Operational Research (2022) 22:5055–5105
https://doi.org/10.1007/s12351-022-00700-6

ORIGINAL PAPER

Mathematical modeling and two metaheuristic algorithms for integrated process planning and group scheduling with sequence-dependent setup time

Mohammad Reza Hosseinzadeh[1] · Mehdi Heydari[1] · Mohammad Mahdavi Mazdeh[1]

Received: 30 July 2020 / Revised: 2 February 2022 / Accepted: 10 February 2022 / Published online: 18 May 2022
© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2022

Abstract

The advancement of technology enables manufacturing companies to employ multi-function machines to increase the flexibility of a system in producing miscellaneous products in a short time. In this situation, goods can be usually produced through different process plans, and considering process planning and scheduling in an integrated framework would be essential. Furthermore, group processing is regarded to overcome the difficulty of long setup times and consequently increase the productivity of a manufacturing system. This paper deals with the integrated process planning and group scheduling problem with sequence-dependent setup time between each group of jobs. Two mixed-integer linear programming models with different approaches are presented. Moreover, two metaheuristic algorithms are proposed to solve the problems heuristically. The experiments show the high performance of the combination-based mathematical model for small-size problems as well as the proposed metaheuristic algorithms for medium-size and large-size instances.

Keywords Process planning · Group scheduling · Sequence-dependent setup time · Mathematical modeling · Genetic algorithm · Water cycle algorithm

✉ Mehdi Heydari
mheydari@iust.ac.ir

Mohammad Reza Hosseinzadeh
m_hosseinzadeh@ind.iust.ac.ir

Mohammad Mahdavi Mazdeh
mazdeh@iust.ac.ir

1 Department of Industrial Engineering, Iran University of Science and Technology, 5th Floor, Chamran Building, Narmak, Tehran, Iran

1 Introduction

In today's highly competitive business environment, producing not only high quality with minimum cost but also customized products in a short time is very important. In such a competitive market, manufacturing companies must be capable of responding to instantaneous changes rapidly. To achieve this goal, high flexibility is required in the manufacturing system. As two essential functions, process planning and scheduling have significant impacts on the flexibility and thus the efficiency of the manufacturing system. Process planning determines the selection and sequence of production operations based on product design specification as well as the required manufacturing resources, including machines, tools and tool-approach direction (TAD). In general, a process plan identifies how a product can be manufactured according to engineering design. On the other hand, scheduling allocates limited manufacturing resources to the operation in the process plans over time, subject to the precedence relations in the process plan.

These two functions are traditionally performed sequentially. Scheduling plans were generated after process plans had been determined. The disadvantages of this method were reported in the IPPS-related research (Li et al. 2010a, b). For example, in the separate process planning manufacturing systems, it is assumed that the resources have infinite capacity and are always available too. Therefore, process planners consider the most desirable resources for each job. It may cause unbalanced loads of resources and create an unexpected bottleneck. Besides, because of unpredictable shop floor disturbances such as machine breakdown, order cancellation, rush order, adding of a new machine, and the machine maintenance, and especially, the delay between the process planning and scheduling phases, the predefined optimal process plans generated in the process planning phase may even be infeasible. Some surveys show that up to 30% of process plans have to be modified due to the changing shop floor condition (Li et al. 2010d). By integrating process planning and scheduling, the load of the resources is balanced, and flow-time, work-in-process inventory, cycle time, and thus production costs are reduced (Lee and Kim 2001).

Three kinds of flexibility can be considered in the integrated process planning and scheduling (IPPS) context: operation flexibility (OF), sequencing flexibility (SF), and processing flexibility (PF). OF refers to the possibility of performing an operation on different machines and is also called routing flexibility. SF implies the availability of various permutations of manufacturing operations as long as they satisfy the precedence constraints to create a specific feature of a job, and PF means the possibility of producing the same manufacturing feature with an alternative set of operations. By taking all these flexibilities into account, although more reliable and stable plans are generated, but makes the IPPS problem much more complicated.

Furthermore, based on the concept of group technology (GT), jobs are organized into different groups according to their similarity in design, shape, material, processing operations, or other characteristics. This approach results in a reduction of setup time, throughput time, work-in-process inventory, simplification of material, and good flows and thus increases the productivity and efficiency of a manufacturing system (Lu and Logendran 2013). Production scheduling based on group technology is called "group scheduling" (Ham et al. 1985). In group scheduling (with a GT assumption), all jobs of a group are processed consecutively without interruption by jobs of a different group. Two levels of sequencing can be observed in a group scheduling problem. At the first level, a sequence of jobs within each group has to be determined, which is called a job sequence. At the second level, a group sequence is identified. These two levels should be considered simultaneously while solving group scheduling problems because of their strong interactions.

Since jobs in the same group are similar, the setup time required to change from one job to another can be ignored or included in processing times. However, a major setup time would be needed to switch from one group to another, which can be sequence-dependent or sequence-independent. The required setup for a machine is sequence-dependent if it depends on both the current and the group previously processed on that machine.

Sequence-dependent group scheduling (SDGS) is widely studied under various circumstances and objective functions in the flow shop environment. However, to the best of our knowledge, the benefits of group scheduling in other shop floors have not been considered yet, while it can be observed in a lot of production systems, especially where high technology multifunction machines are available. These machines process parts in a short time, but setup times for changing between functions are considerably high. For instance, in a compressor's filter manufacturing company, three kinds of filters are produced: air/oil separator, air filter, and oil filter. To produce each kind of filter, some operations and machines are needed, which are usually common in one or even all two other types. The filters in each category are processed together, aiming to achieve setup time reduction, whereas their process routes are different. In this case, the rolling machine is needed for all three kinds of filters, but its setup requirement is different, and it depends on which type of filters are processed previously. Furthermore, each of the aforementioned products can be manufactured in more than one way, and for some operations such as cutting and spot welding, alternative machines with different processing and setup times are available. Thus, in order to take the characteristics of this real-world instance into account, a novel process planning and group scheduling problem arises.

In this paper, the integrated process planning and group scheduling problem with sequence-dependent setup time (IPPGS-SDST) has been investigated. This problem can be considered as a flexible job shop scheduling problem (FJSP) with multiple routings and sequence-dependent setup time group processing. From another point of view, the IPPGS-SDST problem can be regarded as a sequence-dependent group scheduling in a flexible job shop environment with multiple process plans. Jobs are assigned to different groups. Jobs can be processed through various process plans, while a process plan can be selected for all jobs in the same group, and some or all operations can be performed by alternative machines. All jobs of a group are subsequently processed without being interrupted by a job from a different group, and each group requires a sequence-dependent setup time on each machine. The characteristics of both flow shop and job shop environments can be found in this problem. Within each group, all jobs must be performed in the specified order, but each group of jobs as a whole has its own route. So, it contains a flow shop problem at the job sequencing level and also a job shop problem at the group sequencing level.

Since a two-machine group scheduling problem with sequence-dependent setup time is NP-hard in the strong (Gupta and Darrow 1986; Kleinau 1993), so the IPPGS problem will be a strongly NP-hard problem which is more difficult to achieve optimal solutions by traditional accurate solution methods with reasonable time and effort. However, in order to study the nature of the problem and recognize its characteristics, two mathematical formulations are presented in this paper. One is based on the traditional point of view, and the other is network graph-based which is more applicable and efficient. Moreover, two metaheuristic algorithms are applied to find a good enough solution in an acceptable time: (1) a genetic algorithm (GA) with a new two-section representation, and (2) a hybrid water cycle algorithm (WCA) which utilize both WCA and GA operators. The main contributions of this paper can be summarized as follows:

(1) Inspired by modern manufacturing systems where the mass production of customized products is accomplished by multifunction machines, the IPPGS-SDST problem is considered for the first time.
(2) The problem is modeled mathematically in two different approaches to figure out more the characteristics of the problem.
(3) Due to group processing, the algorithms proposed in the IPPS literature cannot be applied for the considered problem. Furthermore, the group scheduling approaches are only applicable in a flow shop environment. Therefore, one of the most successful algorithms for the IPPS problem has been developed. Moreover, WCA as a novel approach is applied with an efficient discrete strategy to solve the IPPGS problem.
(4) In all solution methods of the IPPS problems, a list of process plans should be prepared as a prerequisite for running the algorithm. This time-consuming phase is eliminated using the second mathematical model approach in both presented algorithms.

The remainder of the paper is organized as follows: in Sect. 2, we review some relevant literature on the integration of process planning and scheduling as well as group scheduling separately. Representation and definition of the IPPGS problem with sequence-dependent setup time is presented in Sect. 3. Section 4 introduces the proposed MILP models and presents a comparison of the models according to the most critical complexity indices. The proposed genetic and hybrid water cycle algorithm are elaborated in Sect. 5. In Sect. 6, the managerial insight of the proposed problem is explained by an example to clarify the importance of the subject. Section 7 evaluates the proposed solution methods using generated benchmark problems. The last section is the conclusion and further research suggestions.

2 Literature review

Group scheduling, firstly presented by Hitomi and Ham (1976), has been discussed by many researchers for a long time. In addition, both sequence-dependent and sequence-independent setup times between processing groups have been considered. The first mixed-integer linear programming model for SDGS problems was investigated by Salmasi et al. (2010) with the minimization of total completion time as the criterion. Along with proposing some metaheuristic algorithms based on tabu search and ant colony optimization, they applied the mathematical model to develop a lower bounding method based on a branch-and-price algorithm. By modifying this model, Naderi and Salmasi (2012) presented two better MILP models for the permutation flow shop sequence-dependent group scheduling (FSDGS) problem. Their model was modified by Behjat and Salmasi (2017) to support the no-wait assumption. Shahvari et al. (2012) proposed a model for the flexible flow shop sequence-dependent group scheduling problem with the minimization of makespan as the criterion using the concept of slots. Their considered problem was formulated by Keshavarz and Salmasi (2013) with sequence-based variables where jobs can skip some stages. They showed that their model is much smaller than the model proposed by Shahvari et al. in terms of the number of constraints, binary variables, and continuous variables. A bi-criteria formulation of FSDGS for minimizing the weighted sum of the total weighted completion time and the total weighted tardiness was considered by Lu and Logendran (2013). A comprehensive review can be found in Neufeld et al. (2016). Nevertheless, the majority of group scheduling research has been discussed in a flow shop or flow line shop floor environment. As a result, group processing of parts has not been studied in the IPPS problem yet. In the following paragraphs, a comprehensive literature review of IPPS has been presented.

The basic idea of process planning and scheduling was first introduced by Chryssolouris et al. (1984). The first heuristic algorithm for the IPPS problem was also proposed in Khoshnevis and Chen (1991) based on opportunistic planning and used a time window to enhance its performance. The IPPS problem has widely been considered by many researchers over the past three decades. Numerous approaches have been proposed, including mathematical programming methods, meta-heuristic algorithms, and multi-agent systems for different single and multiple objectives (Jin et al. 2015). Since the single-objective case of the problem is studied in this paper, most of the related works with multi-objective functions have not been considered here.

Brandimarte and Calderini (1995) developed a bi-level hierarchical model for the IPPS problem. In the first level, a set of process plans was optimized, considering cost and load balancing as objectives. Then in the second one, for each process plan, Tabu search was used to generate the optimized scheduling plan while the makespan was taken into account. The first mixed integer programming model was presented in Kim and Egbelu (1999) for the IPPS problem with a makespan objective. They developed a MILP model for JSP in which each job has multiple process plans or operation routing. However, it is only applicable for small-size problems. Tan and Khoshnevis (2004) proposed a polynomial mixed integer programming model (PMIPM) for the IPPS problem. However, each feature consists of only one operation in their model, and sequence and process flexibility have not been considered. Kim et al. (2003) utilized a symbiotic evolutionary algorithm (SEA) with random symbiotic partner selection, which outperformed the two traditional approaches, i.e., the hierarchical approach (Brandimarte and Calderini 1995) and the co-evolutionary cooperative algorithm (Potter et al. 1994). A new approach based on the multi-agent system was developed by Wang and Shen (2003) in a distributed manufacturing environment. Notable instances of multi-agent approach in IPPS were reported in Wong et al. (2006a, b). They proposed an agent-based negotiation approach in Wong et al. (2006a) and then presented an online hybrid agent-based negotiation (oHAN) system in Wong et al. (2006b) by extending their approach with the supervisor agent and rescheduling function.

A simulated annealing-based optimization approach was utilized by Li and McMahon (2007) to facilitate the optimization of integrated process planning and scheduling. Their proposed algorithm was compared with GA and PSO considering various objective functions like makespan, job tardiness, balanced machine utilization. A multi-agent system (MAS) using data mining with a hybrid TS-SA algorithm was presented in Shukla et al. (2007). Li et al. (2008) proposed a game theory-based cooperation applying Pareto, Nash, and Stackelberg theory and GA, SA, and PSO as optimization algorithms. Some applications of PSO for IPPS can be found in Guo et al. (2009a, b). Chan et al. (2009) used a hybrid of GA, SA, and fuzzy logic controller (FLC), outperforming GA, SA, TS, and hybrid TS and SA. A preliminary implementation of an ant colony-based optimization algorithm on multi-agent systems was proposed in Leung et al. (2010). Li et al. (2010c) developed a hybrid algorithm (HA) based on an effective GA with an efficient local search scheme. Many subsequent scholars used the proposed HA to evaluate their solution methods as a comparison basis. Zhao et al. (2010) discussed the IPPS problem in holonic manufacturing systems, and the hybrid particle swarm optimization was presented to solve the utilization of all machines problem. Other noteworthy approaches developed for the IPPS problem are improved genetic algorithm (IGA) (Lihong and Shengping 2012), the imperialist competitive algorithm (ICA) (Lian et al. 2012a, b), active learning genetic algorithm (ALGA) (Li et al. 2012), and memetic algorithm (MA) (Jin et al. 2016b).

As recent notable works, Zhang and Wong (2014) proposed an enhanced ant colony optimization (E-ACO) to solve the IPPS problem. The result of the algorithm was compared with the SEA, IGA, and two-stage ACO algorithm. An object-coding genetic algorithm (OCGA) was implemented for IPPS (Zhang and Wong 2015a). In OCGA, genetic representation was based on real objects, and operation sequences were directly used as chromosomes. The performance of the proposed algorithm was evaluated against CCGA, SEA, and IGA using benchmark problems proposed in Kim et al. (2003). It shows their algorithm can reach the lower bound for 14 of 24 problems and for the other 10 problems, and the best results are very close to their lower bounds. Jin et al. (2015) proposed an MILP model for the IPPS problem as well as a hybrid honey bee mating optimization (HBMO) and variable neighborhood search (VNS) to solve the IPPS problem. A new ant colony optimization algorithm was used for the IPPS problem by Liu et al. (2016). Their comparison results showed that the proposed algorithm had better performance for 9 of 10 Kim's benchmark problems. Zhang and Wong (2016) presented a generic framework for implementing constructive meta-heuristics. ACO, as a widely-used constructive meta-heuristic, was illustrated as an example while an enhanced mapping rule was constructed. Petrović et al. (2016) considered IPPS with five flexibility types, i.e., machine, tool, tool access direction (TAD), process, and sequence flexibility, and used AND/OR network to describe the flexibilities. They applied the chaotic particle swarm optimization (cPSO) algorithm to solve this problem using ten different chaotic maps. Sobeyko and Mönch (2016) assumed that products could be manufactured with different bills of materials (BOM) and routes for the same product and considered the total weighted tardiness as the objective. They proposed variable neighborhood search and efficient heuristic based on local search to solve the problem.

In the domain of IPPS mathematical modeling, four MILP models for IPPS problems were also developed by Jin et al. (2016a). Unlike previous MILP models in the literature, which assume that all the process plans are generated in advance (Type-1), their proposed Type-2 models are suitable for network graph-based instances and are able to solve the instances expressed by an AND/OR graph while both the Manne's and the Wagner's modeling approaches are used. Other mathematical modeling approaches of IPPS problems can be found in Liu et al. (2016), Sobeyko and Mönch (2016), and Zhang et al. (2015). Regarding the high complexity of the IPPS problem, exact solution methods are rarely used. Barzanji et al. (2019) applied the Benders decomposition algorithm. They divided the IPPS problem into master-problem and sub-problem based on the decision variables. In their approach, process plan and operation-machine assignment variables are optimized in the master-problem while the sub-problem determines the scheduling variables. Since they assumed a set of process plans predetermined in advance for each job, their algorithm is not applicable to the IPPS problem in which multiple process plans are presented by network graphs.

The merits of separate consideration of setup times have been investigated widely in production scheduling problems, and some reviews have been conducted (Allahverdi 2015; Allahverdi et al. 1999, 2008). In the IPPS domain, however, most of the research has assumed setup times to be a part of processing times or can be ignored. A few papers were considered the IPPS problem with setup time issues. Imanipour (2006) presented nonlinear mixed-integer programming for the IPPS problem considering sequence-dependent setup times and proposed a hybrid of tabu search and a greedy neighborhood search to solve the problem. However, process flexibility was not considered in the model, and no comparison has been made between the results of the proposed algorithm and the mathematical model. Wan et al. (2011) applied an ant colony optimization algorithm constructed under a multi-agent system for the IPPS problem with inseparable and sequence-dependent setup time. However, the setup requirements added to IPPS problem tests to conduct the experiments are sequence-independent. In Nourali et al. (2012), a mixed-integer programming model for IPPS based on Manne's approach in the flexible assembly job shop environment with sequence-dependent setup time was developed by modifying the model of Özgüven et al. (2010). Altarazi and Yasin (2015) presented a MIP mathematical model for the IPPS problem with sequence-dependent setup time. However, alternative machines for each operation were not considered. They employed the proposed model to solve a small-size example. Four types of setup time (part loading/unloading, fixture preparation, tool/TAD changing, and transportation) have been studied in the IPPS problem by Zhang and Wong (2015b) and compared with the situation that the setup times are ignored or absorbed into processing times. An enhanced ant colony optimization (E-ACO) has been applied to solve the problem. Their results have shown that when setup times are considered separately, more accurate optimal schedules will be generated in most cases. A recent setup and transportation time consideration in the IPPS problem has been done in Ha (2019). Although three types of setup time, i.e., setup change time, tool change time, and unloading time were taken into account, but they only depended on the type of machine.

As it was mentioned before, technology advancement makes the use of SDGS inevitable in any shop floors with the aim of increasing the productivity and efficiency of a manufacturing system. This is the specific gap discussed in this paper.

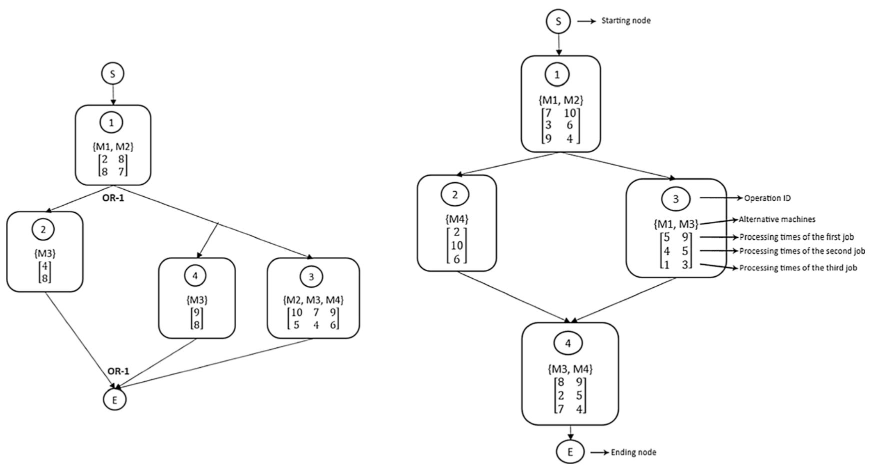
3 Problem definition

The IPPGS-SDST problem can be defined as n jobs to be accomplished by m machines. jobs are assigned to g groups according to group technology so that group s has n_s jobs. Because of the technological similarity among the jobs of a group, a set of process plans T_s or equivalently a set of operation combinations H_s is available for all jobs in the same group. The goal is to assign a process plan to all jobs in each group, select a processing machine for each operation in the process plan and find the best sequence of processing the jobs as well as the groups in order to minimize the makespan.

The alternative process plans and precedence constraints are specified by disjunctive AND/OR graphs. There are three node types in an AND/OR graph: starting node, operation node, and ending node. The starting node and the ending node are dummy nodes and, respectively, indicate the start and the completion of the manufacturing process of a job (or group of jobs). Other nodes are operation nodes, which include the operation number, the available alternative machines, and corresponding processing times. An arrow between two nodes indicates the precedence relation between them: if node A is connected to node B, operation B should be processed after operation A directly or indirectly. If there is an OR mark on a node, it means that the operations on only one OR link path should be processed. For nodes without an OR label, all operations on link paths coming from that node should be performed (Jin et al. 2016a).

lth process plan of job i contains a set of operations ($NO_{i,l}$). Similarly, hth operation combination of job i includes a set of operations ($NO_{i,h}$). Each operation is to be processed by one of the alternative machines with known processing time. Each group requires a setup time on each machine, which depends on both the selected machine and the previously processed group (sequence-dependent setup time).

An IPPGS problem is shown in Fig. 1. It consists of five jobs categorized in two groups ($G_1 = \{J_1, J_2\}$ and $G_2 = \{J_3, J_4, J_5\}$). Four multifunction machies are


S
1
{M1, M2}
$\begin{bmatrix}2 & 8\\8 & 7\end{bmatrix}$
OR-1
2
{M3}
$\begin{bmatrix}4\\8\end{bmatrix}$
4
{M3}
$\begin{bmatrix}9\\8\end{bmatrix}$
3
{M2, M3, M4}
$\begin{bmatrix}10 & 7 & 9\\5 & 4 & 6\end{bmatrix}$
OR-1
E
S
Starting node
1
{M1, M2}
$\begin{bmatrix}7 & 10\\3 & 6\\9 & 4\end{bmatrix}$
2
{M4}
$\begin{bmatrix}2\\10\\6\end{bmatrix}$
3
{M1, M3}
$\begin{bmatrix}5 & 9\\4 & 5\\1 & 3\end{bmatrix}$
Operation ID
Alternative machines
Processing times of the first job
Processing times of the second job
Processing times of the third job
4
{M3, M4}
$\begin{bmatrix}8 & 9\\2 & 5\\7 & 4\end{bmatrix}$
E
Ending node

(a) Graph A: the AND/OR graph of $G_1 = \{J_1, J_2\}$ **(b)** Graph B: the AND/OR graph of $G_2 = \{J_3, J_4, J_5\}$

Fig. 1 An IPPGS problem instance with two groups of jobs

available (M_1, M_2, M_3, M_4). The jobs in groups 1 and 2 can be processed according to graph A and graph B, respectively. According to the graphs, three process plans for group 1 and two process plans for group 2 can be recognized as follows:

$$T_1 = \begin{cases} O_1 \to O_2 \to O_2 \\ O_1 \to O_3 \to O_4, \\ O_1 \to O_4 \to O_3 \end{cases} \quad T_2 = \begin{cases} O_1 \to O_2 \to O_3 \to O_4 \\ O_1 \to O_3 \to O_2 \to O_4 \end{cases}.$$

In other words, the jobs in group 1 can be performed through two operation combinations while the jobs in group 2 have only one operation combination as demonstrated below:

$$H_1 = \{O_1, O_2\}, \{O_1, O_3, O_4\}, \quad H_2 = \{O_1, O_2, O_3, O_4\}.$$

It means the jobs in group 1 can be done by $\{O_1, O_2\}$ or $\{O_1, O_3, O_4\}$ while all operations of graph B are needed to perform the jobs of group 2. Moreover, the following assumptions are taken into account:

- A machine can execute only one operation at a given time.
- Different operations from one job cannot be processed at the same time
- The group technology assumption is considered, i.e., the jobs in the same group must be processed in succession, and the process of the jobs belonging to a group cannot be interrupted by another job belonging to another group.
- If a machine can process more than one kind of operation (multifunction machines are available), the process of one kind of operation cannot be interrupted by another kind of operation even if both operation kinds are related to the same groups or jobs (the operational constraint)

- All setup and processing times are deterministic, known in advance.
- Group setup operations are anticipatory, i.e., they can be started before the arrival of the first job in the group when the processing machine is idle.
- All jobs and machines are independent and available at time zero.
- Jobs are non-preemptive.
- Once the operation of a job is finished, it will be immediately transferred to another machine (the transport time is negligible).
- Setup times between the jobs in the same group are negligible or included in the processing time.

4 Mathematical models

Due to the combination of two manufacturing functions and the aforementioned flexibility types existing in manufacturing systems, the IPPS problem belongs to the complex, difficult NP-hard problem (Petrović et al. 2016). Although mathematical model-based accurate algorithms cannot achieve an optimal solution in a reasonable computational time, but also mathematical modeling can point out the characteristics of the problem accurately and provide a basis for evaluating other algorithms.

Considering the noticeable mathematical programming models in the research area of IPPS and group scheduling, two mathematical models with different approaches are presented for the IPPGS problem in this section: (1) the process plan-based model, and (2) the combination-based model. In both models, we consider a dummy group ($s = 0$) consisting of a dummy job ($i = 0$) with one process plan or operation combination. The dummy job is assumed to have one operation on each machine with zero processing time. Dummy jobs are used to identify the required initial setup on each machine. When a group/job processes after the dummy group/job, it is the first group /job on the related machine.

4.1 The process plan-based model

To develop the process plan-based mathematical model for IPPGS problems, we suppose a list of possible process plans (T_s) are available in advance for the jobs in each group. In order to define the aforementioned problem mathematically, the following notations, parameters, and decision variables are adopted:

1 Subscripts, notations, and sets

s, s'	Groups
i, i'	Jobs
j, j'	Operations
l, l'	Process plans
k	Machines

G	The set of groups
N	The set of jobs
M	The set of machines
G_s	The set of jobs in group s
T_s	The set of process plans for all jobs in group s
$NO_{i,l}$	The set of operations of l th process plan of job i
$O_{i,j,l}$	The j th operation of job i using the l th process plan of the job
MO_k	The set of operations can be processed on machine k
$R_{i,j,l}$	The set of available machines for $O_{i,j,l}$

1 Parameters

$t_{i,j,l,k}$	The processing time of $O_{i,j,l}$ on machine k
$g = \lvert G \rvert$	The number of groups
$n = \lvert N \rvert$	The number of jobs
$m = \lvert M \rvert$	The number of machines
$n_s = \lvert G_s \rvert$	The number of jobs in group $s \in \{1, 2, \dots, g\}$
$a_{s,s',k}$	The setup time of the group s' if it is processed immediately after group $s$on machine k
A	A very large positive number

Variables

C_{max}: Makespan

$$X_{i,l} = \begin{cases} 1, & \text{if the } l\text{th alternative process plan of job } i \text{ is selected} \\ 0, & \text{otherwise} \end{cases}$$

$$Z_{i,j,l,k} = \begin{cases} 1, & \text{if } O_{i,j,l} \text{ is processed on machine } k \\ 0, & \text{otherwise} \end{cases}$$

$$Y_{i,i',j,l} = \begin{cases} 1, & \text{if } O_{i,j,l} \text{ precedes the operation } O_{i',j,l} \\ 0, & \text{otherwise} \end{cases}$$

$$U_{s,j,l,s',j',l',k} = \begin{cases} 1, & \text{If the } j'\text{th operation of a job in group } s' \text{ using the} \\ & l'\text{th process plan is processed immediately} \\ 0 & \text{after the } j\text{th operation of a job in group } s \text{ using the} \\ & l\text{th process plan on machine } k \\ 0, & \text{otherwise} \end{cases}$$

$C_{i,j}$: The completion time of the j th operation of job i

$ST_{s,j,k}$: The starting time of the j th operation of the first job of group s on machine k

$FT_{s,j,k}$: The finishing time of the j th operation of the last job of group s on machine k

According to above definitions, the process plan-based model is formulated as follows:

$$\mathrm{Min} C_{max} \tag{1}$$

s.t.

$$\sum_{l \in T_s} X_{i,l} = 1, \quad \forall s \in G, i \in G_s \tag{2}$$

$$X_{i,l} = X_{i',l}, \quad \forall s \in G, i, i' \in G_s, i < i', l \in T_S \tag{3}$$

$$\sum_{k \in R_{i,j,l}} Z_{i,j,l,k} + \left(1 - X_{i,l}\right) = 1, \quad \forall s \in G, i \in G_s, l \in T_s, j \in NO_{i,l} \tag{4}$$

$$Z_{i,j,l,k} = Z_{i',j,l,k} \quad \forall s \in G, i, i' \in G_s, i < i', l \in T_S, j \in NO_{i,l}, k \in R_{i,j,l} \tag{5}$$

$$C_{i,j} \geq C_{i,j-1} + \sum_{k \in R_{i,j,l}} t_{i,j,l,k} Z_{i,j,l,k}, \quad \forall s \in G, i \in G_s, l \in T_s, j \in NO_{i,l}, j > 1 \tag{6}$$

$$\begin{gathered} C_{i',j} \geq C_{i,j} + t_{i',j,l,k} - A\left(3 - Y_{i,i',j,l} - Z_{i,j,l,k} - Z_{i',j,l,k}\right), \\ \forall s \in G, i, i' \in G_s, i < i', l \in T_s, j \in NO_{i,l}, k \in R_{i,j,l} \end{gathered} \tag{7}$$

$$\begin{gathered} C_{i,j} \geq C_{i',j} + t_{i,j,l,k} - A\left(2 + Y_{i,i',j,l} - Z_{i,j,l,k} - Z_{i',j,l,k}\right), \\ \forall s \in G, i, i' \in G_s, i < i', l \in T_s, j \in NO_{i,l}, k \in R_{i,j,l} \end{gathered} \tag{8}$$

$$Y_{i,i',j,l} \leq X_{i,l}, \quad \forall s \in G, i, i' \in G_s, i < i', l \in T_s, j \in NO_{i,l} \tag{9}$$

$$Y_{i,i',j,l} \leq X_{i',l}, \quad \forall s \in G, i, i' \in G_s, i < i', l \in T_s, j \in NO_{i,l} \tag{10}$$

$$C_{i,j} \geq ST_{s,j,k} + t_{i,j,l,k} - A\left(1 - Z_{i,j,l,k}\right), \forall s \in G, i \in G_s, l \in T_s, j \in NO_{i,l}, k \in R_{i,j,l} \tag{11}$$

$$FT_{s,j,k} \geq C_{i,j} - A\left(1 - Z_{i,j,l,k}\right), \quad \forall s \in G, i \in G_s, l \in T_s, j \in NO_{i,l}, k \in R_{i,j,l} \tag{12}$$

$$\begin{gathered} ST_{s',j',k} \geq FT_{s,j,k} + a_{s,s',k} - A\left(1 - U_{s,j,l,s',j',l',k}\right), \\ \forall s, s' \in G, s' \neq 0, i \in G_s, i' \in G_{s'}, l \in T_s, l' \in T_{s'}, \\ j \in NO_{i,l}, j' \in NO_{i',l'}, k \in R_{i,j,l} \cap R_{i',j',l'} \end{gathered} \tag{13}$$

$$\sum_{s\in G}\sum_{i\in G_s}\sum_{l\in T_s}\sum_{\substack{j\in NO_{i,l}\\ O_{i,j,l}\in MO_k}}\left(\frac{U_{s,j,l,s',j',l',k}}{n_s}\right)=Z_{i',j',l',k},$$
$$\forall s'\in G, s'\neq 0, i'\in G_{s'}, l'\in T_{s'}, j'\in NO_{i',l'}, k\in R_{i',j',l'} \tag{14}$$

$$\sum_{\substack{s'\in G\\ s'\neq 0}}\sum_{i'\in G_{s'}}\sum_{l'\in T_{s'}}\sum_{\substack{j'\in NO_{i',l'}\\ O_{i',j',l'}\in MO_k}}\left(\frac{U_{s,j,l,s',j',l',k}}{n_{s'}}\right)\leq Z_{i,j,l,k},$$
$$\forall s\in G, i\in G_s, l\in T_s, j\in NO_{i,l}, k\in R_{i,j,l} \tag{15}$$

$$C_{max}\geq C_{i,j},\quad \forall s\in G, i\in G_s, l\in T_s, j\in NO_{i,l} \tag{16}$$

$$C_{i,j}, ST_{s,j,k}, FT_{s,j,k}\geq 0 \tag{17}$$

$$X_{i,l}, Z_{i,j,l,k}, Y_{i,i',j,l}, U_{s,j,l,s',j',l',k}\in\{0,1\} \tag{18}$$

The makespan minimization is expressed by Eq. (1) as the objective function. Constraint set (2) ensures that only one process plan is selected for each job. The same process plan should be chosen for all jobs in a group. Constraint set (3) is incorporated into the model in order to support this fact. Constraint set (4) is incorporated into the model to make sure that each operation of a selected process plan for a job is assigned to only one machine. Constraint set (5) implies the jth operation of all jobs in the same group must be processed on the same machine, consequently (group technology assumption). Since a job cannot be processed on two different machines at the same time, the completion time of an operation should be greater than the completion time of the previous operation plus its processing time. Constraint set (6) is incorporated into the model in order to support this fact. Constraint sets (7) and (8) are included in the model to schedule different operations on the same machine. Operation $O_{i,j,l}$ can precede operation $O_{i',j,l}$ if process plan l is selected for job i. In addition, operation $O_{i',j,l}$ can be processed after operation $O_{i,j,l}$ if process plan l is selected for job i'. Constraint sets (9) and (10) express these statements, respectively. Constraint set (11) ensures that each job should start after starting times of the group it belongs to. Constraint set (12) ensures that the completion time of a group on each machine is greater than or equal to the completion time of its jobs on that machine. The start time of processing a job belonging to a specific group (say group s') on a machine (say machine k) is greater than the completion time of the immediately previous group (say group s) process on the machine plus the related sequence-dependent setup time (i.e., $a_{s,s',k}$). Constraint set (13) is incorporated into the model for this reason. Constraint set (14) supports the statement that if group s' is processed on machine k, then only one group precedes immediately group s' or group s' is the first group on the machine. Constraint set (15) ensures that at most one group follows immediately group s on machine k. Constraint set (16) is used to capture the value of the makespan. Constraint sets (17) and (18) define the decision variables.

4.2 The combination-based model

As mentioned above, Type-2 models for IPPS problems were developed by Jin et al. (2016a) based on the term "combination," which defines as the different operation combinations that a job can perform through them. As they discussed, the main motive to develop Type-2 models is that it is sometimes so difficult to identify and generate all the process plans for a job according to its AND/OR graph.

As an example, more than 100 process plans can be identified for a job with the relatively simple AND/OR graph in Fig. 2. However, there is only one combination of operations for the job. Clearly, it is time-consuming and sometimes impossible to list all process plans of jobs with more complicated graphs. Based on this weakness, the combination-based model emerged.

In this section, a mathematical model is presented by applying the combination concept and the results of the Jin et al. research using the Manne's approach. Notations, parameters, and decision variables of the model are as follows:

1 Subscripts, notations, and sets

s, s'	Groups
i, i'	Jobs
j, j'	Operations
h, h'	Combinations
k	Machines
G	The set of groups
N	The set of jobs

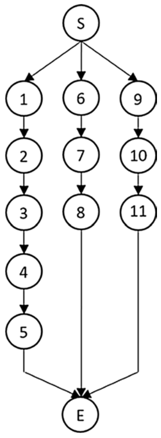


Fig. 2 An AND/OR graph with numerous process plans

M	The set of machines
N_i	The set of operations of job i
G_s	The set of jobs in group s
H_s	The set of operation combinations for all jobs in group s
$NO_{i,h}$	The set of operations of h th combination of job i
$O_{i,j}$	The j th operation of job i
OH_h	The set of operations ($O_{i,j}$) including combination h
MO_k	The set of operations ($O_{i,j}$) can be processed on machine k
$R_{i,j}$	The set of available machines for $O_{i,j}$

Parameters

$t_{i,j,k}$: the processing time of $O_{i,j,l}$ on machine k

$g = |G|$: the number of groups

$n = |N|$: the number of jobs

$m = |M|$: the number of machines

$n_s = |G_s|$: the number of jobs in group $s \in \{1, 2, \ldots, g\}$

$$V_{i,j,j'} = \begin{cases} 1, & \text{if the } O_{i,j} \text{ is an immediate predecessor of } O_{i,j'} \\ & \text{according to the network graph of job } i \\ 0, & \text{otherwise} \end{cases}$$

$$Q_{i,j,j'} = \begin{cases} 10, & \text{if the } O_{i,j} \text{ should be processed before } O_{i,j'} \\ & \text{directly or indirectly according to the network graph of job } i \\ 0, & \text{otherwise} \end{cases}$$

$a_{s,s',k}$: the setup time of the group s' if it is processed immediately after group s on machine k

A: a very large positive number

Variables

C_{max}: Makespan

$$X_{i,h} = \begin{cases} 1, & \text{if the } h \text{ th alternative combination of} \\ & \text{operations is selected to accomplish job } i \\ 0, & \text{otherwise} \end{cases}$$

$$Z_{i,j,h,k} = \begin{cases} 1, & \text{if the } h \text{ th combination is selected for job } i \text{ and } O_{i,j} \\ & \text{is processed on machine } k \\ 0, & \text{otherwise} \end{cases}$$

$$Y_{i,j,j'} = \begin{cases} 1, & \text{if } O_{i,j} \text{ precedes the operation } O_{i,j'} \\ 0, & \text{otherwise} \end{cases}$$

$$U_{i,i',j} = \begin{cases} 1, & \text{if } O_{i,j} \text{ precedes the operation } O_{i',j}, i, i' \in G_s \\ 0, & \text{otherwise} \end{cases}$$

$$W_{s,j,s',j',k} = \begin{cases} 10, & \text{If the } j'\text{th operation of a job in group } s' \text{ is processed immediately} \\ & \text{after the } j\text{th operation of a job in group } s \text{ on machine } k \\ 0, & \text{otherwise} \end{cases}$$

$C_{i,j}$: the completion time of the j th operation of job i

$ST_{s,j,k}$: the starting time of the j th operation of the first job of group s on machine k

$FT_{s,j,k}$: the finishing time of the j th operation of the last job of group s on machine k

With regard to above definitions, the combination-based model of the IPPGS is formulated as follows:

$$\text{Min} C_{max} \tag{19}$$

s.t.

$$\sum_{h \in H_s} X_{i,h} = 1, \quad \forall s \in G, i \in G_s \tag{20}$$

$$X_{i,h} = X_{i',h}, \quad \forall s \in G, i, i' \in G_s, i < i', h \in H_s \tag{21}$$

$$\sum_{k \in R_{i,j}} Z_{i,j,h,k} + \left(1 - X_{i,h}\right) = 1, \quad \forall s \in G, i \in G_s, h \in H_s, j \in NO_{i,h} \tag{22}$$

$$Z_{i,j,h,k} = Z_{i',j,h,k} \quad \forall s \in G, i, i' \in G_s, i < i', h \in H_s, j \in NO_{i,h}, k \in R_{i,j} \tag{23}$$

$$\begin{gathered} C_{i,j'} \geq C_{i,j} + \sum_{k \in R_{i,j'}} t_{i,j',k} Z_{i,j',h,k} - A\left(1 - X_{i,h}\right), \\ \forall s \in G, i \in G_s, h \in H_s, j, j' \in NO_{i,h}, j \neq j', V_{i,j,j'} = 1 \end{gathered} \tag{24}$$

$$\begin{gathered} Y_{i,j,j'} + Y_{i,j',j} = 1, \\ \forall s \in G - \{0\}.i \in G_s, h \in H_s, j, j' \in NO_{i,h}, j < j', Q_{i,j,j'} + Q_{i,j',j} = 0, \end{gathered} \tag{25}$$

$$Y_{i,j,j'} = Y_{i',j,j'} \quad \forall s \in G - \{0\}.i.i' \in G_s, i < i'.h \in H_s, j, j' \in NO_{i,h}, j \neq j', Q_{i,j,j'} + Q_{i,j',j} = 0 \tag{26}$$

$$\begin{gathered} C_{i,j'} \geq C_{i,j} + \sum_{k \in R_{i,j'}} t_{i,j',k} Z_{i,j',h,k} - A\left(1 - Y_{i,j,j'}\right) \\ \forall s \in G - \{0\}.i \in G_s, h \in H_s, j, j' \in NO_{i,h}, j \neq j', Q_{i,j,j'} + Q_{i,j',j} = 0 \end{gathered} \tag{27}$$

$$\begin{gathered} C_{i',j} \geq C_{i,j} + t_{i',j,k} - A\left(3 - U_{i,i',j} - Z_{i,j,h,k} - Z_{i',j,h,k}\right), \\ \forall s \in G, i, i' \in G_s, i < i', h \in H_s, j \in NO_{i,h}, k \in R_{i,j} \end{gathered} \tag{28}$$

$$C_{i,j} \geq C_{i',j} + t_{i,j,k} - A\left(2 + U_{i,i',j} - Z_{i,j,h,k} - Z_{i',j,h,k}\right), \quad \forall s \in G, i, i' \in G_s, i < i', h \in H_s, j \in NO_{i,h}, k \in R_{i,j} \tag{29}$$

$$C_{i,j} \geq ST_{s,j,k} + t_{i,j,k} - A\left(1 - Z_{i,j,h,k}\right), \quad \forall s \in G, i \in G_s, h \in H_s, j \in NO_{i,h}, k \in R_{i,j} \tag{30}$$

$$FT_{s,j,k} \geq C_{i,j} - A\left(1 - Z_{i,j,h,k}\right), \quad \forall s \in G, i \in G_s, h \in H_s, j \in NO_{i,h}, k \in R_{i,j} \tag{31}$$

$$ST_{s',j',k} \geq F_{s,j,k} + a_{s,s',k} - A\left(1 - W_{s,j,s',j',k}\right), \quad \forall s, s' \in G, s' \neq 0, i \in G_s, i' \in G_{s'}, h \in H_s, h' \in H_{s'}, j \in NO_{i,h}, j' \in NO_{i',h'}, k \in R_{i,j} \cap R_{i',j'} \tag{32}$$

$$\sum_{s \in G} \sum_{i \in G_s} \sum_{\substack{j \in N_i \\ O_{i,j} \in MO_k}} \left(\frac{W_{s,j,s',j',k}}{n_s} \right) = \sum_{\substack{h' \in H_{s'} \\ O_{i',j'} \in OH_{h'}}} Z_{i',j',h',k}, \quad \forall s' \in G, s' \neq 0, i' \in G_{s'}, j' \in N_{i'}, k \in R_{i',j'} \tag{33}$$

$$\sum_{\substack{s' \in G \\ s' \neq 0}} \sum_{i' \in G_{s'}} \sum_{\substack{j' \in N_{i'} \\ O_{i',j'} \in MO_k}} \left(\frac{W_{s,j,s',j',k}}{n_{s'}} \right) \leq \sum_{\substack{h \in H_s \\ O_{i,j} \in OH_h}} Z_{i,j,h,k}, \quad \forall s \in G, i \in G_s, j \in N_i, k \in R_{i,j} \tag{34}$$

$$C_{max} \geq C_{i,j}, \quad \forall s \in G, i \in G_s, h \in H_s, j \in NO_{i,h} \tag{35}$$

$$C_{i,j}, ST_{s,j,k}, FT_{s,j,k} \geq 0 \tag{36}$$

$$X_{i,l}, Z_{i,j,h,k}, Y_{i,j,j'}, U_{i,i',j}.W_{s,j,s',j',k} \in \{0, 1\} \tag{37}$$

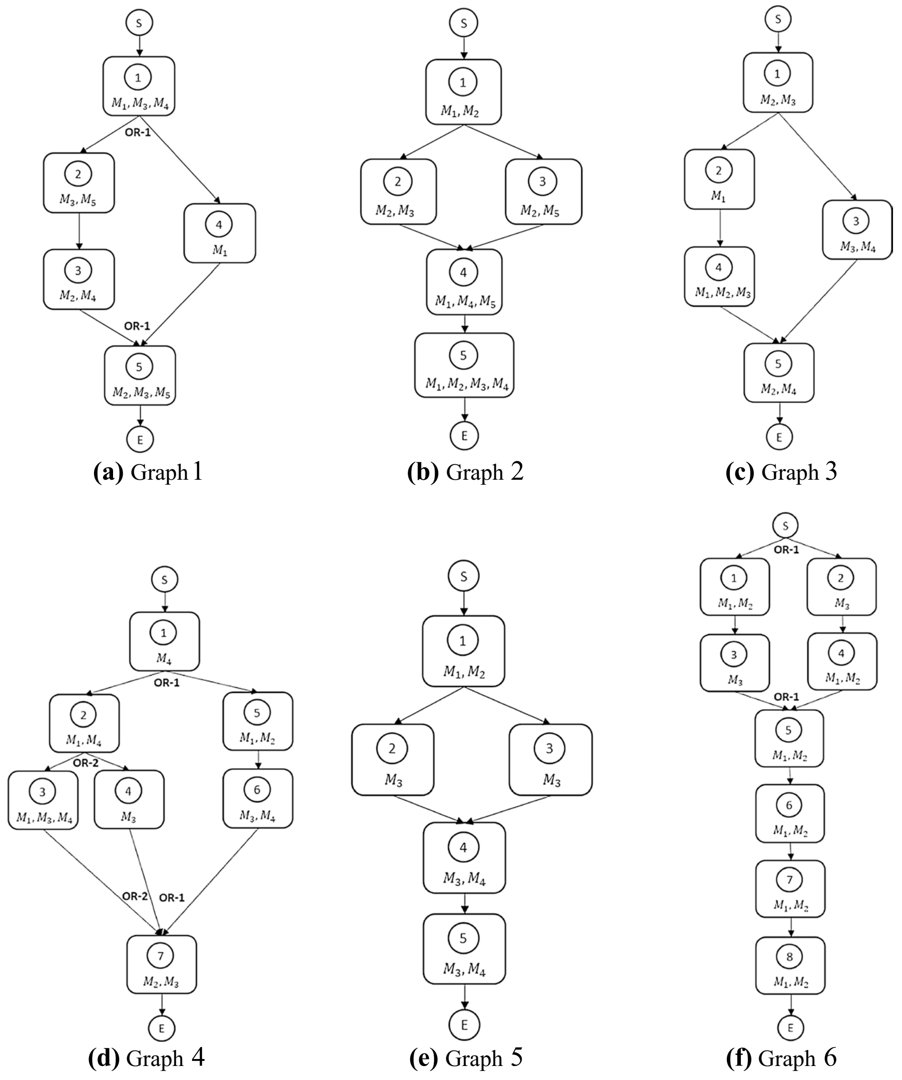
The makespan minimization is expressed by Eq. (19) as an objective function. Constraint set (20) ensures that only one combination is selected for each job. The same combination of operations should be chosen for all jobs in a group. Constraint set (21) is incorporated into the model in order to support this fact. Constraint set (22) is incorporated into the model to make sure that each operation is assigned to only one machine. Constraint set (23) implies the j th operation of all jobs in the same group must be processed on the same machine, consequently (group technology assumption). Constraint set (24) is incorporated into the model to compute the completion time of two operations of a job that have immediate precedence in the related network graph. Constraint sets (25–27) determine the sequence of the operations that have no explicit precedence relationship with each other in the network. Among them, constraint set (27) ensures that all jobs in the same group should select the same process plan and hence should traverse the same route. Constraint sets (28) and (29) are incorporated into the model to schedule different operations on the same machine. Constraint set (30) ensures that each job should start after starting times of the group it belongs to. Constraint set (31) ensures that the completion time of a group on each machine is greater than or equal to the completion time of its jobs on that machine. The start time of processing a job belonging to a specific group (say group s') on a machine (say machine k) is greater than the completion time of the immediately previous group (say group s) process on the machine plus the related sequence-dependent setup time (i.e., $a_{s,s',k}$). Constraint set (32) is incorporated into the model for this reason. Constraint set (33) supports the statement that if group s' is processed on machine k, then only one group precedes immediately group s' or group s' is the first group on the machine. Constraint set (34) ensures that at most one group follows immediately group s on machine k. Constraint set (35) is used to capture the value of the makespan. Constraint sets (36) and (37) define the decision variables.

4.3 Comparison of the models

In this section, two proposed models are compared with each other according to the number of constraints (NC), the number of binary variables (NBV), and the number of continuous variables (NCV). As mentioned in the previous section, in many cases, only the graph of jobs confirms the superiority of the combination-based model, especially when some jobs have high sequence flexibility. In this situation, it is tedious to prepare a full list of process plans as a prerequisite for the process plan-based model. Hence, to compare the models, six low sequence flexibility graphs—taken from Zhang and Wong (2014), Zhang and Wong (2016), and Zhang and Wong (2015a) and presented in Fig. 3—are applied to generate five IPPGS problems with five machines. These problems are provided in different sizes, i.e., the different number of groups, jobs, and consequently, the total numbers of operations (TNO). Each problem has a total n jobs in g groups which n_1 jobs are in the first group with process plans depicted in AND/OR graph 1 (G_1), n_2 jobs are in the second group with process plans depicted in AND/OR graph 2 (G_2), and n_g jobs are in the g th group with process plans depicted in AND/OR graph g (G_g).

Table 1 provides the NC of the models for different problem sizes The combination-based model consists of less NC than the process plan-based one in all sizes. Table 2 includes the same problems to compare the NBV of the proposed models. The combination-based model has significantly less the NBV than the other model. In most sizes, the NBV of the process plan-based model is more than twice the NBV of the combination-based model. Finally, Table 3 compares the model based on the NCV. The process plan-based model formulates the problem with fewer continuous variables. However, the difference between the two models according to the NCV is less than 7%.

The comparison of the two models shows considerable superiority in the NC and the NBV and, on the other hand, low inferiority in the NCV of the combination-based model. Since the type of both models above is mixed-integer linear programming, the NCV has a less substantial effect on the model performance. As a result,



(a) Graph 1 **(b)** Graph 2 **(c)** Graph 3

(d) Graph 4 **(e)** Graph 5 **(f)** Graph 6

Fig. 3 AND/OR graphs used for comparison of the models

the combination-based model is more efficient. So, it is used to provide the basis for evaluating the following proposed solution algorithms. However, in the problem instances where all jobs have low sequence flexibility, the process plan-based model can be more efficient because the limited process plans are simply recognizable, and also, it is not required to build $V_{i,j,j'}$ and $Q_{i,j,j'}$ matrixes.

Table 1 Number of constraints (NC) comparison of the models

No	Problem size			TNO	The NC of the models	
	n	g	$G_1(n_1)-G_2(n_2)-\cdots-G_g(n_g)$		PP-based	C-based
1	7	2	$G_1(3)-G_2(4)$	35	6764	3226
2	11	3	$G_1(5)-G_2(3)-G_4(3)$	68	13,374	9390
3	21	4	$G_1(5)-G_2(3)-G_3(6)-G_4(7)$	119	62,881	29,475
4	30	5	$G_1(4)-G_2(6)-G_4(5)-G_5(8)-G_6(7)$	181	87,876	58,890
5	44	6	$G_1(6)-G_2(5)-G_3(7)-G_4(8)-G_5(8)-G_6(10)$	266	238,147	126,140

Table 2 Number of binary variables (NBV) comparison of the models

No	Problem size			TNO	The NBV of the models	
	n	g	$G_1(n_1)-G_2(n_2)-\cdots-G_g(n_g)$		PP-based	C-based
1	7	2	$G_1(3)-G_2(4)$	35	680	314
2	11	3	$G_1(5)-G_2(3)-G_4(3)$	68	1321	633
3	21	4	$G_1(5)-G_2(3)-G_3(6)-G_4(7)$	119	3184	1222
4	30	5	$G_1(4)-G_2(6)-G_4(5)-G_5(8)-G_6(7)$	181	3706	1887
5	44	6	$G_1(6)-G_2(5)-G_3(7)-G_4(8)-G_5(8)-G_6(10)$	266	6913	2917

Table 3 Number of continuous variables (NCV) comparison of the models

No	Problem size			TNO	The NCV of the models	
	n	g	$G_1(n_1)-G_2(n_2)-\cdots-G_g(n_g)$		PP-based	C-based
1	7	2	$G_1(3)-G_2(4)$	35	105	100
2	11	3	$G_1(5)-G_2(3)-G_4(3)$	68	138	152
3	21	4	$G_1(5)-G_2(3)-G_3(6)-G_4(7)$	119	214	230
4	30	5	$G_1(4)-G_2(6)-G_4(5)-G_5(8)-G_6(7)$	181	272	316
5	44	6	$G_1(6)-G_2(5)-G_3(7)-G_4(8)-G_5(8)-G_6(10)$	266	381	421

5 Solution methods

As mentioned in Sect. 1, the high level of complexity and huge solution space of the IPPGS problem makes the use of metaheuristic algorithms inevitable because it is almost impossible to achieve an optimal solution by accurate algorithms in reasonable times. In this section, two efficient metaheuristic algorithms are developed. The first one is a genetic algorithm with a new two-section and object-coding representation, and the other one is a hybrid of the relatively novel water cycle algorithm and genetic algorithm. As it is mentioned, in all optimization approaches of the IPPS domain, a list of process plans should be prepared at first, which is a tedious and time-consuming initial phase of these solution methods, mainly when the problem includes at least one graph with a high level of sequence flexibility. To settle these difficulties, the same approach in mathematical modeling, that is, the combination-based approach, is used for both metaheuristic optimization algorithms, too.

5.1 Genetic algorithm

Genetic algorithms were extensively applied with various characteristics in the IPPS domain. In some early studies, the basic procedure of the algorithm was used (Morad and Zalzala 1999). A few researchers combined it with other optimization methods to construct a hybrid algorithm for IPPS problem (Amin-Naseri and Afshari 2012; Li et al. 2019a, b; Uslu et al. 2018), while in many cases, an improved version of GA was applied (Li et al. 2012; Lihong and Shengping 2012; Luo et al. 2017; Mohapatra et al. 2015; Shao et al. 2009; Zhang et al. 2014). Also, since chromosome representation is the most important factor in the performance of a GA, some researchers focus on presenting efficient representation (Lee and Ha 2019).

In this section, a classical genetic algorithm using an object-coding representation presented by Zhang and Wong (2015a) is adopted with some modifications for the IPPGS problem. The most applicable procedures of GA, where each iteration consists mainly of the population generation, selection of parents, genetic operations, and population evaluation and truncation, were adopted to solve the problem.

5.1.1 Genetic representations

In order to gather all information included in AND/OR graphs of the IPPGS problem in an encoded chromosome and use it during the optimization process, a two-section representation is applied. Section 1 (numerical section) is a numerical representation that shows the operation combination used by each group, respectively, while Sect. 2 (object section) is based on real objects and indicates the sequence of operations based on the precedence relationship between them. The first section has g (number of groups) genes, but the number of genes for the second section depends on the information provided in the first section.$O_{i,j}^{k}$ in each gene of the chromosome represents j th operation of job i which performs on machine k from its set of alternative machines.

A chromosome is presented in Fig. 4 representing a problem instance with five jobs in two groups. Jobs 1 and 2 are in group 1, and jobs 3, 4, and 5 are in group 2. Two groups can be processed on five machines, according to respectively graph 1 and graph 2 illustrated in Fig. 3a and b. Due to group processing, the same achine should be selected for the same operations of all jobs in a group.

In the above chromosome, the jobs in group 1 use their operations in combination 2 (O_1, O_4, O_5), and the jobs in group 2 are planning to perform through their operations in only one available combination (O_1, O_2, O_3, O_4, O_5). The object-coding section of the chromosome indicates the order of operation to enter the corresponding schedule. It should be noted that all the same operations of the jobs in the same group must be processed consecutively according to group technology assumption.

5.1.2 Initial population, individual evaluation, and selection of parents

With regard to the encoding scheme, the initial population, including N_{pop} individuals are generated randomly, while precedence constraints are satisfied. On the other hand, the same machine is selected for the same operations of all jobs in a group. For example, in Fig. 4, machine 2 is assigned to all operations $O_{3,1}, O_{4,1}$, and $O_{5,1}$. Individual evaluation is based on the optimization criteria, which is to minimize the makespan in this paper. The popular tournament selection method is adopted in this algorithm to select a pair of parent individuals.

5.1.3 Genetic operators

Due to three kinds of flexibilities, i.e., process flexibility, sequence flexibility, and operation flexibility, and on the other hand, two levels of sequencing, i.e., job sequencing and group sequencing, GA operators should be capable of exploring and exploiting the entire search space according to all flexibilities and sequencing levels. In this proposed GA, the group processing version of the Precedence Preserving Order-based Crossover (POX) and three mutation operators, that is, shifting genes' loci, shifting processing machines, and shifting operation combinations, are implemented.

5.1.3.1 Crossover Crossover, which was applied on both the numerical and object sections of two individuals, focuses on all different flexibilities and sequencing levels in the IPPGS problem to construct new neighborhood space toward different dimensions for the GA. Crossover is executed with the probability of P_c for each population member; in other words, it is applied by the GA with the number of $\frac{P_c.N_{pop}}{2}$ times in each iteration, and hence, $n_c = P_c.N_{pop}$ offspring are reproduced. The Precedence Preserving Order-based Crossover (POX) (Luo et al. 2017; Zhang et al. 2005; Zhang and Wong 2015a) with group consideration is applied. In this recombination, operations

Section 1:

2	1

Section 2:

$O^1_{1,1}$	$O^2_{3,1}$	$O^2_{4,1}$	$O^1_{2,1}$	$O^2_{4,3}$	$O^3_{3,2}$	$O^2_{5,1}$	$O^3_{4,2}$	$O^1_{1,4}$	$O^2_{3,3}$	$O^4_{4,4}$	$O^3_{5,2}$	$O^4_{3,4}$
$O^2_{5,3}$	$O^3_{4,5}$	$O^1_{2,4}$	$O^4_{5,4}$	$O^5_{1,5}$	$O^5_{2,5}$	$O^3_{5,5}$	$O^3_{3,5}$					

Fig. 4 A chromosome for the IPPGS problem instance

of the same job between two individuals are exchanged in the situation precedence relationships are satisfied. The approach of the crossover operator is illustrated by an example in Fig. 5. P_1 and P_2 are selected from the population as parent chromosomes, and C_1 and C_2 are initially two empty indivisuals. A group is selected, and all jobs within it are put in *JobSet* 1. Other remaining jobs are gathered in *JobSet* 2. Then, for the object section, the operations of jobs in *JobSet* 1 of P_1 and P_2 are copied to the corresponding positions in C_1 and C_2. Afterwards, the vacancies in C_1 and C_2 are filled by the operations of jobs in *JobSet* 2 of P_2 and P_1 respectively with the same orders from left to right. Finally, remaining vacancies are omitted from new offspring. For the numerical section, the related positions of the selected group in C_1 and C_2 are filled by the corresponding numbers in P_1 and P_2, and for other groups, contrariwise, they are filled by the corresponding numbers in P_2 and P_1 respectively.

5.1.3.2 Shifting genes' loci The mutation of shifting genes' loci is used to alter operation sequences in the object section of an individual while the numerical sec-

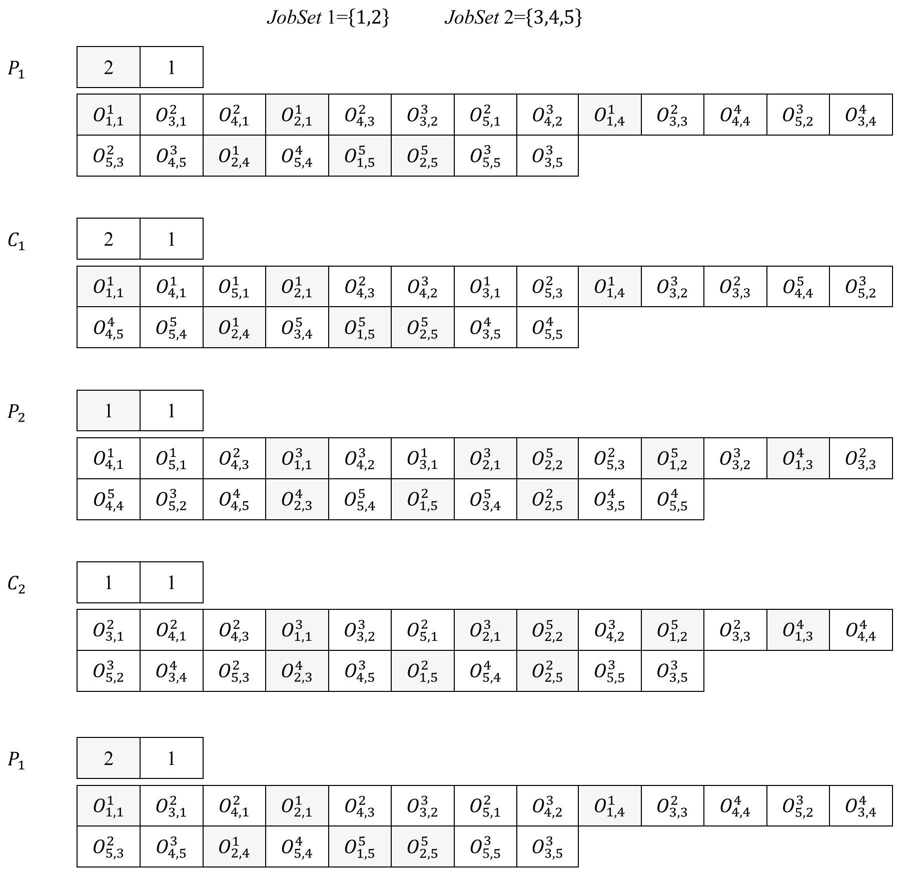

JobSet 1={1,2} *JobSet* 2={3,4,5}
P_1 | 2 | 1
$O^1_{1,1}$ $O^2_{3,1}$ $O^2_{4,1}$ $O^1_{2,1}$ $O^2_{4,3}$ $O^3_{3,2}$ $O^2_{5,1}$ $O^3_{4,2}$ $O^1_{1,4}$ $O^2_{3,3}$ $O^4_{4,4}$ $O^3_{5,2}$ $O^4_{3,4}$
$O^2_{5,3}$ $O^3_{4,5}$ $O^1_{2,4}$ $O^4_{5,4}$ $O^5_{1,5}$ $O^5_{2,5}$ $O^3_{5,5}$ $O^3_{3,5}$
C_1 | 2 | 1
$O^1_{1,1}$ $O^1_{4,1}$ $O^1_{5,1}$ $O^1_{2,1}$ $O^2_{4,3}$ $O^3_{4,2}$ $O^1_{3,1}$ $O^2_{5,3}$ $O^1_{1,4}$ $O^3_{3,2}$ $O^2_{3,3}$ $O^5_{4,4}$ $O^3_{5,2}$
$O^4_{4,5}$ $O^5_{5,4}$ $O^1_{2,4}$ $O^5_{3,4}$ $O^5_{1,5}$ $O^5_{2,5}$ $O^4_{3,5}$ $O^4_{5,5}$
P_2 | 1 | 1
$O^1_{4,1}$ $O^1_{5,1}$ $O^2_{4,3}$ $O^3_{1,1}$ $O^3_{4,2}$ $O^1_{3,1}$ $O^3_{2,1}$ $O^5_{2,2}$ $O^2_{5,3}$ $O^5_{1,2}$ $O^3_{3,2}$ $O^4_{1,3}$ $O^2_{3,3}$
$O^5_{4,4}$ $O^3_{5,2}$ $O^4_{4,5}$ $O^4_{2,3}$ $O^5_{5,4}$ $O^2_{1,5}$ $O^5_{3,4}$ $O^2_{2,5}$ $O^4_{3,5}$ $O^4_{5,5}$
C_2 | 1 | 1
$O^2_{3,1}$ $O^2_{4,1}$ $O^2_{4,3}$ $O^3_{1,1}$ $O^3_{3,2}$ $O^2_{5,1}$ $O^3_{2,1}$ $O^5_{2,2}$ $O^3_{4,2}$ $O^5_{1,2}$ $O^2_{3,3}$ $O^4_{1,3}$ $O^4_{4,4}$
$O^3_{5,2}$ $O^4_{3,4}$ $O^2_{5,3}$ $O^4_{2,3}$ $O^3_{4,5}$ $O^2_{1,5}$ $O^4_{5,4}$ $O^2_{2,5}$ $O^3_{5,5}$ $O^3_{3,5}$
P_1 | 2 | 1
$O^1_{1,1}$ $O^2_{3,1}$ $O^2_{4,1}$ $O^1_{2,1}$ $O^2_{4,3}$ $O^3_{3,2}$ $O^2_{5,1}$ $O^3_{4,2}$ $O^1_{1,4}$ $O^2_{3,3}$ $O^4_{4,4}$ $O^3_{5,2}$ $O^4_{3,4}$
$O^2_{5,3}$ $O^3_{4,5}$ $O^1_{2,4}$ $O^4_{5,4}$ $O^5_{1,5}$ $O^5_{2,5}$ $O^3_{5,5}$ $O^3_{3,5}$

Fig. 5 Illustration of the crossover operator

tion remains unchanged. To prevent the violation of precedence relationship between operations of the same job, a precedence preserving version of shifting genes' loci is exerted with the probability of P_m, and therefore $n_m = P_m.N_{pop}$ individuals are reproduced in each iteration as follows:

Step 1: select an individual, say, I_1.

Step 2: make a copy of I_1 and obtain I_2.

Step 3: select two loci on I_2 randomly (L_1 and L_2).

Step 4: if the related operations on L_1 and L_2 belong to the same job, eliminate L_1 and L_2, and go to Step 3; otherwise, go to Step 5.

Step 5: assume the operations on L_1 and L_2 belong to Job 1 and Job 2, respectively. Eliminate the operation on L_1, and shift all operations of Job 2 backward one by one to keep their orders until the operation on L_2 is shifted, and the gene on L_2 is empty.

Step 6: shift forward the operation of Job 1 one by one from left to right to keep their orders until all genes are filled. Other operations do not move.

Figure 6 illustrates precedence preserving shifting genes' loci. Numbers on the arcs show the shifting orders of operations.

5.1.3.3 Shifting processing machines In this mutation, which affects only the object section of an individual, the same operations of the jobs in the same group can change their processing machine simultaneously with the probability of P_k. It is performed on one of the offspring obtained from the crossover. Figure 7 explains how this mutation works on C_1. In this case, the processing machine of the second operations of the jobs in group 2 is changed from machine 3 to machine 2.

5.1.3.4 Shifting operation combinations While shifting genes' loci and shifting processing machines focus on respectively sequence and operation flexibilities, the main mutation idea in shifting operation combinations is based on the process flexibility and the number of OR link paths. It is applied to the other offspring which is not mutated with shifting processing machines, here C_2. In this mutation, another opera-

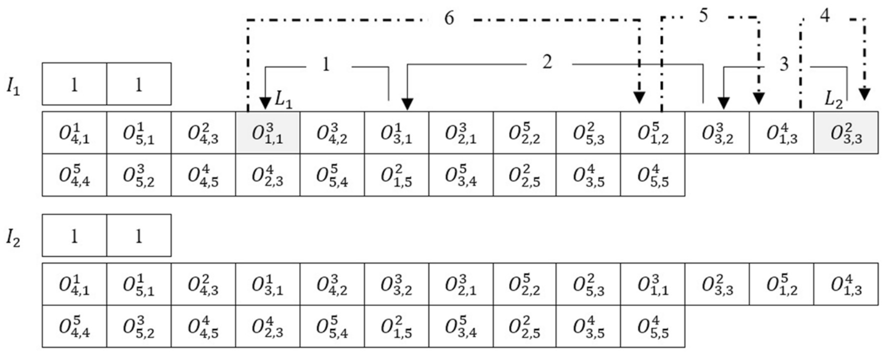
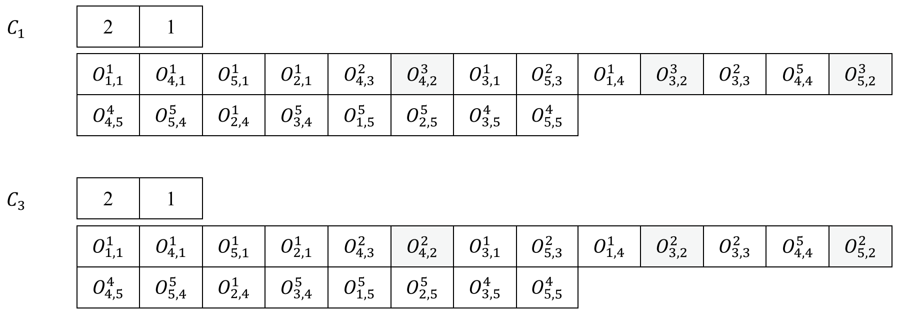

6
5
4
1
2
3
L_1
L_2
I_1
1 1
$O^1_{4,1}$ $O^1_{5,1}$ $O^2_{4,3}$ $O^3_{1,1}$ $O^3_{4,2}$ $O^1_{3,1}$ $O^3_{2,1}$ $O^5_{2,2}$ $O^2_{5,3}$ $O^5_{1,2}$ $O^3_{3,2}$ $O^4_{1,3}$ $O^2_{3,3}$
$O^5_{4,4}$ $O^3_{5,2}$ $O^4_{4,5}$ $O^4_{2,3}$ $O^5_{5,4}$ $O^2_{1,5}$ $O^5_{3,4}$ $O^2_{2,5}$ $O^4_{3,5}$ $O^4_{5,5}$
I_2
1 1
$O^1_{4,1}$ $O^1_{5,1}$ $O^2_{4,3}$ $O^1_{3,1}$ $O^3_{4,2}$ $O^3_{3,2}$ $O^3_{2,1}$ $O^5_{2,2}$ $O^2_{5,3}$ $O^3_{1,1}$ $O^2_{3,3}$ $O^5_{1,2}$ $O^4_{1,3}$
$O^5_{4,4}$ $O^3_{5,2}$ $O^4_{4,5}$ $O^4_{2,3}$ $O^5_{5,4}$ $O^2_{1,5}$ $O^5_{3,4}$ $O^2_{2,5}$ $O^4_{3,5}$ $O^4_{5,5}$

Fig. 6 Illustration of precedence preserving shifting genes' loci



C_1

2	1

$O_{1,1}^{1}$	$O_{4,1}^{1}$	$O_{5,1}^{1}$	$O_{2,1}^{1}$	$O_{4,3}^{2}$	$O_{4,2}^{3}$	$O_{3,1}^{1}$	$O_{5,3}^{2}$	$O_{1,4}^{1}$	$O_{3,2}^{3}$	$O_{3,3}^{2}$	$O_{4,4}^{5}$	$O_{5,2}^{3}$
$O_{4,5}^{4}$	$O_{5,4}^{5}$	$O_{2,4}^{1}$	$O_{3,4}^{5}$	$O_{1,5}^{5}$	$O_{2,5}^{5}$	$O_{3,5}^{4}$	$O_{5,5}^{4}$					

C_3

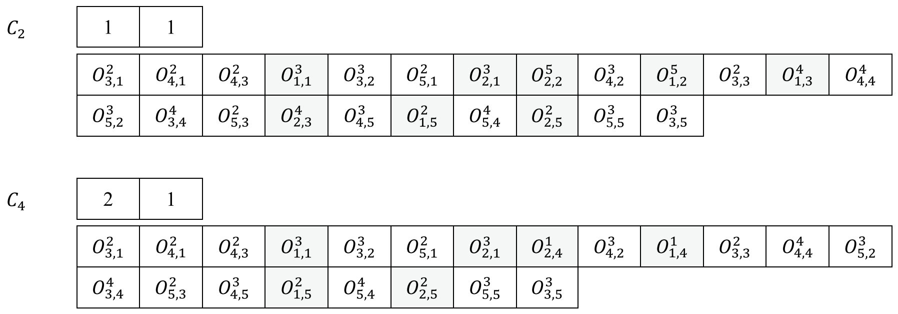
2	1

$O_{1,1}^{1}$	$O_{4,1}^{1}$	$O_{5,1}^{1}$	$O_{2,1}^{1}$	$O_{4,3}^{2}$	$O_{4,2}^{2}$	$O_{3,1}^{1}$	$O_{5,3}^{2}$	$O_{1,4}^{1}$	$O_{3,2}^{2}$	$O_{3,3}^{2}$	$O_{4,4}^{5}$	$O_{5,2}^{2}$
$O_{4,5}^{4}$	$O_{5,4}^{5}$	$O_{2,4}^{1}$	$O_{3,4}^{5}$	$O_{1,5}^{5}$	$O_{2,5}^{5}$	$O_{3,5}^{4}$	$O_{5,5}^{4}$					

Fig. 7 Shifting processing machines

tion combination is selected for a group which has more than one operation combination – or have at least one OR node in its graph- with the probability of P_h (usually $P_h < 0.1$), and the related number in the numerical section is changed. This alteration completely affects the object section, and operations in one OR link path are substituted by the operations in a different OR link path but the same OR nodes. Omitted operations are replaced with new ones in the same order. If new operations are less than omitted one, empty genes should be eliminated; otherwise, if new operations are more than omitted ones, the genes with the related order should be multiplied to create enough genes for further operations. The mutation of shifting operation combinations helps GA to avoid getting trapped in a suboptimal solution or premature convergence.

An overview of this mutation is given in Fig. 8, where operation combination 2 with OR link path $\{O_1, O_4, O_5\}$ replace operation combination 1 with OR link path $\{O_1, O_2, O_3, O_5\}$ for all jobs in group 1. Hence, two empty genes are eliminated. When assigning processing machines to newly added operations, the same machine should be chosen for the same operations of all jobs in a group.



C_2

1	1

$O_{3,1}^{2}$	$O_{4,1}^{2}$	$O_{4,3}^{2}$	$O_{1,1}^{3}$	$O_{3,2}^{3}$	$O_{5,1}^{2}$	$O_{2,1}^{3}$	$O_{2,2}^{5}$	$O_{4,2}^{3}$	$O_{1,2}^{5}$	$O_{3,3}^{2}$	$O_{1,3}^{4}$	$O_{4,4}^{4}$
$O_{5,2}^{3}$	$O_{3,4}^{4}$	$O_{5,3}^{2}$	$O_{2,3}^{4}$	$O_{4,5}^{3}$	$O_{1,5}^{2}$	$O_{5,4}^{4}$	$O_{2,5}^{2}$	$O_{5,5}^{3}$	$O_{3,5}^{3}$			

C_4

2	1

$O_{3,1}^{2}$	$O_{4,1}^{2}$	$O_{4,3}^{2}$	$O_{1,1}^{3}$	$O_{3,2}^{3}$	$O_{5,1}^{2}$	$O_{2,1}^{3}$	$O_{2,4}^{1}$	$O_{4,2}^{3}$	$O_{1,4}^{1}$	$O_{3,3}^{2}$	$O_{4,4}^{4}$	$O_{5,2}^{3}$
$O_{3,4}^{4}$	$O_{5,3}^{2}$	$O_{4,5}^{3}$	$O_{1,5}^{2}$	$O_{5,4}^{4}$	$O_{2,5}^{2}$	$O_{5,5}^{3}$	$O_{3,5}^{3}$					

Fig. 8 Shifting operation combinations

5.1.4 Population evolution and truncation

As explained before, in each iteration of the proposed GA, the crossover operator produces n_c individuals which $\frac{n_c}{2}$ of them are used in the mutation of shifting processing machines, and remaining $\frac{n_c}{2}$ offspring are mutated through shifting operation combinations. Shifting processing machines may not produce new individuals if the selected operations do not have an alternative machine. Shifting operation combinations will not produce new individuals if the chosen group has only one operation combination. On the other hand, shifting genes' loci reproduces n_m new individuals in each iteration. In total, at most $2n_c + n_m$ individuals are added to initial N_{pop} individuals. Therefore, the worst individuals are eliminated at the end of each iteration to reduce the population size to N_{pop}. The general procedure of the proposed GA is as follows:

```
Begin iteration $i$
Initializing the population by generating $N_{pop}$ initial solutions.
Select $n_c$ pairs of parent individuals by tournament selection method.
[offspring set 1, offspring set 2] ← crossover operator [selected parent individuals]
[offspring set 3] ← shifting processing machines [offspring set 1]
[offspring set 4] ← shifting operation combinations [offspring set 2]
While (current population size > $N_{pop}$)
    [$pop_i \leftarrow pop_i$ \the worst individual]
If (Terminate criteria are not fulfilled)
    [$i \leftarrow i + 1$
    Go to the next iteration.]
Else [Terminate]
```

5.2 Hybrid water cycle algorithm (HWCA)

The WCA introduced by Eskandar et al. (2012) is a population-based metaheuristic algorithm. It is inspired by the hydrologic cycle process in nature and how rivers and streams flow downhill toward the sea. Despite the capabilities of the WCA to explore and exploit the search space of complex problems, the WCA was utilized in only two studies. Nayak et al. (2018) used WCA for the multiprocessor scheduling problem and compared it to GA. Their results showed that WCA outperforms GA. An improved discrete WCA (DWCA) was employed in Gao et al. (2017) for solving remanufacturing rescheduling problem.

Using the water from the rain, streams are created. The population of streams, as individuals, forms an initial population. After that, a number of N_{sr} good individuals in terms of the objective function are considered the sea and rivers, and the best individual among them is chosen as the sea. Hence, N_{sr} is the summation of the number of rivers plus one (sea). All other individuals are considered streams which flow to rivers and sea. In a N-dimension problem (i.e., a problem with N design variables), a stream is defined as follows:

$$\text{A stream} = \left[x_1, x_2, x_3, \ldots, x_N\right], \tag{38}$$

Consequently, a sorted population with N_{pop} streams are given as follows (Sadollah et al. 2015b):

$$\text{Population of streams} = \begin{bmatrix} Stream_1 \\ Stream_2 \\ Stream_3 \\ \vdots \\ Stream_{N_{pop}} \end{bmatrix} = \begin{bmatrix} x_1^1 & x_2^1 & x_3^1 & \cdots & x_N^1 \\ x_1^2 & x_2^2 & x_3^2 & \cdots & x_N^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1^{N_{pop}} & x_2^{N_{pop}} & x_3^{N_{pop}} & \cdots & x_N^{N_{pop}} \end{bmatrix} \tag{39}$$

which can be given as a sorted population of streams based on the optimization criterion as follows:

$$Sorted\,population\,of\,streams = \begin{bmatrix} Sea \\ River_1 \\ River_2 \\ River_3 \\ \vdots \\ Stream_{N_{sr}+1} \\ Stream_{N_{sr}+2} \\ Stream_{N_{sr}+3} \\ \vdots \\ Stream_{N_{pop}} \end{bmatrix} \tag{40}$$

The number of streams which flow directly or indirectly to the rivers or sea is $N_{Streams}$, and is calculated using Eq. (41) as follows:

$$N_{Streams} = N_{pop} - N_{sr} \tag{41}$$

The sea and each river absorb a number of streams according to their flow magnitude. In fact, some of $N_{Streams}$ streams flow to the sea, and others flow to rivers. The number of streams which flow to the sea and each river is calculated using Eqs. (42) and (43) as follows (Eskandar et al. 2012):

$$C_n = Cost_n - Cost_{N_{sr}+1} \tag{42}$$

$$NS_n = round\left\{ \left| \frac{C_n}{\sum_{n=1}^{N_{sr}} C_n} \right| \times N_{Streams} \right\}, \quad n = 1, 2, \ldots, N_{sr} \tag{43}$$

where NS_1 is the designated streams for the sea, NS_2 is the designated streams for the best river, NS_3 is the designated streams for the second-best river, and so on. So,

$NS_{N_{sr}}$ is the number of streams flow to the worst river. Using the above equations, streams tend to move toward the sea and rivers based on their quality in the term of the fitness function. In this way, the sea as the best individual possesses more streams. Each stream is controlled by one of the rivers or sea. Therefore, the sum of designated streams ($\sum_{n=1}^{N_{sr}} NS_n$) should be equal to the total numbers of streams ($N_{Streams}$). However, $\sum_{n=1}^{N_{sr}} NS_n$ may not be equal to $N_{Streams}$ after the designation of streams. In this situation, some modifications on NS_n are needed (Sadollah et al. 2015b).

Hence, as the exploitation phase in the WCA, streams move toward their related river or sea, and rivers also move toward the sea. The new positions for streams and rivers have been proposed as follows:

$$\vec{X}_{Stream}(t+1) = \vec{X}_{Stream}(t) + rand \times C \times \left(\vec{X}_{River}(t) - \vec{X}_{Stream}(t)\right), \tag{44}$$

$$\vec{X}_{Stream}(t+1) = \vec{X}_{Stream}(t) + rand \times C \times \left(\vec{X}_{Sea}(t) - \vec{X}_{Stream}(t)\right), \tag{45}$$

$$\vec{X}_{River}(t+1) = \vec{X}_{River}(t) + rand \times C \times \left(\vec{X}_{Sea}(t) - \vec{X}_{River}(t)\right), \tag{46}$$

where $1 < C < 2$ and rand is a uniformly distributed random number between 0 and 1. When the solution provided by a stream is better than its joining river, then the positions of river and stream are exchanged. Similarly, the exchange of the positions can happen for rivers and sea, and sea and streams as well.

Moreover, for the exploration phase, evaporation and raining processes are defined in Sadollah et al. (2015b) to enhance the capability of the algorithm to escape from local optima. Considering the mathematical perspective, if norm distances among rivers, streams, and sea are smaller than a predefined value (d_{max}), new streams are generated flowing into the rivers and sea. The following criteria are used as evaporation conditions:

$$\|\vec{X}_{Sea} - \vec{X}^{i}_{River}\| < d_{max} \text{ or } rand < 0.1 \quad i = 1, 2, \ldots, N_{sr} - 1, \tag{47}$$

$$\|\vec{X}_{Sea} - \vec{X}^{i}_{Stream}\| < d_{max} \quad i = 1, 2, \ldots, NS_1, \tag{48}$$

where d_{max} is a small number close to zero, which controls the search intensity near the sea. The value of d_{max} decreases in each iteration as follows:

$$d_{max}(t+1) = d_{max}(t) - \frac{d_{max}(t)}{max\_iteration} \tag{49}$$

More details about the original idea of the WCA, as well as a comparison to some other population-based algorithms, can be found in Eskandar et al. (2012), Sadollah et al. (2015a), Sadollah et al. (2015b), and Jafar et al. (2018). The steps of WCA are summarized as follows:

Step 1. Set the main parameters of the WCA (N_{pop}, N_{sr}, d_{max}, max_*iteration*).

Step 2. Generate an initial population of streams randomly using Eq. (38) and Eq. (39).

Step 3. Calculate the cost of all streams in the population.

Step 4. Choose a number of N_{sr} from the best individuals as the sea and rivers. The individual, which has the best fitness value, is considered the sea.

Step 5. Determine the flow magnitude for rivers and sea using Eqs. (42) and (43). If $\sum_{n=1}^{N_{sr}} NS_n = N_{Streams}$ go to Step 7; otherwise, Step 6.

Step 6. Perform NS_n modification procedure until $\sum_{n=1}^{N_{sr}} NS_n = N_{Streams}$ and then go to Step 7.

Step 7. Streams flow to its joining river, according to Eq. (44). If the solution provided by a stream is better than its joining river, then exchange the positions of river and stream and go to Step 8; otherwise, go to Step 9.

Step 8. If the solution provided by a new river is better than the sea, then exchange the positions of a new river and sea.

Step 9. Streams move towards the sea based on Eq. (45). If the solution provided by a stream is better than the sea, then exchange their positions.

Step 10.Rivers flow to the sea using Eq. (46). If the solution provided by a river is better than the sea, then exchange their positions.

Step 11. Check evaporation conditions for rivers and the sea using Eq. (47). If it is satisfied, the raining process will take place, and the river is replaced by a new randomly generated stream.

Step 12. Check evaporation conditions for streams and the sea using Eq. (48). If it is satisfied, the raining process will take place, and the stream is replaced by a new randomly generated one.

Step 13. Reduce the value of d_{max} using Eq. (49).

Step 14. If the stop criterion is satisfied, output the sea as the best solution; otherwise, return to Step 7 and repeat.

5.2.1 Encoding and decoding scheme

The WCA is a continuous algorithm in essence. Therefore, in order to apply the WCA for discrete search space (like the IPPGS problem), a discrete strategy should be provided. Due to all kinds of flexibility and two levels of job and group sequencing in the IPPGS problem, a discrete strategy is needed to cover all search space as much as possible. In this paper, $(3m + 2)$ random number matrixes (m is the number of machines) with elements distributed uniformly between $(0, 1)$ are used as follows:

Random number matrix	Size
X	$g \times h_{max}$
$Z_k \forall k = 1, 2, \dots, m$	$g \times J_{max}$
$J_k \forall k = 1, 2, \dots, m$	$1 \times n * J_{max}$
$S_k \forall k = 1, 2, \dots, m$	$1 \times g$

Random number matrix	Size
K	$1 \times m$

where g: the number of groups, n: the number of jobs, m: the number of machines, h_{max}: the maximum number of combinations, J_{max}: the maximum number of operations.

The set of above matrixes represents a stream as an individual, and therefore it is equivalent to a solution of the problem. Matrix X is used to identify the operation combination of each job. Based on the group, including the job, the operation combinations of the group according to its AND/OR graph, and the maximum random number of the related row in this matrix, an operation combination is selected for each job. For the problem illustrated in Sect. 5.1.1, five jobs are categorized in two groups with at most two operation combinations. So, matrix X can be as follows:

$$X = \begin{bmatrix} 0.0835 & 0.6260 \\ 0.3215 & 0.5901 \end{bmatrix}$$

Jobs 1 and 2 in the first group, perform through their second operation combination (O_1, O_4, O_5), while jobs 3,4, and 5 in the second group used their only one available combination (O_1, O_2, O_3, O_4, O_5).

Matrixes Z_k identify the selected machine for each operation. The processing machine is selected based on the group, including the job, the selected operation combination of the group according to matrix X, the available operations in the related combination, the available processing machines for each operation, and, finally, the maximum of associated random numbers. As an example of the considered problem instance, there are five matrixes $Z_1, Z_2, \ldots, Z_5$ as follows:

$$Z_1 = \begin{bmatrix} 0.7829 & 0.0427 & 0.6730 & 0.7669 & 0.1932 \\ 0.6938 & 0.3782 & 0.4775 & 0.6671 & 0.2959 \end{bmatrix}$$

$$Z_2 = \begin{bmatrix} 0.3119 & 0.6289 & 0.9976 & 0.9274 & 0.1248 \\ 0.1790 & 0.1015 & 0.8116 & 0.9175 & 0.5306 \end{bmatrix}$$

$$Z_3 = \begin{bmatrix} 0.8352 & 0.6195 & 0.9727 & 0.3569 & 0.5906 \\ 0.3225 & 0.3606 & 0.3278 & 0.6627 & 0.6604 \end{bmatrix}$$

$$Z_4 = \begin{bmatrix} 0.7150 & 0.1386 & 0.8770 & 0.1892 & 0.8112 \\ 0.8562 & 0.5882 & 0.3531 & 0.9345 & 0.0193 \end{bmatrix}$$

$$Z_5 = \begin{bmatrix} 0.4035 & 0.3480 & 0.0474 & 0.8936 & 0.7218 \\ 0.1220 & 0.1217 & 0.3424 & 0.0548 & 0.8778 \end{bmatrix}$$

Regarding matrix X, Jobs 1 and 2 in the first group, should be done through their second operation combination, i.e.,O_1, O_4, and O_5. The first operation (O_1) can be processed on M_1, M_3, and M_4. In order to select the processing machine, the elements in the first row and the second column of Z_1, Z_3, and Z_4 are considered (0.0427, 0.6195, 0.1386), and the corresponding machine with the maximum random number is selected (M_3). The random vectors K, S_k, and J_k are used to determine the sequence of the operations in the real schedule. The object-coding vectors K' and S'_k are constructed based on sorted arrays of random number vectors K and S_k in ascending order, respectively. Vector K' indicates the order of machines, while vectors S'_k determine the sequence of groups on machine k.

Vectors J_k need some modification to use as an ordered list of operations for entering to the schedule. This modification procedure will be applied to random vectors J_k through six steps as follows:

(1) Determine the ascending order of J_k members to generate the corresponding object-coding vector (J_k^1).
(2) Remove omitted operations based on the information obtained from $Z_1, Z_2, \ldots, Z_m$, and X (J_k^2).
(3) Categorize operations in J_k^2 based on the corresponding vector S'_k (J_k^3).
(4) Apply *Repair Schedule_1* on all smallest sections of J_k^3 ($J'^{3}_{k,q}, q = 1, 2, \ldots, g \times m$) to reorder the operations according to precedence relationship (J_k^4).
(5) Apply *Repair Schedule_2* on all smallest sections of J_k^4 ($J'^{4}_{k,q}, q = 1, 2, \ldots, g \times m$) to ensure that all the same operations of the jobs in the same group are processed consecutively according to group technology assumption (J_k^5).
(6) Order operations in J_k^5 based on their processing machine and K' (J_f).

The general procedures of *Repair Schedule_1* and *Repair Schedule_2* are presented below:

Repair Schedule_1

```
For $j \epsilon J'^{3}_{k,q}$
    $pred(j)$ ← precedence operations of $j$
    If $pred(j)$ is not empty then
        For $p \epsilon\ pred(j)$
            If $p \epsilon J'^{3}_{k,q}$ then
                $PRED(j) \leftarrow PRED(j) + \{p\}$
        End For
    End If
End For
While $J'^{3}_{k,q}$ is not empty
    For $j \epsilon J'^{3}_{k,q}$
        If all $PRED(j)$ members $\epsilon\, J'^{4}_{k,q}$
            Break;
        End If
    End For
    $J'^{4}_{k,q} \leftarrow J'^{4}_{k,q} + \{j\}$
    $J'^{3}_{k,q} \leftarrow J'^{3}_{k,q} - \{j\}$
End While
Put $J'^{4}_{k,q}$ vectors in order to obtain $J^{4}_{k}$.
```

Repair Schedule_2

```
$R$ ← the related job of $J'^{4}_{k,q}$ members
$R'$ ← remove duplicate members of $R$
For $r \epsilon R'$
    $J'^{5}_{k,q} \leftarrow J'^{5}_{k,q}$ + [the members of $J'^{4}_{k,q}$ which its equivalent members in $R$ are equal to $r$]
End For
Put $J'^{5}_{k,q}$ vectors in order to obtain $J^{5}_{k}$.
```

To clarify how to obtain K' and S'_k, and covert J_k to J_f through six steps as well as *Repair Schedule_1* and *Repair Schedule_2*, the problem instance illustrated in Sect. 5.1.1 is used regarding matrixes X and Z_k given above. Due to space limitations, some matrixes are transposed.

$$S_1 = \begin{bmatrix} 0.8608 & 0.0856 \end{bmatrix} \rightarrow S'_1 = \begin{bmatrix} S^2 & S^1 \end{bmatrix}$$

$$S_2 = \begin{bmatrix} 0.4665 & 0.0674 \end{bmatrix} \rightarrow S'_2 = \begin{bmatrix} S^2 & S^1 \end{bmatrix}$$

$$S_3 = \begin{bmatrix} 0.4981 & 0.8884 \end{bmatrix} \rightarrow S'_3 = \begin{bmatrix} S^1 & S^2 \end{bmatrix}$$

$$S_4 = \begin{bmatrix} 0.4874 & 0.2332 \end{bmatrix} \rightarrow S'_4 = \begin{bmatrix} S^2 & S^1 \end{bmatrix}$$

$$S_5 = \begin{bmatrix} 0.2295 & 0.8616 \end{bmatrix} \rightarrow S'_5 = \begin{bmatrix} S^1 & S^2 \end{bmatrix}$$

$$K = \begin{bmatrix} 0.6580 & 0.8896 & 0.1096 & 0.4378 & 0.2802 \end{bmatrix} \rightarrow K' = \begin{bmatrix} M_3 & M_5 & M_4 & M_1 & M_2 \end{bmatrix}$$

$$J_1^T = \begin{bmatrix} 0.2469 \\ 0.0954 \\ 0.6480 \\ 0.6505 \\ 0.8354 \\ 0.3215 \\ 0.0496 \\ 0.7485 \\ 0.4486 \\ 0.8992 \\ 0.7061 \\ 0.3429 \\ 0.7231 \\ 0.0240 \\ 0.1709 \\ 0.4431 \\ 0.2690 \\ 0.9594 \\ 0.2687 \\ 0.4169 \\ 0.4723 \\ 0.4081 \\ 0.9254 \\ 0.9596 \\ 0.0149 \end{bmatrix} J_2^T = \begin{bmatrix} 0.0089 \\ 0.3526 \\ 0.4334 \\ 0.8574 \\ 0.8357 \\ 0.8065 \\ 0.2832 \\ 0.5678 \\ 0.8160 \\ 0.8999 \\ 0.8314 \\ 0.6382 \\ 0.2788 \\ 0.4911 \\ 0.3993 \\ 0.4333 \\ 0.5597 \\ 0.7753 \\ 0.9867 \\ 0.3801 \\ 0.3334 \\ 0.4620 \\ 0.7390 \\ 0.6463 \\ 0.1567 \end{bmatrix} J_3^T = \begin{bmatrix} 0.8149 \\ 0.5934 \\ 0.1398 \\ 0.0844 \\ 0.0499 \\ 0.6014 \\ 0.6535 \\ 0.2990 \\ 0.0983 \\ 0.5241 \\ 0.0348 \\ 0.3430 \\ 0.5824 \\ 0.2783 \\ 0.6976 \\ 0.1752 \\ 0.9448 \\ 0.6077 \\ 0.7722 \\ 0.2133 \\ 0.9758 \\ 0.8263 \\ 0.5674 \\ 0.3796 \\ 0.4716 \end{bmatrix} J_4^T = \begin{bmatrix} 0.1405 \\ 0.5852 \\ 0.7519 \\ 0.9721 \\ 0.5459 \\ 0.7896 \\ 0.4897 \\ 0.2561 \\ 0.8596 \\ 0.1202 \\ 0.7578 \\ 0.2165 \\ 0.4210 \\ 0.3398 \\ 0.2037 \\ 0.1932 \\ 0.7145 \\ 0.9480 \\ 0.4754 \\ 0.3829 \\ 0.5554 \\ 0.9912 \\ 0.9688 \\ 0.4766 \\ 0.5430 \end{bmatrix} J_5^T = \begin{bmatrix} 0.8799 \\ 0.6677 \\ 0.2418 \\ 0.0315 \\ 0.9432 \\ 0.7992 \\ 0.9729 \\ 0.8866 \\ 0.0276 \\ 0.1778 \\ 0.9571 \\ 0.7862 \\ 0.0921 \\ 0.2873 \\ 0.6663 \\ 0.6164 \\ 0.6792 \\ 0.0596 \\ 0.6809 \\ 0.0297 \\ 0.8463 \\ 0.5239 \\ 0.8245 \\ 0.9119 \\ 0.0597 \end{bmatrix}$$

$$\left(J_1^1\right)^T=\begin{bmatrix}O_{5,5}\\O_{3,4}\\O_{2,2}\\O_{1,2}\\O_{3,5}\\O_{1,1}\\O_{4,4}\\O_{4,2}\\O_{2,1}\\O_{3,2}\\O_{5,2}\\O_{4,5}\\O_{4,1}\\O_{2,4}\\O_{5,1}\\O_{1,3}\\O_{1,4}\\O_{3,1}\\O_{3,3}\\O_{2,3}\\O_{1,5}\\O_{2,5}\\O_{5,3}\\O_{4,3}\\O_{5,4}\end{bmatrix}\left(J_2^1\right)^T=\begin{bmatrix}O_{1,1}\\O_{5,5}\\O_{3,3}\\O_{2,2}\\O_{5,1}\\O_{1,2}\\O_{4,5}\\O_{3,5}\\O_{4,1}\\O_{1,3}\\O_{5,2}\\O_{3,4}\\O_{4,2}\\O_{2,3}\\O_{3,2}\\O_{5,4}\\O_{5,3}\\O_{4,3}\\O_{2,1}\\O_{2,4}\\O_{3,1}\\O_{1,5}\\O_{1,4}\\O_{2,5}\\O_{4,4}\end{bmatrix}\left(J_3^1\right)^T=\begin{bmatrix}O_{3,1}\\O_{1,5}\\O_{1,4}\\O_{2,4}\\O_{1,3}\\O_{4,1}\\O_{4,5}\\O_{3,4}\\O_{2,3}\\O_{3,2}\\O_{5,4}\\O_{5,5}\\O_{2,5}\\O_{5,3}\\O_{3,3}\\O_{1,2}\\O_{2,1}\\O_{4,3}\\O_{2,2}\\O_{3,5}\\O_{4,4}\\O_{1,1}\\O_{5,2}\\O_{4,2}\\O_{5,1}\end{bmatrix}\left(J_4^1\right)^T=\begin{bmatrix}O_{2,5}\\O_{1,1}\\O_{4,1}\\O_{3,5}\\O_{3,2}\\O_{2,3}\\O_{3,4}\\O_{4,5}\\O_{3,3}\\O_{4,4}\\O_{5,4}\\O_{2,2}\\O_{5,5}\\O_{1,5}\\O_{5,1}\\O_{1,2}\\O_{4,2}\\O_{1,3}\\O_{3,1}\\O_{2,1}\\O_{2,4}\\O_{4,3}\\O_{5,3}\\O_{1,4}\\O_{5,2}\end{bmatrix}\left(J_5^1\right)^T=\begin{bmatrix}O_{2,4}\\O_{4,5}\\O_{1,4}\\O_{4,3}\\O_{5,5}\\O_{3,3}\\O_{2,5}\\O_{1,3}\\O_{3,4}\\O_{5,2}\\O_{4,1}\\O_{3,5}\\O_{1,2}\\O_{4,2}\\O_{4,4}\\O_{3,2}\\O_{2,1}\\O_{5,3}\\O_{5,1}\\O_{1,1}\\O_{2,3}\\O_{5,4}\\O_{1,5}\\O_{3,1}\\O_{2,2}\end{bmatrix}$$

$$J_1^2=\left[\,O_{4,1}\;O_{2,4}\;\;O_{5,1}\;O_{1,4}\;O_{3,1}\;\right],\quad J_2^2=\left[\,O_{3,3}\;O_{5,3}\;O_{4,3}\,\right],$$

$$J_3^2=\left[\,O_{4,5}\;O_{3,2}\;\;O_{5,5}\;O_{2,1}\;\;O_{3,5}\;O_{1,1}\;\;O_{5,2}\;O_{4,2}\;\;\right],$$

$$J_4^2=\left[\,O_{3,4}\;O_{4,4}\;O_{5,4}\,\right],J_5^2=\left[\,O_{2,5}\;O_{1,5}\,\right]$$

$$J_1^3=\left[\left[J'^3_{1,1}\right]\;\left[J'^3_{1,2}\right]\right]=\left[\,[O_{4,1}\;O_{5,1}\;\;O_{3,1}]\;[O_{2,4}\;O_{1,4}]\;\right],$$
$$J_2^3=\left[\left[J'^3_{2,1}\right]\;\left[J'^3_{2,2}\right]\right]=\left[\,[O_{3,3}\;O_{5,3}\;O_{4,3}][\varnothing]\,\right],$$

$$J_3^3=\left[\left[J'^3_{3,1}\right]\;\left[J'^3_{3,2}\right]\right]=\left[\,[O_{2,1}\;O_{1,1}]\;\;[O_{4,5}\;O_{3,2}\;\;O_{5,5}\;O_{3,5}\;\;O_{5,2}\;O_{4,2}]\;\;\right],$$

$$J_4^3=\left[\left[J'^3_{4,1}\right]\;\left[J'^3_{4,2}\right]\right]=\left[\,[O_{3,4}\;O_{4,4}\;O_{5,4}][\varnothing]\,\right],J_5^3=\left[\left[J'^3_{5,1}\right]\;\left[J'^3_{5,2}\right]\right]\left[\,[O_{2,5}\;O_{1,5}\;][\varnothing]\right]$$

$$J_1^4 = \left[\left[J'^4_{1,1}\right]\ \left[J'^4_{1,2}\right]\right] = \left[[O_{4,1}\ O_{5,1}\ O_{3,1}]\ [O_{2,4}\ O_{1,4}]\ \right],$$
$$J_2^4 = \left[\left[J'^4_{2,1}\right]\ \left[J'^4_{2,2}\right]\right] = \left[[O_{3,3}\ O_{5,3}\ O_{4,3}][\varnothing]\right],$$

$$J_3^4 = \left[\left[J'^4_{3,1}\right]\ \left[J'^4_{3,2}\right]\right] = \left[[O_{2,1}\ O_{1,1}]\ [O_{3,2}\ O_{3,5}\ O_{5,2}\ O_{5,5}\ O_{4,2}\ O_{4,5}]\ \right],$$

$$J_4^4 = \left[\left[J'^4_{4,1}\right]\ \left[J'^4_{4,2}\right]\right] = \left[[O_{3,4}\ O_{4,4}\ O_{5,4}][\varnothing]\right], J_5^4 = \left[\left[J'^4_{5,1}\right]\ \left[J'^4_{5,2}\right]\right]\left[[O_{2,5}\ O_{1,5}\ ][\varnothing]\right]$$

$$J_1^5 = \left[\left[J'^5_{1,1}\right]\ \left[J'^5_{1,2}\right]\right] = \left[[O_{4,1}\ O_{5,1}\ O_{3,1}]\ [O_{2,4}\ O_{1,4}]\ \right],$$
$$J_2^5 = \left[\left[J'^5_{2,1}\right]\ \left[J'^5_{2,2}\right]\right] = \left[[O_{3,3}\ O_{5,3}\ O_{4,3}][\varnothing]\right],$$

$$J_3^5 = \left[\left[J'^5_{3,1}\right]\ \left[J'^5_{3,2}\right]\right] = \left[[O_{2,1}\ O_{1,1}]\ [O_{3,2}\ O_{5,2}\ O_{4,2}\ O_{3,5}\ O_{5,5}\ O_{4,5}]\ \right],$$

$$J_4^5 = \left[\left[J'^5_{4,1}\right]\ \left[J'^5_{4,2}\right]\right] = \left[[O_{3,4}\ O_{4,4}\ O_{5,4}][\varnothing]\right], J_5^5 = \left[\left[J'^5_{5,1}\right]\ \left[J'^5_{5,2}\right]\right]\left[[O_{2,5}\ O_{1,5}\ ][\varnothing]\right]$$

$$\begin{aligned} J_f = [&O_{2,1}\quad O_{1,1}\quad O_{3,2}\quad O_{5,2}\quad O_{4,2}\quad O_{3,5}\quad O_{5,5}\quad O_{4,5}\quad O_{2,5}\quad O_{1,5}\\ &O_{3,4}\quad O_{4,4}\quad O_{5,4}\quad O_{4,1}\quad O_{5,1}\quad O_{3,1}\quad O_{2,4}\quad O_{1,4}\quad O_{3,3}\quad O_{5,3}\quad O_{4,3}] \end{aligned}$$

Vector J_f indicates the sequence of operations in order to enter the schedule. To select an operation for scheduling based on J_f, precedence relationships of the operation should be considered. Operations are removed from J_f one by one and are added to the schedule. If an operation cannot go into the schedule because of the precedence constraints, it should be transferred to the end of the vector J_f to prevent an operation from processing between other different operations on a machine.

5.2.2 Hybrid WCA using mutation operators of GA

In order to increase exploration of the proposed WCA, the mutation operation of the GA is employed for K', J_k^2 and S'_k with the probability of P_m for each individual. Shifting genes' loci and shifting processing machines, explained in Sect. 5.1.4, are applied respectively for J_k^2, while one of the three popular operators, swap, reversion, and insertion, are used randomly for K' and S'_k. For J_k^2 (or S'_k), one machine is selected randomly, and the related vector has been mutated. Shifting genes' loci changes operation sequences in one of the randomly chosen vectors J_k^2 and shifting processing machines alter the processing machine of the same operations of the jobs in the same group with the probability of P_k. Indeed, it moves the same operations of the jobs in the same group between different vectors of J_k^2 where the order of operations is preserved. The mutation operators are applied *nm* times in each iteration. The following examples clarify how to use the aforementioned operators.

The mutation of K' (Insertion): $[M_3 \quad \overset{\overset{2}{\downarrow}}{M_5} \quad M_4 \quad M_1 \quad \overset{\overset{1}{\downarrow}}{M_2}] \rightarrow [M_3 \quad M_1 \quad M_5 \quad M_4 \quad M_2]$

The mutation of S_3' (Swap): $\left[S^1 \; S^2 \right] \rightarrow \left[S^2 \; S^1 \right]$

Shifting genes' loci on J_3^2: $[O_{4,5} \quad \boxed{O_{3,2}} \quad O_{5,5} \quad O_{2,1} \quad O_{3,5} \quad \boxed{O_{1,1}} \quad O_{5,2} \quad O_{4,2}] \rightarrow [O_{4,5} \quad O_{2,1} \quad O_{5,5} \quad O_{1,1} \quad O_{3,5} \quad O_{3,2} \quad O_{5,2} \quad O_{4,2}]$

Shifting processing machines on the fifth operation of the jobs in the second group where M_4 is selected instead of M_3: $J_3^2 = [\boxed{O_{4,5}} \quad O_{3,2} \quad \boxed{O_{5,5}} \quad O_{2,1} \quad \boxed{O_{3,5}} \quad O_{1,1} \quad O_{5,2} \quad O_{4,2}] \rightarrow [O_{3,2} \quad O_{2,1} \quad O_{1,1} \quad O_{5,2} \quad O_{4,2}]$

$J_5^2 = [O_{3,4} \quad O_{4,4} \quad O_{5,4}] \rightarrow [O_{3,4} \quad O_{4,4} \quad O_{5,4} \quad O_{4,5} \quad O_{5,5} \quad O_{3,5}]$

5.2.3 Movements of streams and rivers

The new position of the streams and rivers are determined by applying the Eqs. 44–46 on all the corresponding random matrixes as follows ($C = 2$):

$$X^{Stream(t+1)} = X^{Stream(t)} + 2 \times rand \times \left(X^{River(t)} - X^{Stream(t)} \right) \tag{50}$$

$$Z_k^{Stream(t+1)} = Z_k^{Stream(t+1)} + 2 \times rand \times \left(Z_k^{River(t)} - Z_k^{Stream(t)} \right) \quad \forall k = 1, 2, \ldots, m \tag{51}$$

$$J_k^{Stream(t+1)} = J_k^{Stream(t+1)} + 2 \times rand \times \left(J_k^{River(t)} - J_k^{Stream(t)} \right) \quad \forall k = 1, 2, \ldots, m \tag{52}$$

$$S_k^{Stream(t+1)} = S_k^{Stream(t+1)} + 2 \times rand \times \left(S_k^{River(t)} - S_k^{Stream(t)} \right) \quad \forall k = 1, 2, \ldots, m \tag{53}$$

$$K^{Stream(t+1)} = K^{Stream(t)} + 2 \times rand \times \left(K^{River(t)} - K^{Stream(t)} \right) \tag{54}$$

For the movement of streams and rivers to the sea, the same equations are utilized.

5.2.4 Evaporation condition

To check the evaporation condition, the summation of the 2-norm of all random matrixes related to an individual is used. The final condition for streams and the sea would be as follows:

$$\begin{aligned} &\|X^{Sea} - X^{Stream_i}\|_2 + \sum_{k=1}^{m} \|Z_k^{Sea} - Z_k^{Stream_i}\|_2 + \sum_{k=1}^{m} \|J_k^{Sea} - J_k^{Stream_i}\|_2 \\ &+ \sum_{k=1}^{m} \|S_k^{Sea} - S_k^{Stream_i}\|_2 + \|K^{Sea} - K^{Stream_i}\|_2 < d_{max} i = 1, 2, \ldots, NS_1 \end{aligned} \tag{55}$$

The same condition can be applied for streams and rivers.

6 Managerial insights

As mentioned in Sect. 1, the IPPGS problem appears in manufacturing systems where setup times are significantly high relative to processing times. To clarify why considering group processing in IPPS problems is beneficial, a small-size IPPGS problem instance with low flexibility as delineated in Fig. 1, as well as sequence-dependent setup times reported in Table 4, is considered.

The illustrated problem is solved using combination-based MILP with an optimum makespan of 69 in 0.73 s. As a second scenario, we ignore job grouping and therefore release the group technology constraint. Indeed, an IPPS problem, including five jobs with SDST, would be available, whereas the setup times between jobs 1 and 2 on the one hand and also between jobs 3, 4, and 5, on the other hand, are equal to zero. Other characteristics remain unchanged. Solving the newly generated problem, the optimum makespan is equal to 69 as achieved before, but with the CPU time of 128 s, i.e., more than 175 times larger than the IPPGS scenario. In other words, considering group processing reduces the solution time by more than 99%.

It should be noted that such a significant difference in the computational time exists just for a small and simple problem. The difference will increase drastically when a medium, large, or more complex problem is under consideration. As a result, job grouping in the IPPS environment enables achieving an optimum or a proper solution faster and handling larger and more complex problems.

7 Computational results

In this section, the performance of the proposed metaheuristic algorithms is evaluated. As mentioned in the previous sections, the IPPGS problem has not been studied before. Then there are no related benchmark problems in the literature. Therefore, 45 IPPGS problem instances are generated in different sizes and flexibility levels based on 18 Kim's benchmark jobs (Kim et al. 2003) with 15 machines to evaluate the performance of the algorithms. The problems are categorized into three sizes: small (with two groups of jobs), medium (with four groups of jobs), and large (with six groups of jobs). In addition, five levels of process flexibility (PF), sequence

Table 4 Sequence-dependent setup times for the problem related to Fig. 1

Machines	From groups	To groups		Machines	From groups	To groups	
		1	2			1	2
1	Dummy	35	30	3	Dummy	30	27
	1	0	45		1	0	25
	2	31	0		2	37	0
2	Dummy	37	32	4	Dummy	38	32
	1	0	33		1	0	30
	2	34	0		2	26	0

flexibility (SF), and operation flexibility (OF) are considered based on Kim's benchmark categorization, i.e., low, medium, high, low-medium, and medium–high. For example, L-LowSF is a large instance, including jobs with a low level of sequence flexibility, and M-MediumHighPF is a medium one, including jobs with both medium and high process flexibility. The measures introduced by Kim et al. (2003) are used to categorize three kinds of flexibility. The classification basis is shown in Table 5.

Two to five jobs in each group have been taken into account. The selected AND/OR graph from Kim's Jobs, as well as the number of jobs in each group, are listed in Table 6. Also, Table 7 accounts for some general information for the benchmark problems, including the number of groups, the number of jobs, the maximum number of operations (NO), the maximum number of combinations (NC), and the total number of operations (TNO) of all jobs. The processing and setup times are randomly generated from uniform distributions between (1,20) and (20,50), respectively. The algorithms are implemented in Matlab 9.4 and run on an Intel Core i5-3470@3.20 GHz with 8 GB RAM PC. The parameters of both algorithms are set, as shown in Table 8.

In order to compare the performance of the proposed algorithms in the same situation, a limited CPU time set to 2× the number of jobs × the number of machines is used as a stopping criterion for both metaheuristic algorithms under consideration. In this way, large problems have more time than medium and small ones, and similarly, medium problems in comparison with small ones. The experiment for each algorithm is repeated 10 times for every problem instance, and the best solution of each run is recorded. On the other hand, each benchmark problem is executed by the combination-based mathematical model using Cplex solver (version 12.8) within the GAMS (version 25.0.2) environment with a limitation of 2 h for Cplex execution time, and the best-found integer value of the objective function (BF) is reported for the exact method. It is evident that BF can be an optimum solution. Table 9 shows BF as well as the mean and the standard deviation of the solutions yielded by GA and HWCA for each problem instance.

Due to increased complexity, the exact method found the optimum solution for only 4 problems, whilst for 18 of 45 problems, it could not reach any feasible solution after 2 h. However, the best solution can be found by the exact method for 15 problems, including 14 small-size instances and one medium-size instance (problem 2). On the other hand, the best-found solution for all small-size instances except one (problem 28) can be achieved by the exact method. As a result, although the exact method is unable to reach the optimum solution in most cases, its output is better than or equal to two metaheuristic algorithms for 14 of

Table 5 Classification basis according to flexibility indices

Classification	PF Index	SF Index	OF Index
Low	$[1, 2]$	$(0, 0.4)$	$(1, 2)$
Medium	$[3, 5]$	$(0.4, 0.7)$	$(2, 3.5)$
High	≥ 6	$(0.7, 1)$	≥ 3.5

Table 6 Specification of benchmark problems

No	Problem	The selected AND/OR graph for each group – the number of jobs of each graph																	
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	S-LowPF	2									4								
2	M-LowPF	3	4								2	5							
3	L-LowPF	3	3	5							2	4	4						
4	S-MediumPF				4									2					
5	M-MediumPF				5	3								2	4				
6	L-MediumPF				3	5	4							4	4	3			
7	S-HighPF							4										2	
8	M-HighPF							3	5								4	2	
9	L-HighPF							4	2	5							5	3	3
10	S-LowSF				2			3											
11	M-LowSF	4			4			4			3								
12	L-LowSF	3			5			3			4			2			4		
13	S-MediumSF		4												3				
14	M-MediumSF		3			5						4			3				
15	L-MediumSF		4			3			4			5			4			3	
16	S-HighSF			3									3						
17	M-HighSF			4			4						5			2			
18	L-HighSF			5			3			4			4			3			3
19	S-LowOF	4							2										
20	M-LowOF	2			3				5				5						
21	L-LowOF	3			5				4				3			4		3	
22	S-MediumOF		4								4								
23	M-MediumOF		3					3			5				4				
24	L-MediumOF		2				5	4			4				3				3

Table 6 (continued)

No	Problem	The selected AND/OR graph for each group – the number of jobs of each graph																	
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
25	S-HighOF					4								2					
26	M-HighOF			5		4						4		3					
27	L-HighOF			3		5				3		3		4			4		
28	S-LowMediumPF					3					4								
29	M-LowMediumPF		3	5	3									4					
30	L-LowMediumPF	4					5					5	3		3	3			
31	S-MediumHighPF					4				3									
32	M-MediumHighPF				5			3						3			4		
33	L-MediumHighPF						4		3						5	4		3	3
34	S-LowMediumSF					3		3											
35	M-LowMediumSF		4		5									2	4				
36	L-LowMediumSF	3							5		5	3					3	4	
37	S-MediumHighSF						4					2							
38	M-MediumHighSF			5		3									4	4			
39	L-MediumHighSF		4						5	3			4					2	4
40	S-LowMediumOF							2								4			
41	M-LowMediumOF				4		5				3		3						
42	L-LowMediumOF	4	5						3						4			4	3
43	S-MediumHighOF			4							3								
44	M-MediumHighOF					3		5							2		5		
45	L-MediumHighOF		4				4			3		5		3					4

Table 7 General information for benchmark problems

No	Problem	Number of groups	Number of jobs	Max. NO	Max. NC	TNO
1	S-LowPF	2	6	11	2	60
2	M-LowPF	4	14	14	2	147
3	L-LowPF	6	21	19	2	291
4	S-MediumPF	2	6	18	5	100
5	M-MediumPF	4	14	18	5	222
6	L-MediumPF	6	23	20	5	387
7	S-HighPF	2	6	22	12	128
8	M-HighPF	4	14	22	12	291
9	L-HighPF	6	22	22	12	446
10	S-LowSF	2	5	21	9	95
11	M-LowSF	4	15	21	9	213
12	L-LowSF	6	21	21	9	331
13	S-MediumSF	2	7	14	4	95
14	M-MediumSF	4	15	18	4	207
15	L-MediumSF	6	23	22	12	353
16	S-HighSF	2	6	19	2	111
17	M-HighSF	4	15	20	4	276
18	L-HighSF	6	22	20	8	403
19	S-LowOF	2	6	20	10	72
20	M-LowOF	4	15	20	10	254
21	L-LowOF	6	22	22	12	364
22	S-MediumOF	2	8	14	2	100
23	M-MediumOF	4	15	21	9	212
24	L-MediumOF	6	21	21	9	346
25	S-HighOF	2	6	18	5	108
26	M-HighOF	4	16	19	5	257
27	L-HighOF	6	22	21	8	390
28	S-LowMediumPF	2	7	18	3	98
29	M-LowMediumPF	4	15	19	5	257
30	L-LowMediumPF	6	23	20	4	315
31	S-MediumHighPF	2	7	20	8	132
32	M-MediumHighPF	4	15	21	9	281
33	L-MediumHighPF	6	22	22	12	382
34	S-LowMediumSF	2	6	21	9	117
35	M-LowMediumSF	4	15	18	5	224
36	L-LowMediumSF	6	23	22	12	357
37	S-MediumHighSF	2	6	20	3	98
38	M-MediumHighSF	4	16	19	4	261
39	L-MediumHighSF	6	22	22	12	400
40	S-LowMediumOF	2	6	21	9	102
41	M-LowMediumOF	4	15	20	4	251
42	L-LowMediumOF	6	23	22	12	353

Table 7 (continued)

No	Problem	Number of groups	Number of jobs	Max. NO	Max. NC	TNO
43	S-MediumHighOF	2	7	19	2	109
44	M-MediumHighOF	4	15	21	9	290
45	L-MediumHighOF	6	23	20	8	363

15 small-size problems. The metaheuristic algorithms have better performance for medium and large-size problems.

Figure 9 depicts the superiority of the solution methods for finding the best-found solution in terms of the size of the problems. As shown in Fig. 9, the best-found makespan is achieved by GA for 4 medium problems. However, HWCA is more capable of reaching a better solution for more complex and real-world problems. The best-found solution for all large problems and most medium problems (10 of 15) has been obtained by HWCA. Additionally, the flexibility is relatively low for five cases in which the best-found solution of the exact method or GA is better than HWCA.

The "improved rate" (IR rate), reported in Table 9, indicates the relative improved ratio of the HWCA result compared to the minimum result obtained by GA or the exact method. HWCA presents the best result where the improved rate is positive. It is shown the improvement for 17 of 18 medium and large-size problems with medium or high flexibility. For large-size problems, improved ratios are more than 5% when a flexibility factor is medium or high, and they are more than 9% when flexibility is high for all jobs. Figure 10 gives the Gantt chart of the last instance.

To examine the evolution trend of GA and HWCA, 6 problems 1, 2, 3, 7, 8, and 9 with different sizes and flexibility levels and also different search spaces are chosen. Figure 11 illustrates search capability within the limited computational time for the selected benchmark problems. As the figure shows, GA presents a better solution in the early stages of processing for all problems. In other words, GA tends to converge more quickly than HWCA. Combining genetic and WCA operators in HWCA increases more reproduced individuals and consequently more required computational time at each iteration. On the other hand, it causes a powerful and broad

Table 8 The parameter setting of the proposed algorithms

GA	HWCA
$N_{pop} = 150$	$N_{pop} = 90$
$P_m = 0.8$	$P_m = 0.8$
$P_k = 0.6$	$P_k = 0.6$
$P_c = 0.6$	$N_{sr} = 5$
$P_h = 0.08$	$d_{max} = 1e-4$
tournament size = 15	$nm = 10$
$max\_iteration = 100$	$max\_iteration = 200$

Table 9 Comparison of makespan for the exact method and two proposed algorithms

No	Problem	Exact	GA			HWCA			Best***	IR rate (%)
			BF	Mean	SD	BF	Mean	SD		
1	S-LowPF	169*	169*	171.4	1.59	169*	170.2	0.95	Exact	−0.71
2	M-LowPF	236	240	257.4	10.61	242	247.6	4.10	Exact	−4.92
3	L-LowPF	**	722	749.5	15.90	700	721.6	11.08	HWCA	3.72
4	S-MediumPF	178	180	188.6	4.98	180	183.6	3.31	Exact	−3.15
5	M-MediumPF	313	296	318.5	9.65	258	281	11.72	HWCA	10.22
6	L-MediumPF	**	606	646.3	18.25	601	613.6	8.24	HWCA	5.06
7	S-HighPF	190	190	200.6	5.83	194	200.4	4.33	Exact	−5.47
8	M-HighPF	**	328	344.8	7.99	317	323.6	4.40	HWCA	6.15
9	L-HighPF	**	662	706.3	20.75	600	627.1	13.78	HWCA	11.21
10	S-LowSF	166*	172	175.6	2.29	168	174.8	3.48	Exact	−5.30
11	M-LowSF	304	282	309.3	12.18	286	298	10.00	HWCA	1.97
12	L-LowSF	**	514	547.8	18.05	487	516	12.43	HWCA	5.81
13	S-MediumSF	195	195	204.1	4.24	201	204.3	2.16	Exact	−4.77
14	M-MediumSF	245	227	244.7	11.77	203	222.6	11.16	HWCA	9.03
15	L-MediumSF	**	561	625.8	19.14	536	552.2	16.40	HWCA	11.76
16	S-HighSF	223	223	232.7	3.53	229	232.9	2.48	Exact	−4.44
17	M-HighSF	**	669	694.3	13.07	666	674.8	6.38	HWCA	2.81
18	L-HighSF	**	866	904.2	21.23	757	777.4	11.47	HWCA	14.02
19	S-LowOF	155*	162	170.7	4.66	158	163.6	4.31	Exact	−5.55
20	M-LowOF	370	369	392.3	10.83	378	391.6	7.24	Exact	−5.84
21	L-LowOF	**	660	738.3	20.33	621	652.6	15.05	HWCA	11.61
22	S-MediumOF	137*	137*	146.5	4.91	137*	142.4	4.40	Exact	−3.94

Table 9 (continued)

No	Problem	Exact	GA			HWCA			Best***	IR rate (%)
			BF	Mean	SD	BF	Mean	SD		
23	M-Medium-mOF	271	282	310.5	10.57	266	278	7.59	Exact	−2.58
24	L-Mediu-mOF	**	656	687.6	19.06	610	638.1	15.79	HWCA	7.20
25	S-HighOF	137	142	151.8	6.02	142	145	1.92	Exact	−5.84
26	M-HighOF	**	318	329.4	7.09	305	317.1	5.39	HWCA	3.73
27	L-HighOF	**	650	680.1	18.35	608	617.9	9.49	HWCA	9.15
28	S-LowMedi-umPF	148	147	157.3	4.44	152	156.7	2.38	Exact	−5.88
29	M-LowMedi-umPF	398	329	348.5	15.37	336	350.9	11.03	GA	−0.69
30	L-LowMedi-umPF	**	647	673.6	18.49	631	648.5	12.41	HWCA	3.73
31	S-Medium-HighPF	193	205	210.8	3.93	199	201.1	1.71	Exact	−4.20
32	M-Medium-HighPF	**	457	475	9.52	435	449	8.75	HWCA	5.47
33	L-Medium-HighPF	**	661	694.7	17.38	607	630.4	13.79	HWCA	9.26
34	S-LowMedi-umSF	166	166	171.4	2.69	169	172.3	2.33	Exact	−3.80
35	M-LowMedi-umSF	301	295	315.7	10.45	308	316.8	7.12	Exact	−5.25
36	L-LowMedi-umSF	**	512	544.1	17.22	511	527.2	10.08	HWCA	3.11

Table 9 (continued)

No	Problem	Exact	GA			HWCA			Best***	IR rate (%)
			BF	Mean	SD	BF	Mean	SD		
37	S-Medium-HighSF	194	201	207.8	4.13	201	204.7	2.45	Exact	−5.52
38	M-Medium-HighSF	416	392	413.9	12.98	374	395	10.34	HWCA	4.57
39	L-Medium-HighSF	**	758	787.4	17.02	717	742.3	11.32	HWCA	5.73
40	S-LowMedi-umOF	180	180	186.8	5.71	184	187.2	3.15	Exact	−4.00
41	M-LowMedi-umOF	407	386	403.1	8.87	368	388.7	9.12	HWCA	3.57
42	L-LowMedi-umOF	853	548	578.1	18.64	524	546	11.86	HWCA	5.55
43	S-Medium-HighOF	222	238	244.2	3.90	231	235	3.73	Exact	−5.86
44	M-Medium-HighOF	471	322	355.7	13.37	286	306	9.96	HWCA	13.97
45	L-Medium-HighOF	**	724	757.9	21.35	663	689	11.47	HWCA	9.09

*An optimum solution

**A feasible solution is not available after 2 h

***According to the mean solution

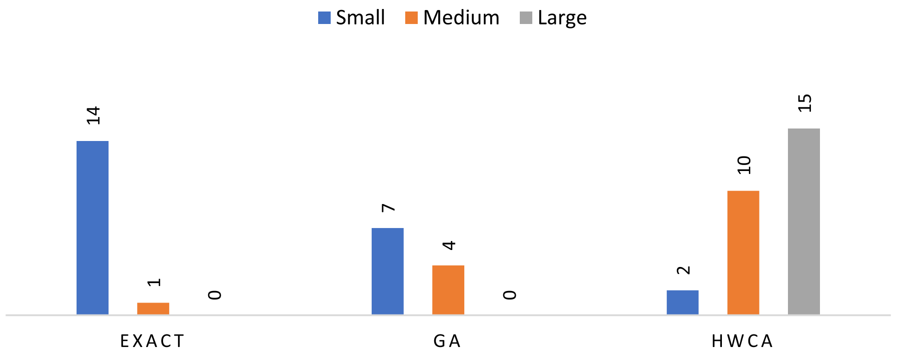


Fig. 9 The superiority of the solution methods to achieve the best-found solution

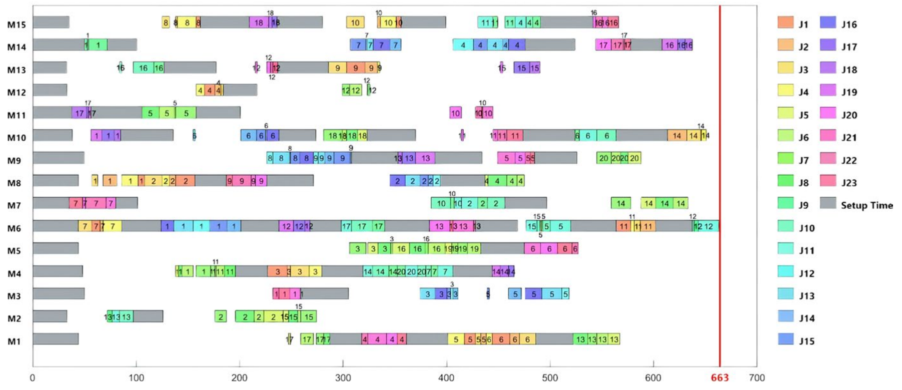
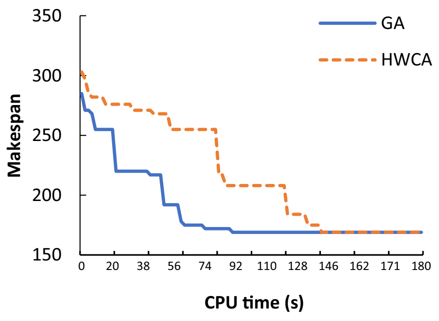


Fig. 10 The Gantt chart of the last instance

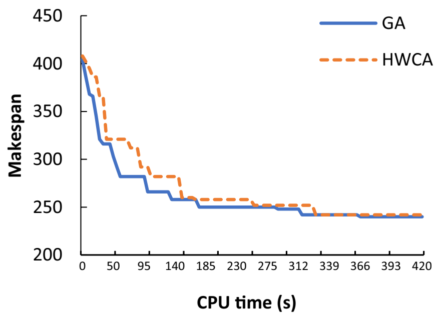
exploration in HWCA as a more qualified solution can be achieved when more time is available.

8 Conclusions

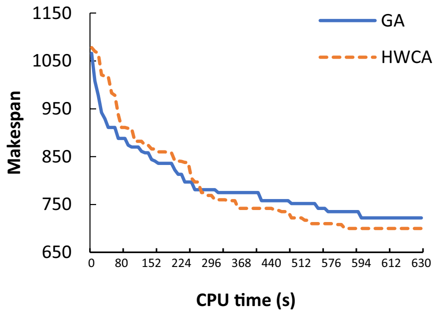
In this research, two efficient MILP models for the integrated process planning and group scheduling (IPPGS) problem with sequence-dependent setup time (SDST) are proposed. The models were formulated based on different approaches called process plan-based and combination-based models. The combination-based mathematical model has superior performance compared to the other one. However, it can only be applied for small-size problems with a low level of flexibility. This mathematical model can be used in future research to provide high-quality lower bounds for the proposed problem. Also, two metaheuristic algorithms are proposed to provide an upper bound for the IPPGS problem. The first one is a genetic algorithm with a new two-section representation, including a



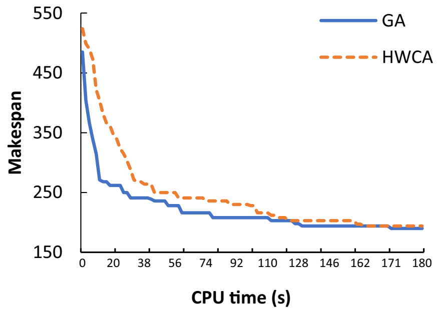
(a) Problem 1



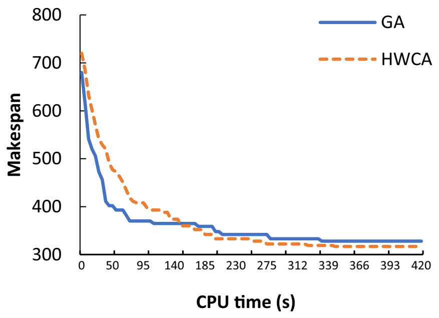
(b) Problem 2



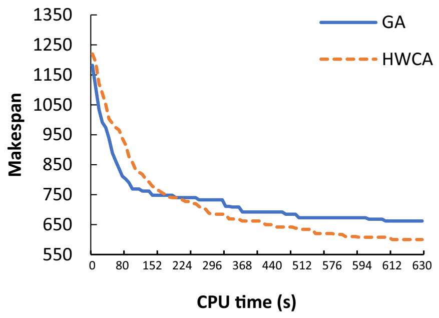
(c) Problem 3



(d) Problem 7



(e) Problem 8



(f) Problem 9

Fig. 11 Comparison of search capability within the limited computational time

numerical and an object-coding section. Genetic operators are customized to cope with the new genetic representation. The other optimization method is a hybrid water cycle algorithm (HWCA). The naturally continuous WCA was developed to accommodate the IPPGS problem using a discrete strategy. Moreover, in order to increase the exploration of the proposed WCA, the mutation operation of the GA is employed. Although the experimental results indicate that HWCA provides better outcomes than GA, but due to faster convergence in the early stages of the search, GA presents better results in a short time.

For future research, a lower bound can be provided for the proposed problem by applying the proposed combination-based model as well as the related techniques such as Lagrangian relaxation or branch-and-price. Furthermore, the IPPGS problem can be studied using total tardiness as an objective function. In addition, the proposed algorithms can be extended to cope with the IPPGS problem for other shop environments, such as assembly operations.

Funding Not applicable.

Availability of data and material The data that support the findings of this study are available on request from the corresponding author.

Code availability Algorithms are implemented in Matlab 9.4 while mathematical models are solved by Cplex solver (version 12.8) within the GAMS (version 25.0.2) environment. All related codes are available on request from the corresponding author.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

Allahverdi A (2015) The third comprehensive survey on scheduling problems with setup times/costs. Eur J Oper Res 246:345–378. https://doi.org/10.1016/j.ejor.2015.04.004

Allahverdi A, Gupta J, Aldowaisan T (1999) A review of scheduling research involving setup considerations. Omega. https://doi.org/10.1016/S0305-0483(98)00042-5

Allahverdi A, Ng CT, Cheng TCE, Kovalyov MY (2008) A survey of scheduling problems with setup times or costs. Eur J Oper Res 187:985–1032. https://doi.org/10.1016/j.ejor.2006.06.060

Altarazi S, Yasin O (2015) Integration of process planning and scheduling with sequence dependent setup time: a case study from electrical wires and power cable industry. In, Cham, 2015. Modelling, computation and optimization in information systems and management sciences. Springer International Publishing, pp 283–294

Amin-Naseri MR, Afshari AJ (2012) A hybrid genetic algorithm for integrated process planning and scheduling problem with precedence constraints. Int J Adv Manuf Technol 59:273–287. https://doi.org/10.1007/s00170-011-3488-y

Barzanji R, Naderi B, Begen MA (2019) Decomposition algorithms for the integrated process planning and scheduling problem. Omega 93:102025

Behjat S, Salmasi N (2017) Total completion time minimisation of no-wait flowshop group scheduling problem with sequence dependent setup times. Eur J Indust Eng 11:22–48. https://doi.org/10.1504/ejie.2017.081418

Brandimarte P, Calderini M (1995) A hierarchical bicriterion approach to integrated process plan selection and job shop scheduling. Int J Prod Res 33:161–181. https://doi.org/10.1080/00207549508930142

Chan FTS, Kumar V, Tiwari MK (2009) The relevance of outsourcing and leagile strategies in performance optimization of an integrated process planning and scheduling model. Int J Prod Res 47:119–142. https://doi.org/10.1080/00207540600818195

Chryssolouris G, Chan S, Cobb W (1984) Decision making on the factory floor: an integrated approach to process planning and scheduling. Robot Comput-Integrat Manuf 1:315–319

Eskandar H, Sadollah A, Bahreininejad A, Hamdi M (2012) Water cycle algorithm—a novel metaheuristic optimization method for solving constrained engineering optimization problems. Comput Struct 110–111:151–166. https://doi.org/10.1016/j.compstruc.2012.07.010

Gao K, Duan P, Su R, Li J (2017) Bi-objective water cycle algorithm for solving remanufacturing rescheduling problem. In: Shi Y et al (eds) Simulated evolution and learning. Springer International Publishing, Cham, pp 671–683

Guo YW, Li W, Mileham AR, Owen GW (2009a) Applications of particle swarm optimisation in integrated process planning and scheduling. Robot Comput-Integrat Manuf 25:280–288

Guo YW, Li WD, Mileham AR, Owen GW (2009b) Optimisation of integrated process planning and scheduling using a particle swarm optimisation approach. Int J Prod Res 47:3775–3796. https://doi.org/10.1080/00207540701827905

Gupta JND, Darrow WP (1986) The two-machine sequence dependent flowshop scheduling problem. Eur J Oper Res 24:439–446

Ha C (2019) Evolving ant colony system for large-sized integrated process planning and scheduling problem considering sequence-dependent setup times. Flex Serv Manuf J. https://doi.org/10.1007/s10696-019-09360-9

Ham I, Hitomi K, Yoshida T (1985) Basic principles of group technology. In: Group technology: applications to production management. Springer Netherlands, Dordrecht, pp 7–20. https://doi.org/10.1007/978-94-009-4976-8_2

Hitomi K, Ham I (1976) Operations scheduling for group technology applications. Ann CIRP 25:419–422

Imanipour N (2006) Modeling and solving flexible job shop problem with sequence dependent setup times. In: 2006 International conference on service systems and service management, 25–27 Oct. 2006. pp 1205–1210. https://doi.org/10.1109/ICSSSM.2006.320680

Jafar RMS, Geng S, Ahmad W, Hussain S, Wang H (2018) A comprehensive evaluation: water cycle algorithm and its applications. In: Qiao J, Zhao X, Pan L, Zuo X, Zhang X, Zhang Q, Huang S (eds) Bio-inspired computing: theories and applications. Springer, Singapore, pp 360–376

Jin L, Zhang C, Shao X (2015) An effective hybrid honey bee mating optimization algorithm for integrated process planning and scheduling problems. Int J Adv Manuf Technol 80:1253–1264. https://doi.org/10.1007/s00170-015-7069-3

Jin L, Tang Q, Zhang C, Shao X, Tian G (2016a) More MILP models for integrated process planning and scheduling. Int J Prod Res 54:4387–4402. https://doi.org/10.1080/00207543.2016.1140917

Jin L, Zhang C, Shao X, Yang X, Tian G (2016b) A multi-objective memetic algorithm for integrated process planning and scheduling. Int J Adv Manuf Technol 85:1513–1528. https://doi.org/10.1007/s00170-015-8037-7

Keshavarz T, Salmasi N (2013) Makespan minimisation in flexible flowshop sequence-dependent group scheduling problem. Int J Prod Res 51:6182–6193. https://doi.org/10.1080/00207543.2013.825055

Khoshnevis B, Chen QM (1991) Integration of process planning and scheduling functions. J Intell Manuf 2:165–175

Kim KH, Egbelu PJ (1999) Scheduling in a production environment with multiple process plans per job. Int J Prod Res 37:2725–2753. https://doi.org/10.1080/002075499190491

Kim YK, Park K, Ko J (2003) A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling. Comput Oper Res 30:1151–1171

Kleinau U (1993) Two-machine shop scheduling problems with batch processing. Math Comput Model 17:55–66. https://doi.org/10.1016/0895-7177(93)90196-6

Lee HC, Ha C (2019) Sustainable integrated process planning and scheduling optimization using a genetic algorithm with an integrated chromosome representation. Sustainability 11:502

Lee H, Kim S-S (2001) Integration of process planning and scheduling using simulation based genetic algorithms. Int J Adv Manuf Technol 18:586–590. https://doi.org/10.1007/s001700170035

Leung CW, Wong TN, Mak K-L, Fung RY (2010) Integrated process planning and scheduling by an agent-based ant colony optimization. Comput Ind Eng 59:166–180

Li WD, McMahon CA (2007) A simulated annealing-based optimization approach for integrated process planning and scheduling. Int J Comput Integr Manuf 20:80–95

Li X, Gao L, Shao X, Zhang C, Wang C (2010a) Mathematical modeling and evolutionary algorithm-based approach for integrated process planning and scheduling. Comput Oper Res. https://doi.org/10.1016/j.cor.2009.06.008

Li X, Gao L, Zhang C, Shao X (2010b) A review on integrated process planning and scheduling. Int J Manuf Res 5:161–180

Li X, Shao X, Gao L, Qian W (2010c) An effective hybrid algorithm for integrated process planning and scheduling. Int J Prod Econ 126:289–298

Li X, Zhang C, Gao L, Li W, Shao X (2010d) An agent-based approach for integrated process planning and scheduling. Expert Syst Appl 37:1256–1264. https://doi.org/10.1016/j.eswa.2009.06.014

Li X, Gao L, Shao X (2012) An active learning genetic algorithm for integrated process planning and scheduling. Expert Syst Appl 39:6683–6691. https://doi.org/10.1016/j.eswa.2011.11.074

Li X, Gao L, Pan Q, Wan L, Chao K-M (2019a) An effective hybrid genetic algorithm and variable neighborhood search for integrated process planning and scheduling in a packaging machine workshop. IEEE Trans Syst Man Cybernet 49:1933–1945

Li X, Gao L, Wang W, Wang C, Wen L (2019b) Particle swarm optimization hybridized with genetic algorithm for uncertain integrated process planning and scheduling with interval processing time. Comput Ind Eng 135:1036–1046

Li WD, Gao L, Li XY, Guo Y (2008) Game theory-based cooperation of process planning and scheduling. In: 2008 12th international conference on computer supported cooperative work in design, 16–18 April 2008. pp 841–845. https://doi.org/10.1109/CSCWD.2008.4537088

Lian K, Zhang C, Gao L, Li X (2012a) Integrated process planning and scheduling using an imperialist competitive algorithm. Int J Prod Res 50:4326–4343. https://doi.org/10.1080/00207543.2011.622310

Lian K, Zhang C, Shao X, Gao L (2012b) Optimization of process planning with various flexibilities using an imperialist competitive algorithm. Int J Adv Manuf Technol 59:815–828

Lihong Q, Shengping L (2012) An improved genetic algorithm for integrated process planning and scheduling. Int J Adv Manuf Technol 58:727–740

Liu X, Ni Z, Qiu X (2016) Application of ant colony optimization algorithm in integrated process planning and scheduling. Int J Adv Manuf Technol 84:393–404. https://doi.org/10.1007/s00170-015-8145-4

Lu D, Logendran R (2013) Bi-criteria group scheduling with sequence-dependent setup time in a flow shop. J Oper Res Soc 64:530–546. https://doi.org/10.1057/jors.2012.61

Luo G, Wen X, Li H, Ming W, Xie G (2017) 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 91:3145–3158. https://doi.org/10.1007/s00170-017-0020-z

Mohapatra P, Nayak A, Kumar SK, Tiwari MK (2015) Multi-objective process planning and scheduling using controlled elitist non-dominated sorting genetic algorithm. Int J Prod Res 53:1712–1735. https://doi.org/10.1080/00207543.2014.957872

Morad N, Zalzala A (1999) Genetic algorithms in integrated process planning and scheduling. J Intell Manuf 10:169–179

Naderi B, Salmasi N (2012) Permutation flowshops in group scheduling with sequence-dependent setup times. Eur J Indust Eng 6:177–198

Nayak SK, Padhy SK, Panda CS (2018) Efficient multiprocessor scheduling using water cycle algorithm. In, Singapore, 2018. Soft Computing: Theories and Applications. Springer Singapore, pp 559–568

Neufeld JS, Gupta JND, Buscher U (2016) A comprehensive review of flowshop group scheduling literature. Comput Oper Res 70:56–74. https://doi.org/10.1016/j.cor.2015.12.006

Nourali S, Imanipour N, Shahriari MR (2012) A mathematical model for integrated process planning and scheduling in flexible assembly job shop environment with sequence dependent setup times vol 6.

Özgüven C, Özbakır L, Yavuz Y (2010) Mathematical models for job-shop scheduling problems with routing and process plan flexibility. Appl Math Model 34:1539–1548. https://doi.org/10.1016/j.apm.2009.09.002

Petrović M, Vuković N, Mitić M, Miljković Z (2016) Integration of process planning and scheduling using chaotic particle swarm optimization algorithm. Expert Syst Appl 64:569–588. https://doi.org/10.1016/j.eswa.2016.08.019

Potter MA, De Jong KA (1994) A cooperative coevolutionary approach to function optimization. In, Berlin, Heidelberg, Parallel problem solving from nature—PPSN III. Springer, Berlin, pp 249–257

Sadollah A, Eskandar H, Bahreininejad A, Kim JH (2015a) Water cycle algorithm for solving multi-objective optimization problems. Soft Comput 19:2587–2603. https://doi.org/10.1007/s00500-014-1424-4

Sadollah A, Eskandar H, Bahreininejad A, Kim JH (2015b) Water cycle algorithm with evaporation rate for solving constrained and unconstrained optimization problems. Appl Soft Comput 30:58–71. https://doi.org/10.1016/j.asoc.2015.01.050

Salmasi N, Logendran R, Skandari MR (2010) Total flow time minimization in a flowshop sequence-dependent group scheduling problem. Comput Oper Res 37:199–212

Shahvari O, Salmasi N, Logendran R, Abbasi B (2012) An efficient tabu search algorithm for flexible flow shop sequence-dependent group scheduling problems. Int J Prod Res 50:4237–4254. https://doi.org/10.1080/00207543.2011.604051

Shao X, Li X, Gao L, Zhang C (2009) Integration of process planning and scheduling—a modified genetic algorithm-based approach. Comput Oper Res 36:2082–2096

Shukla SK, Tiwari MK, Son YJ (2007) Bidding-based multi-agent system for integrated process planning and scheduling: a data-mining and hybrid tabu-SA algorithm-oriented approach. Int J Adv Manuf Technol 38:163. https://doi.org/10.1007/s00170-007-1087-8

Sobeyko O, Mönch L (2016) Integrated process planning and scheduling for large-scale flexible job shops using metaheuristics. Int J Prod Res. https://doi.org/10.1080/00207543.2016.1182227

Tan W, Khoshnevis B (2004) A linearized polynomial mixed integer programming model for the integration of process planning and scheduling. J Intell Manuf 15:593–605. https://doi.org/10.1023/B:JIMS.0000037710.80847.b6

Uslu MF, Uslu S, Bulut F (2018) An adaptive hybrid approach: combining genetic algorithm and ant colony optimization for integrated process planning and scheduling. Appl Comput Inform. S2210832718300310

Wan S, Wong TN, Zhang S, Zhang L (2011) Integrated process planning and scheduling with setup time consideration by ant colony optimization.

Wang L, Shen W (2003) DPP: an agent-based approach for distributed process planning. J Intell Manuf 14:429–439. https://doi.org/10.1023/a:1025797124367

Wong TN, Leung CW, Mak KL, Fung RYK (2006a) An agent-based negotiation approach to integrate process planning and scheduling. Int J Prod Res 44:1331–1351. https://doi.org/10.1080/00207540500409723

Wong TN, Leung CW, Mak KL, Fung RYK (2006b) Integrated process planning and scheduling/rescheduling—an agent-based approach. Int J Prod Res 44:3627–3655. https://doi.org/10.1080/00207540600675801

Zhang S, Wong TN (2014) Integrated process planning and scheduling: an enhanced ant colony optimization heuristic with parameter tuning. J Intell Manuf 29:585–601. https://doi.org/10.1007/s10845-014-1023-3

Zhang L, Wong TN (2015a) An object-coding genetic algorithm for integrated process planning and scheduling. Eur J Oper Res 244:434–444. https://doi.org/10.1016/j.ejor.2015.01.032

Zhang S, Wong TN (2015b) Studying the impact of sequence-dependent set-up times in integrated process planning and scheduling with E-ACO heuristic. Int J Prod Res. https://doi.org/10.1080/00207543.2015.1098786

Zhang L, Wong TN (2016) Solving integrated process planning and scheduling problem with constructive meta-heuristics. Inf Sci 340–341:1–16. https://doi.org/10.1016/j.ins.2016.01.001

Zhang C, Li P, Rao Y, Li SA (2005) New hybrid GA/SA algorithm for the job shop scheduling problem. Berlin, Heidelberg, Evolutionary computation in combinatorial optimization. Springer, Berlin, pp 246–259

Zhang W, Gen M, Jo J (2014) Hybrid sampling strategy-based multiobjective evolutionary algorithm for process planning and scheduling problem. J Intell Manuf 25:881–897. https://doi.org/10.1007/s10845-013-0814-2

Zhang R, Ong SK, Nee AYC (2015) A simulation-based genetic algorithm approach for remanufacturing process planning and scheduling. Appl Soft Comput 37:521–532. https://doi.org/10.1016/j.asoc.2015.08.051

Zhao F, Hong Y, Yu D, Yang Y, Zhang Q (2010) A hybrid particle swarm optimisation algorithm and fuzzy logic for process planning and production scheduling integration in holonic manufacturing systems. Int J Comput Integr Manuf 23:20–39. https://doi.org/10.1080/09511920903207472

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.