang, L., Li, H., & Li, F. (2023). Joint optimization of integrated mixed maintenance and distributed two-stage hybrid flow-shop production for multi-site maintenance requirements. *Expert Systems with Applications*, *215*, 119422.

Wu, C. C., Azzouz, A., Chung, I. H., Lin, W. C., & Ben Said, L. (2019). A two-stage three-machine assembly scheduling problem with deterioration effect. *International Journal of Production Research*, *57*(21), 6634-6647.

Wu, C. C., Bai, D., Azzouz, A., Chung, I. H., Cheng, S. R., Jhwueng, D. C., ... & Said, L. B. (2020). A branch-and-bound algorithm and four metaheuristics for minimizing total completion time for a two-stage assembly flow-shop scheduling problem with learning consideration. *Engineering Optimization*, *52*(6), 1009-1036.

Wu, C. C., Liu, S. C., Lin, T. Y., Yang, T. H., Chung, I. H., & Lin, W. C. (2018). Bicriterion total flowtime and maximum tardiness minimization for an order scheduling problem. *Computers & Industrial Engineering*, *117*, 152-163.

Yan, H. S., Wan, X. Q., & Xiong, F. L. (2014). A hybrid electromagnetism-like algorithm for two-stage assembly flow shop scheduling problem. *International Journal of Production Research*, *52*(19), 5626-5639.

Yang, S., & Xu, Z. (2021). The distributed assembly permutation flowshop scheduling problem with flexible assembly and batch delivery. *International Journal of Production Research*, *59*(13), 4053-4071.

Yang, Y., Li, P., Wang, S., Liu, B., & Luo, Y. (2017, June). Scatter search for distributed assembly flowshop scheduling to minimize total tardiness. In *2017 IEEE Congress on evolutionary computation (CEC)* (pp. 861-868). IEEE.

Yin, Y., Wu, W. H., Wu, W. H., & Wu, C. C. (2014). A branch-and-bound algorithm for a single machine sequencing to minimize the total tardiness with arbitrary release dates and position-dependent learning effects. *Information Sciences*, *256*, 91-108.

Ying, K. C., & Lin, S. W. (2023). Reinforcement learning iterated greedy algorithm for distributed assembly permutation flowshop scheduling problems. *Journal of Ambient Intelligence and Humanized Computing*, *14*(8), 11123-11138.

Yu, F., Lu, C., Zhou, J., & Yin, L. (2024). Mathematical model and knowledge-based iterated greedy algorithm for distributed assembly hybrid flow shop scheduling problem with dual-resource constraints. *Expert Systems with Applications*, *239*, 122434.

Zhang, G., & Xing, K. (2018). Memetic social spider optimization algorithm for scheduling two-stage assembly flowshop in a distributed environment. *Computers & Industrial Engineering*, *125*, 423-433.

Zhang, G., Xing, K., & Cao, F. (2018). Scheduling distributed flowshops with flexible assembly and set-up time to minimise makespan. *International Journal of Production Research*, *56*(9), 3226-3244.

Zhang, Y., Zhou, Z., & Liu, J. (2010). The production scheduling problem in a multi-page invoice printing system. *Computers & Operations Research*, *37*(10), 1814-1821.

Zhang, Z., & Tang, Q. (2021). Integrating flexible preventive maintenance activities into two-stage assembly flow shop scheduling with multiple assembly machines. *Computers & industrial engineering*, *159*, 107493.

Zhang, Z. Q., Qian, B., Hu, R., Jin, H. P., & Wang, L. (2021). A matrix-cube-based estimation of distribution algorithm for the distributed assembly permutation flow-shop scheduling problem. *Swarm and Evolutionary Computation*, *60*, 100785.

Zhao, F., Hu, X., Wang, L., Xu, T., Zhu, N., & Jonrinaldi. (2023). A reinforcement learning-driven brain storm optimisation algorithm for multi-objective energy-efficient distributed assembly no-wait flow shop scheduling problem. *International Journal of Production Research*, *61*(9), 2854-2872.

Zhao, F., Qin, S., Zhang, Y., Ma, W., Zhang, C., & Song, H. (2019). A hybrid biogeography-based optimization with variable neighborhood search mechanism for no-wait flow shop scheduling problem. *Expert Systems with Applications*, *126*, 321-339.

Zhao, F., Zhao, J., Wang, L., & Tang, J. (2021). An optimal block knowledge driven backtracking search algorithm for distributed assembly No-wait flow shop scheduling problem. *Applied Soft Computing*, *112*, 107750.

Zhao, F., Zhang, L., Cao, J., & Tang, J. (2021). A cooperative water wave optimization algorithm with reinforcement learning for the distributed assembly no-idle flowshop scheduling problem. *Computers & Industrial Engineering*, *153*, 107082.

Zheng, J., & Wang, Y. (2021). A Hybrid Bat Algorithm for Solving the Three-Stage Distributed Assembly Permutation Flowshop Scheduling Problem. *Applied Sciences*, *11*(21), 10102.

Zou, Y., Wang, D., Lin, W. C., Chen, J. Y., Yu, P. W., Wu, W. H., & Wu, C. C. (2020). Two-stage three-machine assembly scheduling problem with sum-of-processing-times-based learning effect. *Soft Computing*, *24*, 5445-5462.