Scientia Iranica E (2023) 30(2), 738–756

Sharif University of Technology
Scientia Iranica
Transactions E: Industrial Engineering
http://scientiairanica.sharif.edu

A modified integer and categorical PSO algorithm for solving integrated process planning, dynamic scheduling, and due date assignment problem

C. Erden[a], H.İ. Demir[b], and O. Canpolat[b,*]

a. *Department of International Trade and Finance, Faculty of Applied Science, Sakarya University of Applied Science, Sakarya, Türkiye.*
b. *Department of Industrial Engineering, Faculty of Engineering, Sakarya University, Türkiye.*

Received 16 January 2020; received in revised form 26 January 2021; accepted 8 March 2021

KEYWORDS
Dynamic scheduling;
Dynamic scheduling and due date assignment;
Integer and categorical PSO;
Genetic algorithms;
Integrated process planning scheduling and due date assignment.

Abstract. Particle Swarm Optimization (PSO) has many successful applications in solving continuous optimization problems. It has been adopted to solve discrete optimization problems using different variants, such as Integer PSO (IPSO), Discrete PSO (DPSO), and Integer and Categorical PSO (ICPSO). ICPSO, a recent PSO variant, uses probability distributions instead of solution values. In this study, ICPSO algorithm is applied to solve the Dynamic Integrated Process Planning, Scheduling, and Due Date Assignment (DIPPSDDA) problem, which is a higher integration level of well-known problems including Integrated Process Planning and Scheduling (IPPS) and Scheduling With Due Date Assignment (SWDDA). Briefly, the due date assignment function is integrated into IPPS problem as the third manufacturing function in DIPPSDDA. Furthermore, DIPPSDDA implements the scheduling function in a dynamic environment where jobs arrive at the shop floor at any time. The objective of the DIPPSDDA problem is to minimize the earliness, tardiness, and length of given due dates. Since experimental results show that ICPSO converges, crossover and mutation operators used in genetic algorithms are applied to ICPSO, namely Modified ICPSO (MICPSO). Finally, experimental results indicate that the proposed MICPSO outperforms genetic algorithms, ICPSO, and modified DPSO.

1. Introduction

Process planning, scheduling, and due date assignment functions are three important functions for manufacturing environments. The first function is process planning, which prepares an engineering design for a final product by providing detailed processing instructions and then, determines the necessary resources, machines, and routes to produce a product [1]. The second function is scheduling, which is a decision-making process that considers the time-to-task assignment of scarce resources and aims to optimize one or more purposes [2]. The third function is due date assignment, which has an increasing importance with the arise of just-in-time concept in manufacturing environments, and aims to deliver products to customers at the right time. In traditional manufacturing environments, these functions are usually handled separately, which may cause inefficient schedules and due dates. Also, process plans, which are prepared independently, provide poor inputs to the scheduling, as illustrated in Figure 1.

*. *Corresponding author. Tel.: +90 264 295 7442*
E-mail addresses: cerden@subu.edu.tr (C. Erden); hidemir@sakarya.edu.tr (H.İ Demir); onurcanpolat@sakarya.edu.tr (O. Canpolat)

doi: 10.24200/sci.2021.55250.4130

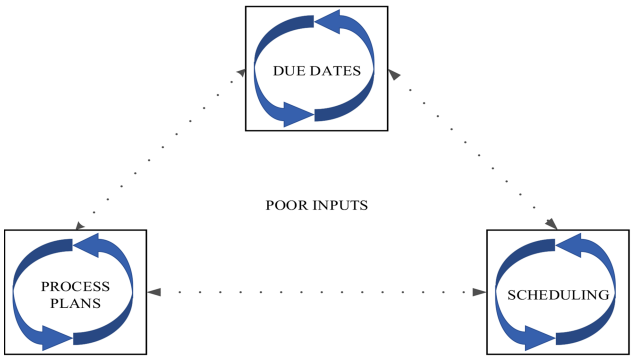

Figure 1. Unintegrated process planning, scheduling, and due date assignment.

Research on the integration of these three functions is too limited in scope. In addition, most of the related studies have determined common due dates while they overlooked the issue of customer weight. The issue of customer weight is crucial for businesses because it is undesirable to process low-priority jobs earlier than others or to assign a very long due date to very important customers.

Integrated Process Planning and Scheduling (IPPS) is a well-known study area in the literature. IPPS has the benefits of choosing an alternative route, finding solutions for urgent jobs, and balancing the load of machines. IPPS increases companies' production efficiency, helps meet demands on time, and optimizes utilization of processes and resources. In a particular case, having reviewed the studies done in the last decade, Zhang and Wong [3] used the ant colony algorithm from heuristic approaches to solve the IPPS problem in a job shop environment. Sobeyko and Mönch [4] implemented an IPPS application in a large-scale flexible job shop type production environment where different product trees and routes can be found. They addressed the total weighted delay as the objective function and provided mixed integer programming for this problem. Chaudhry [5] proposed a Genetic Algorithm (GA) based on Microsoft Excel for the IPPS problem in the job shop environment. Luo et al. [6] addressed the Multi-Objective Integrated Process Planning and Scheduling (MOIPPS) problem, in which a production system required multiple objectives to be taken into account in a more realistic decision-making process. Zhang and Wong [7] analyzed three different models of IPPS problems, including setup times. They used a customized ant colony algorithm. Petrović et al. [8] tried a new heuristic antlion optimization for IPPS problem and showed its applicability. Manupati et al. [9] discussed a mobile-agent-based approach for IPPS. Besides, some of the studies on this subject include Meenakshi Sundaram and Fu [10], Khoshnevis and Chen [11], Zhang and Mallur [12], Morad and Zalzala [13], Phanden et al. [14], Li and Gao [15], Phanden et al. [16], Li et al. [17], and Lin et al. [18].

There are also many studies in the literature on Scheduling With Due Date Assignment (SWDDA). Some of them have been carried out in single machine environments, while others have been performed in multi-machine environments. In most of the studies in the literature, it is seen that due dates are determined with respect to process times and the number of operations. Chen [19] and Gordon et al. [20] are further references on SWDDA. In recent years, Zhao et al. [21], Xiong et al. [22], Yin et al. [23], Liu et al. [24], Wang et al. [25], Yin et al. [26], and Wang et al. [27] studied this area. Zhao et al. [21] examined a single-machine scheduling and due date assignment, where the processing time of a job depends on both the start time and the position in the queue. Xiong et al. [22] discussed the problem of single-machine scheduling and due date assignment in an environment where machines are disrupted by a certain probability randomly. They aimed to minimize the optimal job sequence and costs while setting the common due date. Wang et al. [25] investigated the multi-agent single-machine SWDDA problem, each aimed at optimizing its own performance. Shabtay [28] investigated a scheduling problem in a single-machine environment where due dates are controllable with batch delivery. Yin et al. [29] carried out a single-machine SWDDA study including the costs of delivering jobs.

Increasing the integration level of these functions will help improve the scheduling performance and global production efficiency of a manufacturing system. When these three functions are integrated, there will be a strong communication between each other and better inputs will be provided for one another. Figure 2 shows the benefits of the integrated manufacturing functions. Following a review of the studies that have Integrated three functions namely Process Planning, Scheduling, and Due Date Assignment (IPPSDDA), we realized the limited scope of relevant findings in the literature. Yuan [30] proved that only the IPPS problem to minimize early and tardy jobs and batch delivery costs

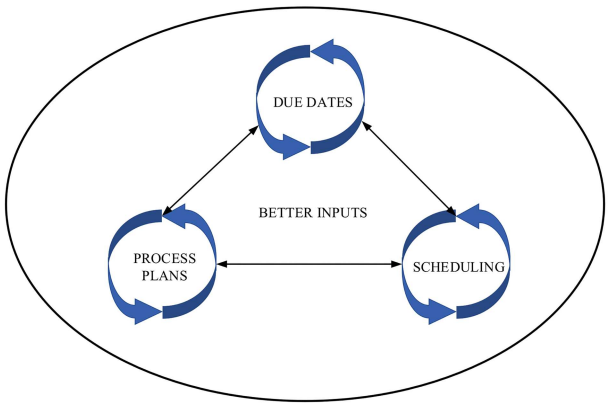

Figure 2. Integrated process planning, scheduling, and due date assignment.

was NP-hard. Thus, solving the problem of integration of the three functions will be even more complex. Demir and Taskin [31] studied IPPSDDA in their PhD thesis. Then, Ceven and Demir [32] evaluated the benefits of integrating the due date assignment with IPPS problem, and Demir and Erden [33] studied the integration of three functions by GA and ant colony optimization to minimize the sum of weighted earliness, tardiness, and due-dates of every job. Demir and Phanden [34] reviewed IPPSDDA literature in the book edited by Phanden et al. [16]. The IPPSDDA problem is a workable and remarkable research area.

This study deals with dynamic events on a Shop Floor (SF). Some internal or external dynamic events may occur on real SF. For example, a machine breakdown, an urgent job, or changes in due dates can lead to the breakdown of the previously prepared schedules or the occurrence of ineffective schedules. To overcome these problems, schedules should be made to react to dynamic events. Thus, the dynamic scheduling approach, which is prepared using real-time information and adapted to unexpected events, will provide more successful results [35]. When the studies on dynamic scheduling are examined, it is revealed that dynamic scheduling is harder to solve than static scheduling [35–37]. In this study, new job arrivals are considered as dynamic events and process planning, scheduling, and due date assignment are integrated.

There are numerous methods to solve combinational problems. One of them is Particle Swarm Optimization (PSO) which is a heuristic algorithm that is often used as it is easy to implement in solving optimization problems. PSO has been widely used in the literature, especially in IPPS problems. Ba et al. [38] developed a multimodal program using the PSO algorithm for mass production to minimize production time. Petrović et al. [39] focused on using a method based on PSO algorithm and chaos theory to investigate the field of research more extensively in IPPS problems and avoid local optima. Yu et al. [40] developed a hybrid algorithm based on GA and PSO to solve the IPPS problem, which includes two different phases, static and dynamic. Petrović et al. [41] offered a new algorithm for the optimization of flexible process plans based on the use of PSO and chaos theory. Wang et al. [42] tried to solve the multipurpose IPPS problem by using PSO. In addition to these studies, many PSO studies are included in the literature. However, almost all PSO studies have variables that are of continuous value. This indicates that the classical PSO will be insufficient to solve discrete optimization models. As a result of the studies, many PSO variations such as Integer PSO (IPSO), Discrete PSO (DPSO), Binary PSO, Veeramachaneni PSO, Angle Modulated PSO, and Discrete Estimation of Distribution Particle Swarm Optimization (DEDPSO) have been developed. One of the PSO variations that is used to eliminate this deficiency is the Integer and Categorical PSO (ICPSO), which is a type of PSO in which the values of the particles in the swarm are expressed by probability distributions. ICPSO has been used to solve Dynamic Integrated Process Planning, Scheduling, and Due Date Assignment (DIPPSDDA) problem in this study, since the due date assignment rules, the dispatching rules, and the routes of the jobs have categorical data characteristics. In most scheduling studies, the makespan is used as an objective function. However, in this study, it is aimed at minimizing the total weighted earliness, total weighted tardiness, and given weighted due dates length (E/T/D).

To the best of our knowledge, studies on DIPPSDDA are quite limited in the literature. Erden et al. [43] studied DIPPSDDA problem for the first time. In their study, GA, simulated annealing, taboo search, and combination of these algorithms were used to solve the problem. Later, Demir and Erden [33] studied DIPPSDDA problem by using ACO algorithm. In this study, ICPSO algorithm was used for the first time for solving DIPPSDDA problem. The developed algorithm was modified, and it was observed that the modified algorithm achieved better results.

The remainder of this paper is organized as follows. The DIPPSDDA problem is discussed in Section 2. PSO variants (ICPSO, MICPSO, and MDPSO) are mentioned in Section 3. The developed algorithm steps are given in Section 4. Comparative results of the algorithm are given in Section 5. Finally, the results of the study and future works are mentioned in Section 6.

2. Problem definition

DIPPSDDA problem can be considered as a contribution to the Dynamic Job Shop Scheduling Problem (DJSSP). In general, in the Job Shop Scheduling Problem (JSSP), n jobs are assigned to m machines regarding an objective function. There are some assumptions with respect to JSSP, some of which are as follows:

(a) Each machine can only perform one job at the same time;

(b) One operation can be processed on one machine;

(c) The machine does not fail when the operation starts on the machine;

(d) All jobs are ready at the beginning and there is no stochastic job arrivals or urgent jobs.

On the contrary, machines may experience breakdown or urgent jobs may arrive on the SF in the dynamic JSSP model. Since the problem of DIPPSDDA is a variant of dynamic JSSP problem, it is assumed that

jobs arrive stochastically and urgent jobs may also arrive at the SF. Besides, the alternative process plans are used to create more effective schedules.

In a DIPPSDDA model, n number of jobs with o number of operations and r number of different routes are processed through m number of machines. The processing time of an operation on a machine is generated using a normal distribution with a mean of 12 and a standard deviation of 6. Jobs are arriving at the system according to an exponential distribution with a mean of 10 and in association with a due date, which is calculated using a due date assignment rule. Operation pending machine queue is selected by machine using dispatching rules, as well. In this study, 8 different sizes of SF are produced for the problem and the data of the SF is shared in Table 1.

As mentioned earlier, in an IPPS model, scheduling is carried out considering all process plans of the jobs. This enables more efficient and balanced scheduling for the SF. Another function integrated with scheduling is the due date assignment function. Significant gains in production efficiency can be achieved with proper time of due dates.

Scheduling problems may have static or dynamic nature. In dynamic scheduling, unexpected events, such as machine break downs, new job arrivals, or changes in due dates may affect the performance of existing schedules. Unexpected events in a SF may result in the loss of optimal schedules or they may generate infeasible schedules. To handle these problems, it is important to consider the unexpected events while scheduling. Dynamic scheduling models are much closer to actual SF and they are the most difficult problems to solve among the scheduling problems. In this study, new job arrivals are studied as dynamic events.

The objective function developed for the problem involves minimizing total weighted earliness, tardiness, and due date length of every job. Tardiness and earliness are calculated as in Eqs. (1) and (2):

$$T_j = \max(c_j - d_j, 0), \tag{1}$$

$$E_j = \max(d_j - c_j, 0), \tag{2}$$

where T_j and E_j denote the tardiness and earliness times of the jth job, respectively. c_j, d_j correspond to the completion and given due date time of the jth job, respectively. If the job is completed after its given due date time, tardiness occurs. In case of tardiness, the penalty for early completion is 0, as expected. If the job is completed before its given due date time, earliness will occur and in case of the early completion time, the tardiness penalty is given as 0. Weighted due dates are penalized along with weighted earliness and tardiness and the penalty values of E/T/D are calculated as in Eqs. (3)–(6):

$$P_D = w_j \times \left(8 \times \left(\frac{d_j}{480}\right)\right), \tag{3}$$

$$P_E = w_j \times \left(5 + 4 \times \left(\frac{E_j}{480}\right)\right), \tag{4}$$

$$P_T = w_j \times \left(8 + 6 \times \left(\frac{T_j}{480}\right)\right), \tag{5}$$

$$P_{total} = P_D + P_E + P_T, \tag{6}$$

where P_D, P_E, P_T and P_{total} denote the penalties of a due date, earliness, tardiness, and total penalty, respectively. w_j is the weight of the jth job. The objective function of the model is to minimize the total penalties. Then, the final objective function ($f_{\min}$), which is a fitness value of the solution, is represented as in Eq. (7):

$$f_{\min} = \sum_{j=1}^{n} P_{total}. \tag{7}$$

Many due date assignment rules have been developed in studies [25,44–47]. It experimentally reveals which of these rules will give better results. The due date assignment rules used in this study are given in Table 2 with explanation and equations, in which a_i, p_i, o_i denote the arrival time, processing time, and the operation number of the ith job, respectively. P_{av} is the average processing time of all waiting jobs. (w_{1x}, w_{2x}) are determined proportionally inverse to the job weights and $k_x = 1, 2, 3$ and q_x : $q_1 = \frac{P_{av}}{2}$, $q_2 = P_{av}$, $q_3 = 3\frac{P_{av}}{2}$.

After determining the due dates of each job, the dispatching rule for scheduling must be determined.

Table 1. Shop floor configuration.

Numbers	SF1	SF2	SF3	SF4	SF5	SF6	SF7	SF8
Jobs	25	50	75	100	125	150	175	200
Machines	5	10	15	20	25	30	35	40
Operations	10	10	10	10	10	10	10	10
Routes	5	5	5	5	3	3	3	3
Iterations	150	150	100	100	75	75	50	50

There are many dispatching rule studies in the literature. With these rules, it is to be determined which of the waiting job is to be processed next. Dispatching rules can be divided into 4 categories: process time based, due date based, combination rules, and mixed-based rules [48]. For example, the SPT rule is a process-based rule. In SPT, the job with the shortest processing time is prioritized. Process time-based rules do not take due dates into account. EDD can be given as an example of rules that consider the due date [36]. The EDD rule prioritizes the job with the shortest due date. In the combination rules, the slack or critical rate is determined. Detailed studies on this subject can be given as follows: Adibi et al. [49], Amin and El-Bouri [50], Dominic et al. [51], Heger et al. [52], Pierreval and Mebarki [53], Qi et al. [54], Baker and Kanet [55], Raghu and Rajendran [56], and Vepsalainen and Morton [57]. As in the rules for due date assignment, it is possible to determine which of the dispatching rules gives a better solution as a result of experimental studies. The formulas of the priority index of dispatching rules are given in Table 3, in which

Table 2. Due date assignment rules.

Rule no.	Name	Explanation	Equations
0,1,2	SLK	Slack	$d_i = a_i + p_i + q_x$
3,4,5	WSLK	Weighted slack	$d_i = a_i + p_i + w_{1x} q_x$
6,7,8	TWK	Total work content	$d_i = a_i + k_x p_i$
9,10,11	WTWK	Weighted total work content	$d_i = a_i + w_{1x} k_x p_i$
12,13,14	NOPPT	Number of operations plus processing time	$d_i = a_i + p_i + 5 k_x o_i$
15,16,17	WNOPPT	The weighted number of operations plus processing time	$d_i = a_i + p_i + 5 w_{1x} k_x o_i$
18	RDM	Random-allowance due dates	$d_i = a_i + N \sim (3P_{av}, P_{av})$
19,20,21,22,23,24,25,26,27	PPW	Processing-time-plus-wait	$d_i = a_i + k_x p_i + q_x$
28,29,30,31,32,33,34,35,36	WPPW	Weighted processing-time-plus-wait	$d_i = a_i + w_{2x} k_x p_i + w_{1x} q_x$

Table 3. Dispatching rules.

Rule no.	Name	Explanation	Equations
0-1-2	WATC	Weighted Apparent Tardiness Cost	$I_i = \frac{w_i}{p_i} e^{\left(\frac{\max(slack,0)}{K\bar{p}}\right)}$
3-4-5	ATC	Apparent Tardiness Cost	$I_i = \frac{1}{p_i} e^{\left(\frac{\max(slack,0)}{K\bar{p}}\right)}$
6	WMS	Weighted Minimum Slack	$I_i = -(slack) w_i$
7	MS	Minimum Slack	$I_i = -(slack)$
8	WSPT	Weighted Shortest Process time	$I_i = \frac{w_i}{p_i}$
9	SPT	Shortest Process Time	$I_i = \frac{1}{p_i}$
10	WLPT	Weighted Longest Process Time	$I_i = \frac{p_i}{w_i}$
11	LPT	Longest process time	$I_i = p_i$
12	WSOT	Weighted Shortest Operation Time	$I_i = \frac{w_i}{p_{ij}}$
13	SOT	Shortest Operation Time	$I_i = \frac{1}{p_{ij}}$
14	WLOT	Weighted Longest Operation Time	$I_i = \frac{p_{ij}}{w_i}$
15	LOT	Longest Operation Time	$I_i = p_{ij}$
16	EDD	Earliest Due Date	$I_i = \frac{1}{d_i}$
17	WEDD	Weighted Earliest Due Date	$I_i = \frac{w_i}{d_i}$
18	ERD	Earliest Release Date	$I_i = \frac{1}{a_i}$
19	WERD	Weighted Earliest Release Date	$I_i = \frac{w_i}{a_i}$
20	SIRO	Service In Random Order	random
21	FIFO	First In First Out	$I_i = \frac{1}{a_i}$
22	LIFO	Last In First Out	$I_i = a_i$

I_i denotes the priority index of the ith job and max I_i is selected among the waiting jobs. Slack is calculated in Eq. (8):

$$slack = d_i - p_i - a_i. \tag{8}$$

3. Application of ICPSO to DIPPSDDA

The proposed ICPSO algorithm is based on the study of Strasser et al. [58]. When the algorithm is applied with Strasser's form, solution performance is stuck in local minima. To overcome this, ICPSO has been modified by the addition of crossover and mutation operators. Thus, the developed algorithm has been made more useful for integrated scheduling problems. The position vector of particles in the classical PSO structure is denoted by X_p, which is a candidate solution of particle p. To store categorical data in ICPSO, a particle representation is created using probability distributions. All dimensions in this representation create probability distributions for a solution to the problem. For this integrated problem, X_p is divided into 3 parts: $X_p = [X_{p,ddrule}, X_{p,dsprule}, X_{p,routes}]$ and each part is valid at different intervals. For the pth particle, its due date assignment rule position can be represented as $X_{p,ddrule} = [D_{p,1,ddrule}, D_{p,2,ddrule}, \cdots, D_{p,N1,ddrule}]$, where $D_{p,i,ddrule}$ represents the probability distribution for variable $X_{p,i,ddrule}$ and $N1$ denotes the due date assignment rule size. Then, every element in the particle due date position vector also consists of a set of distributions $D_{p,i,ddrule} = [d^1_{p,i,ddrule}, d^2_{p,i,ddrule}, \cdots, d^{N1}_{p,i,ddrule}]$, where $d^a_{p,i,ddrule}$ denotes the probability variable $X_{p,i,ddrule}$ that assumes value a for the pth particle. Similarly, for the pth particle, the dispatching rule position vector can be represented as:

$$X_{p,dsprule} = [D_{p,1,dsprule}, D_{p,2,dsprule}, \cdots, D_{p,N2,dsprule}],$$

where $N2$ denotes dispatching rule size. For routes dimension, we need probability distributions for all jobs in the SF, which can be represented as:

$$X_{p,routes} = \left[\left[D^{j=1}_{p,1,routes}, \cdots, D^{j=1}_{p,N3,routes}\right], \left[D^{j=2}_{p,1,routes}, \cdots, D^{j=2}_{p,N3,routes}\right], \cdots, \left[D^{j=n}_{p,1,routes}, \cdots, D^{j=n}_{p,N3,routes}\right]\right],$$

where $D^{j=k}_{p,i,routes}$ denotes the probability distribution of $X_{p,i,routes}$ for the routes of the kth job and the pth particle and $N3$ is the number of routes. The probability vector routes of each job can be represented as:

$$D^{j=k}_{p,i,routes} = \left[d^{1,j=k}_{p,i,routes}, d^{2,j=k}_{p,i,routes}, \cdots, d^{N3,j=k}_{p,i,routes}\right],$$

where $d^{a,j=k}_{p,i,routes}$ denotes the probability that variable $X^{j=k}_{p,i,routes}$ assumes value a of the pth particle.

In classical PSO, particles move using velocity vectors. Again, we have 3 dimensions for velocity vectors. A particle's due date assignment rule vector of n vectors is ϕ, which is one for each variable in the solution. The velocity vector of the due date assignment rules dimension can be represented as:

$$V_{p,ddrule} = [\phi_{p,1,ddrule}, \phi_{p,2,ddrule}, \cdots, \phi_{p,N1,ddrule}],$$

and:

$$\phi_{p,i,ddrule} = \left[\psi^1_{p,i,ddrule}, \psi^2_{p,i,ddrule}, \cdots, \psi^{N1}_{p,i,ddrule}\right],$$

where $\psi^i_{p,ddrule}$ denotes the pth particle velocity for the ith due date assignment rule. Similarly, dispatching rules can be represented as:

$$V_{p,dsprule} = [\phi_{p,1,dsprule}, \phi_{p,2,dsprule}, \cdots, \phi_{p,N2,dsprule}],$$

and:

$$\phi_{p,i,dsprule} = \left[\psi^1_{p,i,dsprule}, \psi^2_{p,i,dsprule}, \cdots, \psi^{N2}_{p,i,dsprule}\right].$$

Lastly, the route vector consists of n jobs represented as:

$$V_{p,routes} = \left[V^{j=1}_{p,routes}, V^{j=2}_{p,routes}, \cdots, V^{j=n}_{p,routes}\right],$$

and routes of the kth job represented as:

$$V^{j=k}_{p,routes} = \left[\phi^{j=k}_{p,1,routes}, \phi^{j=k}_{p,2,routes}, \cdots, \phi^{j=k}_{p,N3,routes}\right],$$

where:

$$\phi^{j=k}_{p,i,routes} = \left[\psi^{1,j=k}_{p,i,routes}, \psi^{2,j=k}_{p,i,routes}, \cdots, \psi^{N3,j=k}_{p,i,routes}\right].$$

The velocity vector in the classical PSO has been modified to make it specifically effective for this problem. The particles update position vectors at each iteration. The velocity vector of due date assignment rule, dispatching rule, and routes of each job are given in Eqs. (9)–(11):

$$V_{p,ddrule} = \omega V_{p,ddrule} + c_1 r_1 \left(p_{bestp,ddrule} + X_{p,ddrule}\right) + c_2 r_2 \left(p_{g_{best}ddrule} + X_{p,ddrule}\right), \tag{9}$$

$$V_{p,dsprule} = \omega V_{p,dsprule} + c_1 r_1 (p_{bestp,dsprule} + X_{p,dsprule}) + c_2 r_2 \left(p_{g_{best}dsprule} + X_{p,dsprule}\right), \tag{10}$$

$$V^{j=k}_{p,routes} = \omega V^{j=k}_{p,routes} + c_1 r_1 \left(p^{j=k}_{best_{routes}} + X^{j=k}_{p,routes}\right) + c_2 r_2 \left(p^{j=k}_{g\,best_{routes}} + X^{j=k}_{p,routes}\right), \tag{11}$$

where ω is the inertia rate; c_1 the cognition constant; r_1

the cognition random number within the range of [0,1]; c_2 the social constant; r_2 the social random number within the range of [0,1]; $p_{g_{best}}$ the global best particle; and p_{best} the personal best particle.

New position vectors created from velocity vectors are given in Eqs. (12)–(14):

$$X_{p,ddrule} = X_{p,ddrule} + V_{p,ddrule}, \tag{12}$$

$$X_{p,dsprule} = X_{p,dsprule} + V_{p,dsprule}, \tag{13}$$

$$X^{j=k}_{p,routes} = X^{j=k}_{p,routes} + V^{j=k}_{p,routes}, \tag{14}$$

where:

$$X_{p,routes} = \left[X^{j=1}_{p,routes}, X^{j=2}_{p,routes}, \cdots, X^{j=n}_{p,routes}\right].$$

The proposed method is presented as follows: First, an initial swarm with random probabilities is generated. Then, the particle's due date assignment, dispatching rules, and routes of each job value are determined according to probabilities. All solution values in the initial swarm are also recorded as personal best values (p_{best}) of the particles. Thus, the initial swarm is obtained. The fitness value of the particle, which has the best fitness value in the swarm, is recorded as global best values ($p_{g_{best}}$). At the same time, the best particle is saved as a particle of $p_{g_{best}}$.

At the next iterations, the probabilities for the particles of the swarm are updated using PSO velocity and position equations. The values of the due date assignment and dispatching rules and routes obtained are calculated according to the assigned probability values and the fitness for the particle. If the fitness value ($\mathcal{F}$) of the current particle ($p_{current}$) is better than the current particle's best fitness (p_{best}), p_{best} of the particle is updated and the particle is recorded as p_{best}. Then, the same process is done for all particles and it is examined whether there is a particle with better fitness than $p_{g_{best}}$. If a better fitness value is achieved, $p_{g_{best}}$ is updated and the relevant particle is saved as $p_{g_{best}}$ particle. The algorithm is iterated until the iteration size ($k_{iter,sf}$). Pseudo codes of the proposed ICPSO are given in Algorithm 1.

4. Other solution approaches

4.1. Genetic Algorithms (GA)

GA were developed by Holland [59]. GA is focused on solving computational optimization problems, inspired by the evolution of species. Iterations are performed based on the high probability of individuals with better compliance values in the GA to move to the next population. Iteration re-selection, mutation, and crossover operators are used. GA has solved many scheduling and IPPS problems via optimization. Similar studies on the problem discussed in this study are as follows: Li et al. [60], Lin et al. [61], Park and Choi [62], Pezzella et al. [63], Xia et al. [64], and Zhang et al. [65]. The proposed algorithm is working through several steps. In the initialized population step, a classical GA for solving the DIPPSDDA problem is proposed, as well. Each gene of the solution chromosome stores a due date assignment rule, a dispatching rule, and routes of jobs, respectively. For initialization of GA, a random search is applied for 20 iterations and the 10 best chromosomes are selected for the initial population.

Selection: GA selection operator selects 3 pairs of chromosomes and 4 chromosomes for crossover and mutation operations, respectively. A ranking probability method is applied to the selection operator that gives better performance in solving this problem, because the performance difference between the best and worst chromosomes is getting smaller towards the end of the iterations. Chromosome probabilities for the selection operator are fixed at every iteration and are given in Table 4. In here, we assign a higher probability of selecting those chromosomes with better fitness values;

Crossover: First, we determine the crossover point number which is based on the number of jobs, given that the number of jobs is related to the chromosome size. The number of crossover points is calculated using $ceil(gene_{size} * 0.1)$ equation. Second, we select crossover points with the selection probability of each gene. In here, we have two dominant genes including due date assignment and dispatching rule genes. Probabilities of those genes are calculated as 0.25 and

Table 4. The parameter setting of GA.

Parameters	SF1
Population size	10
Crossover probability	0.7
Mutation probability	0.3
Number of crossover points	$ceil((n+2)*0.1)$
Number of mutation points	$ceil((n+2)*0.3)$
Chromosome probabilities for selection	$[0.3, 0.2, 0.15, 0.12, 0.10, 0.07, 0.03, 0.02, 0.006, 0.004]$

Selected Shop Floor Number = sf
Determine PSO parameters N, ω, c_1, r_1, c_2, r_2, $k_{iter,sf}$
Generate N particles for initial swarm
$F_{gbest} = inf$
For $p_{current}$ in initial swarm:
    Assign X_p vector probabilities to $p_{current}$ randomly
    Assign V_p as zero-vector
    Make sample values from X_p vector
    Calculate $F_{p_{current}}$
    $F_{pbest} = F_{p_{current}}$
    if $F_{p_{current}} \leq F_{gbest}$:
        $g_{best} = F_{p_{current}}$
        $p_{g_{best}} = p_{current}$
    end if
While $generation < k_{iter,sf}$:
    $F_{total} = 0$
    For $p_{current}$ in new swarm:
        $generation += 1$
        r_1, r_2 = random number
        Determine new velocity vectors V_p
        Determine new position vectors X_p .
        Weight all probabilities to make sum to 1
        Generate sample values of X_p with probabilities of position vector.
        Run simulation and calculate new fitness for $p_{current}$
        If $F_{p_{current}} \leq F_{p_{best}}$:
            $F_{p_{best}} = F_{p_{current}}$
            $p_{best} = p_{current}$
        End if
        if $F_{p_{current}} \leq F_{g_{best}}$:
            $F_{g_{best}} = F_{p_{current}}$
            $g_{best} = p_{current}$
        End if
        $F_{total} += F_{p_{current}}$
    end for

Algorithm 1. Pseudo code of the proposed ICPSO.

0.25, respectively. Other genes, which are the selected routes of the jobs, are given a probability of 0.5, in total, because due date assignment and dispatching rule genes have a significant impact on the fitness value. If we change these genes, the performance function will be dramatically affected. On the other hand, in case the route of a job is changed, the performance function will not be affected that much. These genes should be selected in larger numbers to see which pair works well together. Therefore, these two genes have been identified as the dominant ones in the solution and a higher probability of selection has been given to those genes. Third, a multi-point crossover operator between two parent chromosomes is employed to produce two new offspring chromosomes;

Mutation: Like the crossover operator, we determine the number of mutation points using the following equation $ceil(gene_{size} * 0.3)$ in the first step. In the second step, we apply the mutation operator to the selected genes. After crossover and mutation operators, we have a new population with 20 chromosomes. To fix the population size to 10, we determined the population by selecting the best 10 chromosomes. In the last

step, we iterate the selection, crossover, and mutation operators until the iteration number is reached. The parameter setting of GA is given in Table 4.

4.2. Modified ICPSO (GA/ICPSO)

We applied different variants of PSO to find better solutions. In the algorithm of MICPSO, the mutation and crossover operators are added to classical ICPSO. ICPSO attempts to optimize the probability of obtaining a solution. However, in this study, when pure ICPSO is used, improvements become harder to achieve. That is the reason why we integrate crossover and mutation operators into the ICPSO algorithm and change a certain number of genes in the inertia, cognitive, and social parts of the algorithm. ICPSO calculates the probabilities and generates new solution values for the problem at each iteration using V and X vectors. The application of mutation operator and crossover to each particle with p_{best} and g_{best} improves the solution performance. At first, mutation operator is applied to each particle and with a probability of 0.25, each gene is replaced with other possible values. This part constitutes the moment of inertia part of the algorithm. Later, each particle is crossed over with p_{best} and each gene is replaced with a probability of 0.25 with the associated p_{best} values. This part constitutes the cognitive part of the algorithm. Finally, with a probability of 0.25, each gene is crossed over with associated g_{best} values and this constitutes the social part of the algorithm. These updates are all applied for all particles in the swarm and better results are obtained. These steps are stopped when the iteration size is reached. Pseudo codes of the proposed MICPSO are given in Algorithm 2.

4.3. Modified Discrete Particle Swarm Optimization (MDPSO)

Here, MDPSO [66] is adopted to solve the problem. In this case, the attempt was made to enhance the possible structure of ICPSO. In the study of Pan et al. [66], a probability was given for every gene of particles to be mutated or crossed over. For example, according to Pan et al. [66], first, every gene is mutated with a probability of approximately 25%. Then, each value of gene is changed into associated p_{best} value with a probability of 25%. Finally, the value of each gene is changed into associated g_{best} value with a probability of 25%.

In our MDPSO algorithm, 25% of the genes of the particle are selected randomly. Then, these genes are mutated and their values are changed into other possible values (moment of Inertia applied). After that, we select 25% of genes of the particle again randomly and their values are changed into associated p_{best} values (Cognitive part is applied). Thus, each particle and associated p_{best} values are crossed over. Finally, we select 25% of genes of the particle again and the values of these genes are changed into the associated g_{best} values (Social part is applied). Thus, the crossover is applied between every particle and g_{best} particle. This makes MDPSO more practical. Pseudo codes of the proposed MDPSO are given in Algorithm 3.

5. Experimental results

In this study, the proposed algorithms are coded in Python programming language on a personal computer featuring Intel®Core™i5-6200UCPU@2.30 GHz with 8 GB RAM. Appropriate Python packages such as NumPy [67], Matplotlib [68], Pandas [69], and Salabim [70] are utilized to analyze and solve the problem. Events such as new job arrivals, the end of an operation of the jobs, or the assignment of a job to a machine are simulated with the help of Salabim package. Besides, the job to be selected by the machine among the jobs waiting for machine queue is made by taking into consideration the dispatching rule. Thus, the objective is to find the optimal dispatching rule, due date assignment rule, and routes of each job combination. Because there is no published research data on DIPPSDDA, we generated 8 different sizes of SF and their data for this problem. The data used for this study are given as a supplementary file.

One of the outcomes of this study is the most appropriate schedules for production. The schedules obtained from the last iteration can be shown using Gantt charts. Gantt charts created for this study show the arrival of jobs. The arrival time of the jobs and the first machine to be assigned at the time of arrival are shown with the help of arrows. A Gantt chart is created for all SF, but only a Gantt chart is shown for the first SF, since it is too hard to follow charts in medium- and big-sized shops. Each job in the diagram is shown in a different color. Boxed pieces show the operations of jobs. Since there are 10 operations in every job, the jobs are shown with 10 pieces. The Gantt chart of the optimal schedule obtained by MICPSO is shown in Figure 3.

The proposed MICPSO algorithm is applied to the data and the experimental results of MICPSO are compared with the results of MDPSO, ICPSO, and GA in 8 different sizes of SF, which are illustrated in Figure 4.

According to Figure 4, MICPSO achieved the best results in all SF except SF 1, 4, and 5. From the figures, the MICPSO algorithm exhibits mostly better performance than the other algorithms. Meanwhile, the CPU time of ICPSO is better than that of other algorithms. Moreover, the best, average, and worst results for all SF can be seen in detail in Figure 5.

According to Table 5, MICPSO and MDPSO outperformed GA and pure ICPSO. On five out of the

Define PSO parameters (w, c_1, c_2, $swarm\_size$)
$F_{gbest} = inf$
Generate initial swarm $s_0 = (p_1, p_2, \ldots, p_{swarm\_size})$
    For $P_{dd}, P_{dsp}, P_{route}$ generate random probabilities:
    Define $X_{ddrule}, X_{dsprule}, X_{route}$ values according to the probabilities
    for $p_{current}$ in swarm:
        Evaluate fitness values
        Save $p_{current}$ as the p_{best} of that particle
        Save velocity vector as 0
        if $F_{p_{current}} < g_{best}$:
            $g_{best} = F_{p_{current}}$
            $p_{g_{best}} = p_{current}$
while ($i < iter\_size$):
    $s = s_i - 1$
    for $p_{current}$ in s :
        r_1, r_2 random numbers
        Update velocity vector
        for each gene in particle:
            r_3 random number
            if $r_3 < 0.25$:
                Update $p_{current}$ value by using probs values
        for each gene in particle:
            r_4 random number
            if $r_4 < 0.25$:
                Update $p_{current}$ value with associated p_{best} value
        for each gene in $p_{current}$:
            r_5 random number
            if $r_5 < 0.25$:
                Update $p_{current}$ value with associated g_{best} value
        Evaluate fitness value for $p_{current}$
        If $F_{p_{current}} \leq F_{p_{best}}$:
            $F_{p_{best}} = F_{p_{current}}$
            $p_{best} = p_{current}$
        End if
        if $F_{p_{current}} \leq F_{g_{best}}$:
            $F_{g_{best}} = F_{p_{current}}$
            $g_{best} = p_{current}$
        End if
        $F_{total} += F_{p_{current}}$
        $i += 1$
    end for

Algorithm 2. MICPSO pseudocode.

eight SF, MICPSO gave better performance; however, in the case of SF 1, 4, and 5, MDPSO achieved better performance. Given that MDPSO is quite practical, it is a promising solution technique; however, MICPSO has better performance mostly and can be recommended for resolving DIPPSDDA problems.

According to Table 6, most of the jobs experience earliness. This is because tardiness is undesired with greater fixed and variable cost terms. Fixed and variable cost parameters for earliness are 5 and 4, while fixed and variable cost parameters and coefficients for tardiness are 8 and 6. If a job is tardy rather than early, then we penalize it with an additional 3-unit fixed penalty in terms of fixed cost and variable cost

Define PSO parameters (w, c_1, c_2, $swarm\_size$)
$F_{gbest} = inf$
Give higher probabilities for the expected better rules
Generate initial swarm $s_0 = (p_1, p_2, \ldots, p_{swarm\_size})$
    Define $X_{ddrule}, X_{dsprule}, X_{route}$ values according to the probabilities
    for $p_{current}$ in swarm:
        Calculate $p_{current}$ fitness value
        Save $p_{current}$ fitness as the p_{best} of that particle
        if $F_{p_{current}} \leq F_{g_{best}}$:
            $g_{best} = F_{p_{current}}$
            $p_{g_{best}} = p_{current}$
while ($i < iter\_size$):
    $s = s_i - 1$
    for $p_{current}$ in s :
        Select 25 % of the genes for the inertia part of PSO
            Replace selected genes with other possible values
                according to the probabilities
        Select %25 of the genes for the cognitive part of PSO
            Replace selected genes with related p_{best} values
        Select %25 of the genes for the social part of PSO
            Replace selected genes with related g_{best} values
        If $F_{p_{current}} \leq F_{p_{best}}$:
            $F_{p_{best}} = F_{p_{current}}$
            $p_{best} = p_{current}$
        else:
            $p_{best} = p_{best}$
        End if
        if $F_{p_{current}} \leq F_{g_{best}}$:
            $F_{g_{best}} = F_{p_{current}}$
            $g_{best} = p_{current}$
        End if
        $F_{total} += F_{p_{current}}$
        $i += 1$
    end for

Algorithm 3. MDPSO pseudocode.

Table 5. Best algorithms and fitness values for all shop floors.

Shop floor	Best algorithm	Fitness value
SF1	MDPSO	155.93
SF2	MICPSO	285.09
SF3	MICPSO	348.69
SF4	MDPSO	470.66
SF5	MDPSO	629.16
SF6	MICPSO	682.71
SF7	MICPSO	810.50
SF8	MICPSO	964.31

coefficient becomes 6 instead of 4. The last column of the table shows the total penalty for every job and if we sum up the numbers in the last column, then we get the total penalty of all jobs which gives fitness function for this shop floor.

The best, average (Avg), and worst results of executing eight SF with all algorithms are presented in Table 7. In general, MICPSO gives better performance for eight SF with minimum best values, mostly. Further analysis of the performance of the algorithms was done using one-way Analysis of Variance (ANOVA) test to check if there is a significant difference between the results of the algorithms. Average values of the fitness functions are selected as the response values and the algorithms are assumed as factors. To perform the ANOVA analysis, we run the program ten times with different seed values on the SF 8, where the highest variability is expected. The results are given in Table 8.

Before performing ANOVA test, we need to check the normality assumption. As a result of the normality test, normality was found not satisfied as can be seen in Figure 6 ($p < 0.010$). For this reason, the non-parametric test, i.e., Kruskal-Wallis test, was

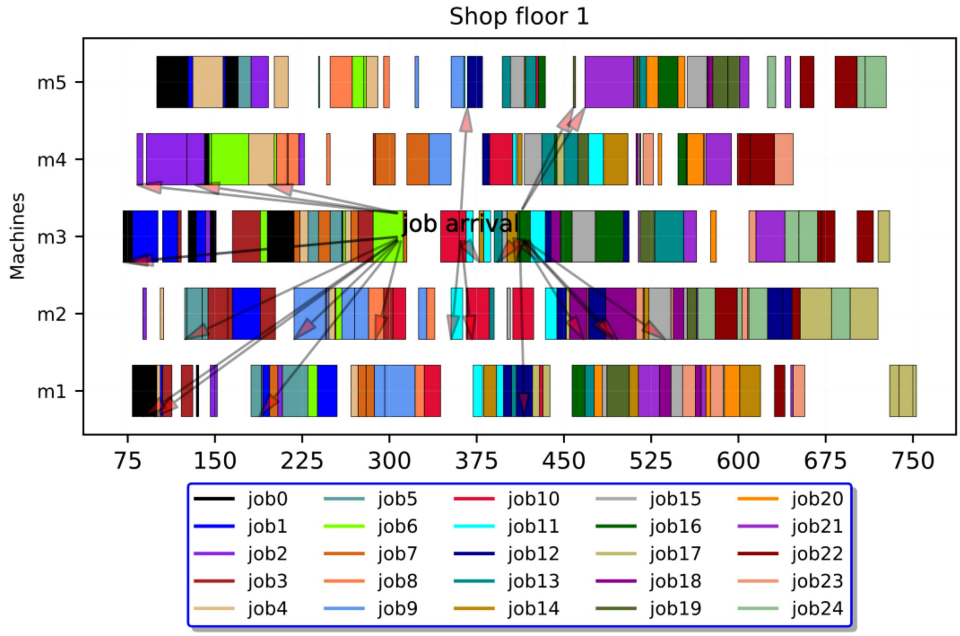

Figure 3. First shop floor Gantt chart of the optimal schedule.

Table 6. Experimental results for SF 1.

Job	Weight	Arrival time (s)	Departure time (s)	Given due date (s)	Earliness	Tardiness	Penalty earliness	Penalty tardiness	Penalty due dates	Penalty total
0	1.00	71	218	274.41	56.41	0	5.47	0	3.39	8.86
1	0.66	76	260	269.28	9.28	0	3.35	0	2.13	5.48
2	0.33	83	227	268.41	41.41	0	1.76	0	1.02	2.78
3	0.33	88	303	284.66	0	18.34	0	2.06	1.08	3.14
4	0.66	98	290	291.28	1.28	0	3.31	0	2.13	5.43
5	1.00	124	259	299.28	40.28	0	5.34	0	2.92	8.26
6	0.33	133	312	321.78	9.78	0	1.68	0	1.04	2.72
7	0.66	188	334	362.16	28.16	0	3.45	0	1.92	5.37
8	0.66	196	339	362.28	23.28	0	3.43	0	1.83	5.26
9	0.66	218	364	442.78	78.78	0	3.73	0	2.47	6.21
10	0.33	288	432	493.66	61.66	0	1.82	0	1.13	2.95
11	0.66	353	484	520.41	36.41	0	3.5	0	1.84	5.34
12	0.66	367	646	554.66	0	91.34	0	4.71	2.06	6.78
13	1.00	371	553	557.53	4.53	0	5.04	0	3.11	8.15
14	0.66	377	619	550.03	0	68.97	0	4.53	1.9	6.43
15	1.00	393	573	595.28	22.28	0	5.19	0	3.37	8.56
16	0.66	408	556	590.03	34.03	0	3.49	0	2	5.49
17	0.66	416	753	616.03	0	136.97	0	5.09	2.2	7.29
18	1.00	429	578	602.03	24.03	0	5.20	0	2.88	8.08
19	0.66	458	601	628.78	27.78	0	3.45	0	1.88	5.33
20	1.00	467	601	620.91	19.91	0	5.17	0	2.57	7.73
21	0.33	468	647	705.16	58.16	0	1.81	0	1.3	3.11
22	1.00	492	716	716.78	0.78	0	5.01	0	3.75	8.75
23	0.66	496	657	664.53	7.53	0	3.34	0	1.85	5.2
24	1.00	537	727	744.91	17.91	0	5.15	0	3.47	8.61

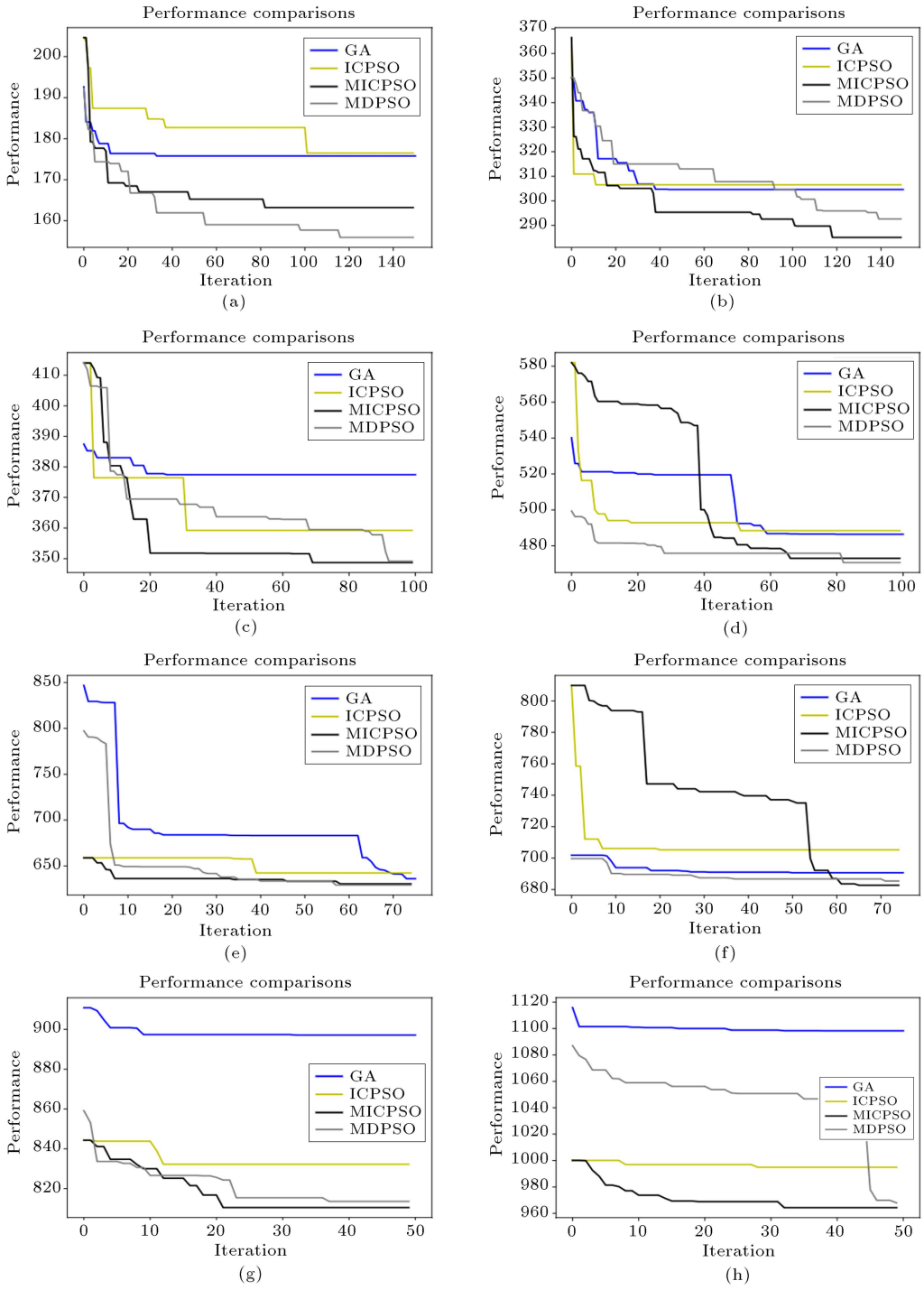

Figure 4. Comparative results of the proposed algorithms for shop floors: (a) Shop Floor 1, (b) Shop Floor 2, (c) Shop Floor 3, (d) Shop Floor 4, (e) Shop Floor 5, (f) Shop Floor 6, (g) Shop Floor 7, and (h) Shop Floor 8.

performed instead of ANOVA. Kruskal-Wallis test results show a strongly significant difference between the algorithm results because the p-value (0.007) is too close to zero, as shown in Tables 9 and 10. The mean plot in the least significant difference intervals at a confidence level of 99% is illustrated in Figure 7. These results indicate the superiority of the MICPSO algorithm over GA, MDPSO, and ICPSO.

6. Conclusion

In this study, process planning, dynamic scheduling, and due date assignment functions were integrated as a novel subject in the literature. It was assumed that the jobs would arrive at the shop floor with an exponential distribution randomly. The problem was modeled, and popular population-based Particle

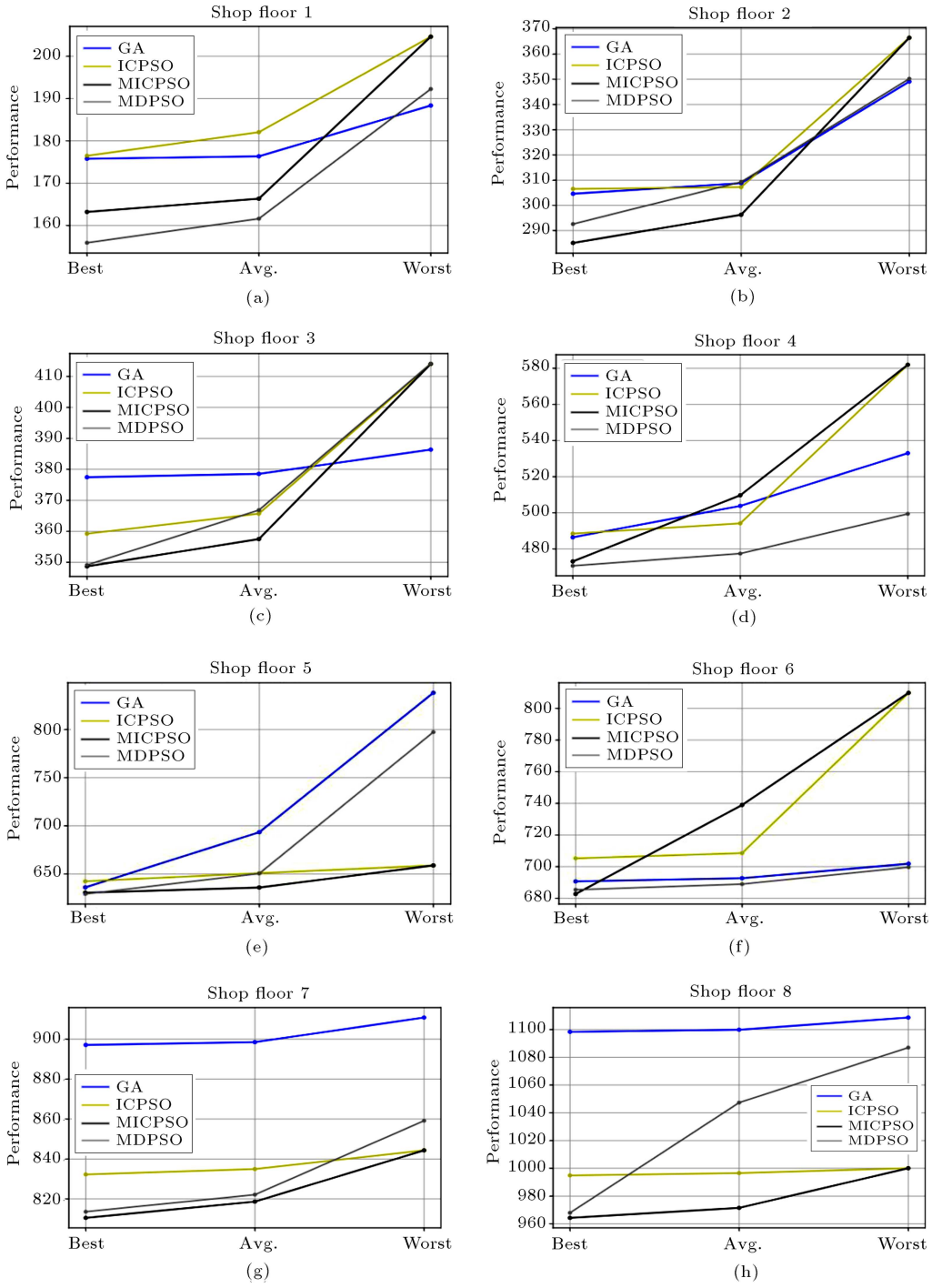

Figure 5. Comparative best, average, and worst results of the proposed algorithms for shop floors.

Swarm Optimization (PSO) and Genetic Algorithms (GA) were preferred from meta-heuristic algorithms as solution methods. Since GA solution was already been introduced in the previous studies [71,72], the structure of PSO, which was developed and modified for the solution of the problem, was mentioned more than GA in the application section of the paper. The results of the experimental studies demonstrated that MICPSO had better performance and quality and was one of the best methods in terms of both the best solution and CPU usage rates, since classical PSO usually works with continuous data. Integer and Categorical PSO (ICPSO), a variation of PSO, was utilized in this study due to the discrete and categorical nature of the problem. It has been ensured that ICPSO is modified for the problem with some improvements. Since ICPSO is a newly developed PSO variation algorithm, the implementation of the algorithm among the NP-hard

Table 7. Comparative results for shop floors.

	GA			ICPSO			MICPSO			MDPSO		
	Best	**Avg**	**Worst**	**Best**	**Avg**	**Worst**	**Best**	**Avg**	**Worst**	**Best**	**Avg**	**Worst**
SF1	175.8	176.4	188.4	176.5	182.1	204.6	163.2	166.4	204.6	**155.9**	161.6	192.3
SF2	304.6	308.8	349.1	306.5	307.2	366.5	**285.1**	296.3	366.5	292.6	309.3	350.2
SF3	377.5	378.5	386.4	359.2	365.7	414.0	**348.7**	357.5	414.0	349.1	366.9	414.1
SF4	486.4	503.8	533.0	488.5	494.1	582.0	473.1	509.7	582.0	**470.7**	477.4	499.4
SF5	636.0	693.3	838.1	642.4	650.9	658.8	630.5	635.9	658.8	**629.2**	650.5	797.3
SF6	690.7	692.7	701.9	705.2	708.6	809.8	**682.7**	739.0	809.8	685.4	689.0	699.7
SF7	897.1	898.5	910.8	832.3	834.9	844.3	**810.5**	818.6	844.3	813.6	822.1	859.1
SF8	1098.3	1099.8	1108.6	994.9	996.6	1000.1	994.9	996.6	1000.1	**967.9**	1047.3	1087.0

Table 8. Algorithm results with different seeds.

Seeds	GA			ICPSO			MICPSO			MDPSO		
	Best	**Avg**	**Worst**	**Best**	**Avg**	**Worst**	**Best**	**Avg**	**Worst**	**Best**	**Avg**	**Worst**
1	974.3	**991.8**	1180.1	988.3	**998.9**	1104.7	966.0	**982.1**	1104.7	993.8	**1086.3**	1292.2
2	976.1	**985.8**	1010.2	982.8	**1018.3**	1220.9	978.4	**993.4**	1220.9	973.5	**981.9**	992.9
3	977.0	**987.5**	1172.3	984.6	**993.3**	1017.9	965.1	**977.8**	1017.9	971.0	**977.2**	1008.7
4	992.8	**1032.2**	1068.1	983.0	**994.3**	1076.1	967.7	**984.6**	1076.1	1042.6	**1051.8**	1082.0
5	1088.7	**1104.5**	1158.4	989.0	**1013.0**	1151.0	974.2	**1029.7**	1151.0	970.1	**977.3**	1005.4
6	988.0	**1046.5**	1079.4	986.0	**987.7**	989.3	962.1	**969.6**	989.3	970.2	**979.9**	1101.2
7	1050.9	**1058.4**	1068.2	988.1	**992.1**	999.4	967.2	**974.2**	999.4	970.1	**1020.9**	1110.9
8	1088.6	**1091.4**	1106.4	989.0	**993.0**	997.9	963.5	**968.0**	997.9	967.9	**1017.0**	1233.6
9	1078.4	**1088.9**	1141.4	990.0	**1005.8**	1221.8	967.8	**1002.8**	1221.8	968.9	**976.7**	1003.3
10	1098.3	**1099.8**	1108.6	994.9	**996.6**	1000.1	964.3	**971.5**	1000.1	967.9	**1047.3**	1087.0

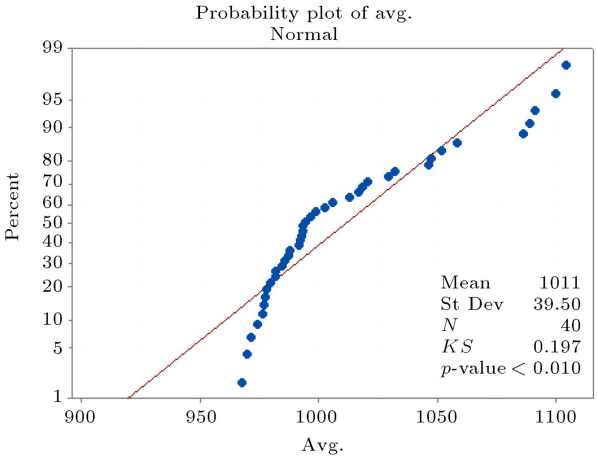

Figure 6. The normality test plot.

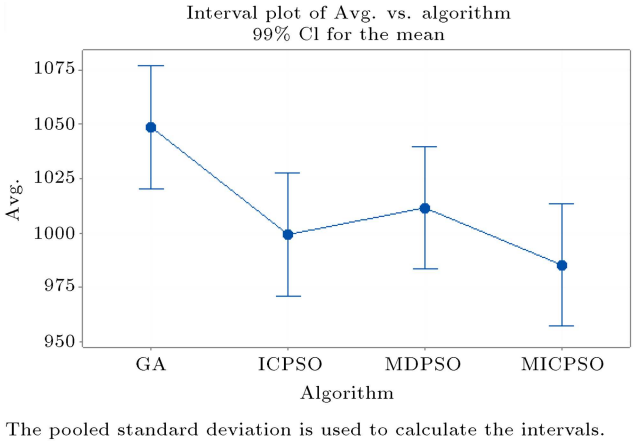

Figure 7. Interval plot of average results for the algorithms.

combinational problems is limited in the literature. Only the scheduling problem with more than 3 machines was an NP-hard optimization problem [73]. This study attempted to fulfill this gap. The developed method is called the modified PSO and serves as a new method for further studies. To sum up, these conditions indicate the original aspects of the study.

With Dynamic Integrated Process Planning, Scheduling, and Due Date Assignment (DIPPSDDA) being more efficient, effective and balanced schedules on the shop floors can be obtained, because process plans, schedules, and due dates were tried to be optimized using the alternative process plans in DIPPSDDA. A certain number of studies have touched

Table 9. Kruskal-Wallis descriptive statistics.

Algorithm	N	Median	Mean rank	Z-value
GA	10	1052.48	29.5	2.81
ICPSO	10	995.44	21.4	0.28
MDPSO	10	999.44	19.6	–0.28
MICPSO	10	979.97	11.5	–2.81
Overall	40		20.5	

Table 10. Kruskal-Wallis test results.

Null hypothesis	H_0: All medians are equal	
Alternative hypothesis	H_1: At least one median is different	
DF	***H*-value**	***P*-value**
3	11.97	0.007

on Integrated Process Planning and Scheduling (IPPS) problems and consequently, many issues have arisen so far. Therefore, there is a need for new study subjects and ideas. We presented a new study area for the researchers working on IPPS and Scheduling With Due Date Assignment (SWDDA). This issue needs further work in the future. The following future research directions are recommended:

Comparison between the other discrete methods and the proposed ICPSO is made.

- Solving the DIPPSDDA model with other successful algorithms (Artificial Bee Colony, Honeybee Colony, etc.);
- Including more objectives such as makespan to consider the DIPPSDDA problem in the form of target programming;
- Adding other dynamic events to the DIPPSDDA problem such as machine breakdowns and job cancellations;
- Integrating other production functions such as delivery manufacturing function into the DIPPSDDA problem.

References

1. Chang, T.C. and Wysk, R.A. "An introduction to automated process planning systems", *Prentice Hall Professional Technical Reference* (1984).
2. Kahlbacher, H.G. and Cheng, T.C.E. "Parallel machine scheduling to minimize costs for earliness and number of tardy jobs", *Discrete Appl. Math.*, **47**(2), pp. 139–164 (1993).
3. Zhang, S. and Wong, T.N. "Integrated process planning and scheduling: an enhanced ant colony optimization heuristic with parameter tuning", *J. Intell. Manuf.*, **29**(3), pp. 585–601 (2018).
4. Sobeyko, O. and Mönch, L. "Integrated process planning and scheduling for large-scale flexible job shops using metaheuristics", *Int. J. Prod. Res.*, **55**(2), pp. 392–409 (2017).
5. Chaudhry, I.A. "A genetic algorithm approach for process planning and scheduling in job shop environment", *Proceedings of the World Congress on Engineering 2012*, pp. 1–6 (2012).
6. Luo, G., Wen, X., Li, H., et al. "An effective multi-objective genetic algorithm based on immune principle and external archive for multi-objective integrated process planning and scheduling", *Int. J. Adv. Manuf. Technol.*, **91**(9), pp. 3145–3158 (2017).
7. Zhang, S. and Wong, T.N. "Studying the impact of sequence-dependent set-up times in integrated process planning and scheduling with E-ACO heuristic", *Int. J. Prod. Res.*, **54**(16), pp. 4815–4838 (2016).
8. Petrović, M., Petronijević, J., Mitić, M., et al. "The ant lion optimization algorithm for integrated process planning and scheduling", *Applied Mechanics and Materials*, **834**(5), pp. 187–192 (2016). DOI: 10.4028/www.scientific.net/AMM.834.187
9. Manupati, V.K., Putnik, G.D., Tiwari, M.K., et al. "Integration of process planning and scheduling using mobile-agent based approach in a networked manufacturing environment", *Comput. Ind. Eng.*, **94**, pp. 63–73 (2016).
10. Meenakshi Sundaram, R. and Fu, S. "Process planning and scheduling – A method of integration for productivity improvement", *Comput. Ind. Eng.*, **15**(1–4), pp. 296–301 (1988).
11. Khoshnevis, B. and Chen, Q.M. "Integration of process planning and scheduling functions", *J. Intell. Manuf.*, **2**(3), pp. 165–175 (1991).
12. Zhang, H.C. and Mallur, S. "An integrated model of process planning and production scheduling", *Int. J. Comput. Integr. Manuf.*, **7**(6), pp. 356–364 (1994).
13. Morad, N. and Zalzala, A. "Genetic algorithms in integrated process planning and scheduling", *J. Intell. Manuf.*, **10**(2), pp. 169–179 (1999).

14. Phanden, R.K., Jain, A., and Verma, R. "Integration of process planning and scheduling: a state-of-the-art review", *Int. J. Comput. Integr. Manuf.*, **24**(6), pp. 517–534 (2011).

15. Li, X. and Gao, L. "Effective methods for integrated process planning and scheduling", *Engineering Applications of Computational Methods*, Springer-Verlag, Berlin Heidelberg (2020).

16. Phanden, R.K., Jain, A., and Davim, J.P., Eds. *Integration of Process Planning and Scheduling: Approaches and Algorithms*, 1st Edn., CRC Press, Boca Raton (2019).

17. Li, X., Gao, L., Pan, Q., et al. "An effective hybrid genetic algorithm and variable neighborhood search for integrated process planning and scheduling in a packaging machine workshop", *IEEE Trans. Syst. Man Cybern. Syst.*, **49**(10), pp. 1933–1945 (2019).

18. Lin, C.S., Li, P.Y., Wei, J.M., et al. "Integration of process planning and scheduling for distributed flexible job shops", *Comput. Oper. Res.*, **124**, p. 105053 (2020).

19. Chen, Z.L. "Scheduling and common due date assignment with earliness-tardiness penalties and batch delivery costs", *Eur. J. Oper. Res.*, **93**(1), pp. 49–60 (1996).

20. Gordon, V., Proth, J.M., and Chu, C. "A survey of the state-of-the-art of common due date assignment and scheduling research", *Eur. J. Oper. Res.*, **139**(1), pp. 1–25 (2002).

21. Zhao, C., Hsu, C.J., Lin, W.C., et al. "Due date assignment and scheduling with time and positional dependent effects", *J. Inf. Optim. Sci.*, **39**, pp. 1613–1626 (2018). DOI: 10.1080/02522667.2017.1367515

22. Xiong, X., Wang, D., Cheng, T.C.E., et al. "Single-machine scheduling and common due date assignment with potential machine disruption", *Int. J. Prod. Res.*, **56**(3), pp. 1345–1360 (2018).

23. Yin, Y., Wang, W., Wang, D., et al. "Multi-agent single-machine scheduling and unrestricted due date assignment with a fixed machine unavailability interval", *Comput. Ind. Eng.*, **111**, pp. 202–215 (2017).

24. Liu, W., Hu, X., and Wang, X. "Single machine scheduling with slack due dates assignment", *Eng. Optim.*, **49**(4), pp. 709–717 (2017).

25. Wang, D.J., Yin, Y., Cheng, S.R., et al. "Due date assignment and scheduling on a single machine with two competing agents", *Int. J. Prod. Res.*, **54**(4), pp. 1152–1169 (2016).

26. Yin, Y., Wang, D., and Cheng, T.C.E. "Due date-related scheduling with two agents: Models and algorithms", *Uncertainty and Operations Research*, Springer Singapore (2020).

27. Wang, Y., Wang, J.Q., and Yin, Y. "Multitasking scheduling and due date assignment with deterioration effect and efficiency promotion", *Comput. Ind. Eng.*, **146**, p. 106569 (2020).

28. Shabtay, D. "Scheduling and due date assignment to minimize earliness, tardiness, holding, due date assignment and batch delivery costs", *Int. J. Prod. Econ.*, **123**(1), pp. 235–242 (2010).

29. Yin, Y., Cheng, T.C.E., Wu, C.C., et al. "Single-machine batch delivery scheduling and common due-date assignment with a rate-modifying activity", *Int. J. Prod. Res.*, **52**(19), pp. 5583–5596 (2014).

30. Yuan, J. "A note on the complexity of single-machine scheduling with a common due date, earliness-tardiness, and batch delivery costs", *Eur. J. Oper. Res.*, **94**(1), pp. 203–205 (1996).

31. Demir, H.İ. and Taskin, H. "Integrated process planning, scheduling and due-date assignment", PhD Thesis, Sakarya University (2005).

32. Ceven, E. and Demir, H.İ. "Benefits of integrating due-date assignment with process planning and scheduling", Master of Science Thesis, Sakarya University (2007).

33. Demir, H.İ. and Erden, C. "Dynamic integrated process planning, scheduling and due-date assignment using ant colony optimization", *Comput. Ind. Eng.*, **149**, p. 106799 (2020).

34. Demir, H.İ. and Phanden, R.K., *Due-Date Agreement in Integrated Process Planning and Scheduling Environment Using Common Meta-Heuristics*, CRC Press (2019).

35. Ouelhadj, D. and Petrovic, S. "A survey of dynamic scheduling in manufacturing systems", *J. Sched.*, **12**(4), pp. 417–431 (2009).

36. Ramasesh, R. "Dynamic job shop scheduling: A survey of simulation research", *Omega*, **18**(1), pp. 43–57 (1990).

37. Yin, L., Gao, L., Li, X., et al. "An improved genetic algorithm with rolling window technology for dynamic integrated process planning and scheduling problem", *2017 IEEE 21st Int. Conf. Comput. Support. Coop. Work Des. CSCWD*, pp. 414–419 (2017).

38. Ba, L., Li, Y., Yang, M., et al. "A mathematical model for multiworkshop IPPS problem in batch production", *Math. Probl. Eng.*, **2018**, p. 7948693 (2018).

39. Petrović, M., Vuković, N., Mitić, M., et al. "Integration of process planning and scheduling using chaotic particle swarm optimization algorithm", *Expert Syst. Appl.*, **64**, pp. 569–588 (2016).

40. Yu, M., Zhang, Y., Chen, K., et al. "Integration of process planning and scheduling using a hybrid GA/PSO algorithm", *Int. J. Adv. Manuf. Technol.*, **78**(1), pp. 583–592 (2015).

41. Petrović, M., Mitić, M., Vuković, N., et al. "Chaotic particle swarm optimization algorithm for flexible process planning", *Int. J. Adv. Manuf. Technol.*, **85**(9), pp. 2535–2555 (2016).

42. Wang, Y.F., Zhang, Y.F., and Fuh, J.Y.H. "A PSO-based multi-objective optimization approach to the integration of process planning and scheduling", *IEEE ICCA 2010*, pp. 614–619 (2010).

43. Erden, C., Demir, H.İ., and Kökçam, A.H. "Solving integrated process planning, dynamic scheduling, and due date assignment using metaheuristic algorithms", *Math. Probl. Eng.*, **2019**, p. 1572614 (2019).

44. Janiak, A., Janiak, W.A., Krysiak, T., et al. "A survey on scheduling problems with due windows", *Eur. J. Oper. Res.*, **242**(2), pp. 347–357 (2015).

45. Yin, Y., Wang, D.J., Wu, C.C., et al. "CON/SLK due date assignment and scheduling on a single machine with two agents", *Nav. Res. Logist. NRL*, **63**(5), pp. 416–429 (2016).

46. Browning, T.R. and Yassine, A.A. "Resource-constrained multi-project scheduling: Priority rule performance revisited", *Int. J. Prod. Econ.*, **126**(2), pp. 212–228 (2010).

47. Sha, D.Y. and Liu, C.H. "Using data mining for due date assignment in a dynamic job shop environment", *Int. J. Adv. Manuf. Technol.*, **25**(11), pp. 1164–1174 (2005).

48. Haupt, R. "A survey of priority rule-based scheduling", *Oper. Res. Spektrum*, **11**(1), pp. 3–16 (1989).

49. Adibi, M.A., Zandieh, M., and Amiri, M. "Multi-objective scheduling of dynamic job shop using variable neighborhood search", *Expert Syst. Appl.*, **37**(1), pp. 282–287 (2010).

50. Amin, G.R. and El-Bouri, A. "A minimax linear programming model for dispatching rule selection", *Comput. Ind. Eng.*, **121**, pp. 27–35 (2018).

51. Dominic, P.D.D., Kaliyamoorthy, S., and Kumar, M.S. "Efficient dispatching rules for dynamic job shop scheduling", *Int. J. Adv. Manuf. Technol.*, **24**(1), pp. 70–75 (2004).

52. Heger, J., Branke, J., Hildebrandt, T., et al. "Dynamic adjustment of dispatching rule parameters in flow shops with sequence-dependent set-up times", *Int. J. Prod. Res.*, **54**(22), pp. 6812–6824 (2016).

53. Pierreval, H. and Mebarki, N. "Dynamic scheduling selection of dispatching rules for manufacturing system", *Int. J. Prod. Res.*, **35**(6), pp. 1575–1591 (1997).

54. Qi, J.G., Burns, G.R., and Harrison, D.K. "The application of parallel multipopulation genetic algorithms to dynamic job-shop scheduling", *Int. J. Adv. Manuf. Technol.*, **16**(8), pp. 609–615 (2000).

55. Baker, K.R. and Kanet, J.J. "Job shop scheduling with modified due dates", *J. Oper. Manag.*, **4**(1), pp. 11–22 (1983).

56. Raghu, T.S. and Rajendran, C. "An efficient dynamic dispatching rule for scheduling in a job shop", *Int. J. Prod. Econ.*, **32**(3), pp. 301–313 (1993).

57. Vepsalainen, A.P.J. and Morton, T.E. "Priority rules for job shops with weighted tardiness costs", *Manag. Sci.*, **33**(8), pp. 1035–1047 (1987).

58. Strasser, S., Goodman, R., Sheppard, J., et al. "A new discrete particle swarm optimization algorithm", *Proc. 2016 Genet. Evol. Comput. Conf.-GECCO 16*, ACM Press, Denver, Colorado, USA, pp. 53–60 (2016).

59. Holland, J.H. "Genetic algorithms", *Sci. Am.*, **267**(1), pp. 66–73 (1992).

60. Li, X., Gao, L., and Shao, X. "An active learning genetic algorithm for integrated process planning and scheduling", *Expert Syst. Appl.*, **39**(8), pp. 6683–6691 (2012).

61. Lin, S., Goodman, E.D., and Punch, W.F. "A genetic algorithm approach to dynamic job shop scheduling", *Probl. Proc. Seventh Int. Conf. Genet. Algorithms*, pp. 481–489 (1997).

62. Park, B.J. and Choi, H.R. "A genetic algorithm for integration of process planning and scheduling in a job shop", *AI 2006 Adv. Artif. Intell.*, A. Sattar and B. Kang, Eds., Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 647–657 (2006).

63. Pezzella, F., Morganti, G., and Ciaschetti, G. "A genetic algorithm for the flexible job-shop scheduling problem", *Comput. Oper. Res.*, **35**(10), pp. 3202–3212 (2008).

64. Xia, H., Li, X., and Gao, L. "A hybrid genetic algorithm with variable neighborhood search for dynamic integrated process planning and scheduling", *Comput. Ind. Eng.*, **102**, pp. 99–112 (2016).

65. Zhang, L., Gao, L., and Li, X. "A hybrid genetic algorithm and tabu search for a multi-objective dynamic job shop scheduling problem", *Int. J. Prod. Res.*, **51**(12), pp. 3516–3531 (2013).

66. Pan, Q.K., Tasgetiren, M.F., and Liang, Y.C. "A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem", *Comput. Oper. Res.*, **35**(9), pp. 2807–2839 (2008).

67. Oliphant, T.E., *A Guide to NumPy*, Trelgol Publishing USA (2006).

68. Hunter, J.D. "Matplotlib: A 2D graphics environment", *Comput. Sci. Eng.*, **9**(3), p. 90 (2007).

69. McKinney, W., *Python for Data Analysis: Data Wrangling with Pandas*, NumPy, and IPython, 2 Edn., O'Reilly Media (2017).

70. Van der Ham, R. "Salabim: discrete event simulation and animation in Python", *J. Open Source Softw.*, **3**(27), p. 767 (2018).

71. Demir, H.İ., Canpolat, O., Erden, C., et al. "Process planning and scheduling with WNOPPT weighted due-date assignment where earliness, tardiness and due-dates are penalized", *J. Intell. Syst.*, p. 10 (2018).

72. Demir, H.İ. and Erden, C. "Solving process planning and weighted scheduling with WNOPPT weighted due-date assignment problem using some pure and hybrid meta-heuristics", *Sak. Univ. J. Sci.*, **21**(2), pp. 210–222 (2017).

73. Garey, M.R., Johnson, D.S., and Sethi, R. "The complexity of flowshop and jobshop scheduling", *Math. Oper. Res.*, **1**(2), pp. 117–129 (1976).

Biographies

Caner Erden is currently working as an Assistant Professor at the Faculty of Applied Sciences, Sakarya University of Applied Sciences, Sakarya, Turkey. He worked as a Research Assistant of Industrial Engineering at Sakarya University and a researcher at Sakarya University working on Artificial Intelligence Systems Application and Research between 2012 and 2020. He holds a PhD degree in Industrial Engineering from Natural Science at the Industrial Engineering Department, Sakarya University, Turkey with the thesis titled "Dynamic Integrated Process Planning, Scheduling and Due Date Assignment". His research interests include scheduling, discrete event simulation, meta-heuristic algorithms, modelling and optimization, decision-making under uncertainty, machine learning, and resource allocation and rough sets.

Halil İbrahim Demir was born in Sivas, Turkey in 1971. In 1988, he received a full scholarship and entered Bilkent University, Ankara, Turkey to study at the Industrial Engineering Department. He obtained his BSc of Science degree in Industrial Engineering in 1993. In 1994, he received a full scholarship for his graduate study in the USA from the Ministry of Education of Turkey. In 1997, he received a MSc of Science degree in Industrial Engineering from Lehigh University, Bethlehem, Pennsylvania, USA. He was then admitted to Northeastern University, Boston, Massachusetts for PhD study. He finished his PhD courses at Northeastern and completed a PhD thesis at Sakarya University, Turkey in 2005 in Industrial Engineering. He secured an academic position at Sakarya University as an Assistant Professor. His research areas of interest are production planning, scheduling, application of OR, simulation, artificial intelligence techniques, genetic algorithms, artificial neural networks, fuzzy logic, and decision-making.

Onur Canpolat is currently a Research Assistant at the Department of Industrial Engineering at Sakarya University, Turkey. He received BSc and MSc degrees in Industrial Engineering from Sakarya University, Sakarya, Turkey in 2012 and 2016, respectively. He is currently a PhD student at the same university. His areas of interest include multi-criteria decision-making, operations research, fuzzy logic, process planning, scheduling, and optimization.