# Decomposition algorithms for the integrated process planning and scheduling problem

R. Barzanji[a], Bahman Naderi[a,b,1], Mehmet A. Begen[b]

[a]*Department of Industrial Engineering, Faculty of Engineering, Kharazmi University, Tehran, Iran*
[b]*Department of Management Science, Ivey Business School, Western University of Ontario, Canada*

**Abstract**
There are several algorithms to solve the integrated process planning and scheduling (IPPS) problem (i.e., flexible job shop scheduling with process plan flexibility) in the literature. All the existing algorithms for IPPS are heuristic-based search methods and no research has investigated the use of exact solution methods for this problem. We develop several decomposition approaches based on the logic-based Benders decomposition (LBBD) algorithm. Our LBBD algorithm allows us to partition the decision variables in the IPPS problem into two models, master-problem and sub-problem. The master-problem determines process plan and operation-machine assignment, while the sub-problem optimizes sequencing and scheduling decisions. To achieve faster convergence, we develop two relaxations for the optimal makespan objective function and incorporate them into the master-problem. We analyze the performance and further enhance the algorithm with two ideas, a Benders optimality cut based on the critical path and a faster heuristic way to solve the sub-problem. 16 standard benchmark instances available in the literature are solved to evaluate and compare the performances of our algorithms with those of the state-of-the-art methods in the literature. The proposed algorithm either results in the optimal solution or improves the best-known solutions in all the existing instances, demonstrating its superiority to the existing state-of-the-art methods in literature.

**Keywords**: Integrated process planning and scheduling problem, Benders decomposition, optimality cut, critical path.

## 1. Introduction

Effective process planning and scheduling are crucial for the productivity of any manufacturing system. A process plan specifies raw materials and parts, develops production plans and determines processes and operations that convert raw materials to final goods. The outcome of process planning includes configuration of suitable machines, tools, and operations arrangement for a product. Process planning connects product design to manufacturing. On the other hand, scheduling specifies both operation-machine assignment and operation sequence for a given

[1] Corresponding author: Bahman Naderi (bahman.naderi@khu.ac.ir)

process plan of products including precedence relationships among operations. Therefore, scheduling is the link between processes preparation and putting them into action.

Although there is a close relationship between process planning and scheduling, the integration of them is still a challenge for both researchers and practitioners (Sugimura et al., 2001). In classical approaches, process planning and scheduling decisions are carried out sequentially, where scheduling is conducted separately after process plans are fixed. Due to recent advances in manufacturing systems and increase in overall flexibility, an increased number of process plans can be used to manufacture a product. Therefore, the integration of process planning and scheduling has become even more important to improve productivity of modern manufacturing systems. The trend in recent publications in production planning related fields shows that academia follows this integration and becomes more attracted to study the integrated process planning and scheduling (IPPS) system (Lee and Kim, 2001; Li et al., 2012).

The IPPS problem belongs to the class of NP-hard problems (Khoshnevis and Chen, 1991; Kis, 2003). The standard scheduling methods have difficulty in solving the IPPS problem. Over the last two decades, researchers have used various approaches to tackle the IPPS problem. Although, as it is reviewed in the next section, there are several papers dealing with the IPPS problem, they all develop solution approaches based on heuristic search methods (i.e., constructive heuristics and metaheuristics).

To the best of our knowledge, there is still no solution algorithm based on exact methods. This paper proposes an effective algorithm based on logic-based Benders decomposition to solve the IPPS problem. This is the first application of the Benders decomposition algorithm for the IPPS problem. The original problem is divided into two smaller problems: master-problem and sub-problem. An optimality cut is generated to exchange information between the two smaller problems. Then we develop two expediting procedures (a Benders optimality cut based on the critical path, and a faster heuristic way to solve the sub-problem) to further enhance the algorithm. Our proposed algorithm is evaluated by comparing its performance with performances of existing solution methods from the literature over standard benchmark sets. The experiments show the superiority of our algorithm: It either results in the optimal solutions or improves the best-known solutions for all the benchmark instances. The proposed algorithm can obtain small optimality gaps in short computational time. Hence, it provides the state-of-the-art results for the IPPS problem.

In addition to scientific novelties, we present an effective method that is applicable to manufacturing industries with type-1 IPPS specifications. The proposed method can help companies to reduce idle time of resources and increase overall productivity. Productivity is one of the most important performance measures in all manufacturing systems, and makespan scheduling is known to be an effective tool in this regard (Pinedo, 2008; Seidel and Arndt, 1998). Makespan scheduling ensures higher resource utilization with zero-dollar added cost, making makespan minimization as inevitable part of every production schedule management. To achieve minimum makespan, unnecessary idle time of resources needs to be reduced as much as possible. It is well-known that resources in manufacturing systems are cost-intensive, and every single unit of idle time of resources corresponds to waste of resource (i.e., capital). Thus, removing unnecessary idle times result in direct savings for manufacturers.

The rest of this paper is organized as follows. Section 2 provides the literature review of the IPPS. The mathematical definition and formulation of the IPPS are presented in Section 3. Section 4 develops our logic-based Benders decomposition algorithm. We evaluate the performance of the algorithm Section 5, and Section 6 concludes the paper.

## 2. Literature reviews

Process planning and scheduling are two influential and complementary components of any manufacturing system. There are three decisions in the IPPS problems: (i) process plan selection, (ii) machine assignment, and (iii) scheduling.  The traditional idea is to solve the problem by a two-phase approach: select the process plan first and then assign and schedule operations accordingly (Weintraub et al., 1999; Li et al., 2010). That is, a process plan for each job is determined and then jobs with fixed process plans are scheduled. Chryssolouris et al. (1984) discuss the flexibility coming out of the integration of process planning and scheduling. Moreover, Li et al., (2010b) shows that the integrated approaches are more effective.

The IPPS problems are either type-1 or type- 2 (Jin et al. [1]). In type-1, there is a set of process plans predetermined in advance for each job, and the decision is to select one process plan for each job. In type- 2, each job is represented by a graph with precedence relations among the operations. Thus, jobs' process plans are inferred from this graph.

Applications of IPPS problems are discussed in Liu et al. (2016) and Petrović et al. (2016). Based on real case examples, there are different standard benchmark sets in the literature. For example, Altarazi, and Yasin (2015) describe an example form electrical wires and power cable industry. Li et al. (2002) present different examples form machinery and manufacturing. The early benchmark sets commonly include small size instances (Nasr and Elsayed, 1990; Hoitomt et al., 1993; Lee and Dicesare, 1994; Sundaram and Fu, 1988). The more recent benchmark sets include larger instances (Dong and Sun, 2006; Jaint et al., 2006, Kim et al., 2003; Chan et al., 2012). Table1 summarizes the available benchmark sets for type-1 and type-2 IPPS problems. *N* and *M* are the numbers of jobs and machines, respectively. Benchmark sets B1-B7 include relatively smaller instances with less than 8 jobs while benchmark sets B8-B11 consist of larger instances up to 100 jobs. Benchmark set B10 is also designed for the type-2 IPPS and the other benchmark sets for the type-1 IPPS.

Table 1. Benchmark sets for IPPSs

| Benchmark | Publication | Size | | No. | IPPS Type |
|---|---|---|---|---|---|
| | | *N* | *M* | | |
| B1 | Nasr and Elsayed (1990) | 4 | 6 | 1 | 1 |
| B2 | Sundaram and Fu (1988) | 5 | 3 | 1 | 1 |
| B3 | Lee and Dicesare (1994) | 5 | 3 | 1 | 1 |
| B4 | Moon et al. (2008) | 5 | 5 | 1 | 1 |
| B5 | Li et al. (2010b) | 6 | 5 | 2 | 1 |
| B6 | Li et al. (2010a) | 8 | 5 | 1 | 1 |
| B7 | Lee et al. (2002) | 8 | 6 | 1 | 1 |
| B8 | Dong and Sun (2007) | 10 | 10 | 1 | 1 |
| B9 | Jain et al. (2006) | 8-16 | 4 | 6 | 1 |
| B10 | Kim et al. (2003) | 6-18 | 15 | 24 | 2 |
| B11 | Chan et al. (2008) | 100 | 10 | 1 | 1 |

There are several papers in the literature attempting to develop solution methods for the IPPS problem. They commonly evaluate their proposed methods over the standard benchmark instances shown in Table 1. Since the IPPS problem is NP-hard, the available solution methods mainly fall into the category of heuristics and metaheuristics. Tables 2 and 3 show the papers that solve type-1 and type-2 benchmark sets of the IPPS, respectively. For last 15 years, the IPPS literature has been an active field of research and almost all researches in this field focus on metaheuristics. The small benchmark sets B1-B7 are mostly solved to optimality by solution algorithms. But, benchmark sets B8, B9 and B11 are harder, and almost none of them are optimally solved. Later in section 5, the best solution obtained for each benchmark instance is discussed.

Table 2. Available literature on type-1 IPPS.

| Publication | Algorithm | Benchmarks |
|---|---|---|
| Lian et al. (2012) | Imperialist competitive algorithm (ICA) | B1, B2, B4, B5, B6, B9, B11 |
| Li et al. (2010b) | Evolutionary algorithm (EA) | B1, B2, B5, B9 |
| Chaudhry and Usman (2017) | spreadsheet based genetic algorithm (SBGA) | B1, B2, B4, B5, B7 |
| Lihong and Shengping (2012) | Improved genetic algorithm (GA) | B2, B3, B4, |
| Ausaf et al. (2015) | Priority based heuristic (PBH) | B2, B3, |
| Li et al. (2010a) | Agent-based method (AB) | B2, B3, B11 |
| Naseri and Afshari (2012) | Hybrid genetic algorithm (HGA) | B2, B4, B5, B7, |
| Moon et al. (2008) | Evolutionary search approach (ESA) | B2, B4, |
| Jin et al. (2015) | Hybrid honey bee mating optimization (HBMO) | B2, B5, B9 |
| Leung et al. (2010) | Ant colony optimization (ACO) | B3 |
| Chan et al. (2008) | Genetic algorithms with dominant genes (GADG) | B3, B11 |
| Lee et al. (2002) | Genetic algorithm (GA1) | B7 |
| Dong and Sun (2007) | Immune genetic algorithm (IGA) | B8 |
| Zhu et al. (2009) | Particle swarm optimization (PSO) | B8 |
| Zhang et al. (2016) | Genetic algorithm (GA2) | B8 |
| Zhao and Wu (2001) | Simple genetic algorithm (SGA) | B11 |

Table 3. Available literature on type-2 IPPS.

| Publication | Algorithm |
|---|---|
| Kim et al. (2003) | Symbiotic evolutionary algorithm |
| Lian et al. (2012) | Imperialist competitive algorithm |
| Lihong and Shengping (2012) | Improved genetic algorithm |
| Ausaf et al. (2015) | Priority based heuristic |
| Jin et al. (2015) | Hybrid honey bee mating optimization |
| Leung et al. (2010) | Ant colony optimization |
| Zhang and Wong (2016) | Ant colony optimization |
| Liu et al. (2018) | Quantum-inspired hybrid algorithm |
| Li et al. (2010c) | Hybrid algorithm |
| Li et al. (2012) | Active learning genetic algorithm |
| Zhang and Wong (2015) | Object-coding genetic algorithm |

As generalization of IPPS problems, Haddadzade et al. (2016) and Zhang and Wong (2016) consider the setup times in the IPPS, Jin et al. (2016a) and Luo et al. (2017) study the multi-objective IPPS, Jin et al. (2016b) consider the IPPS with uncertain processing times, Jin et al. (2017) investigate the impact of rescheduling in the IPPS, Yin et al. (2017) consider the dynamic IPPS, and Zhang et al. (2016, 2017) study the distributed IPPS.

As reviewed, IPPS problems are well-known optimization problems. Although there are many papers proposing algorithms to solve the IPPS problems, there are the following research gaps in the literature:

1. To evaluate the performances of solution methods for the IPPS, there are several standard benchmark sets in the literature but unfortunately there is no comprehensive performance comparison among these algorithms. Our review shows that there are even some papers providing worse results than the state-of-the-art methods. Most of the benchmark sets in the literature are not optimally solved yet. For benchmark sets with larger instances, it is likely that there is still some room for improving the best-known solutions.
2. Due to the in-built hardness of the IPPS problem, research community has focused on using random-based metaheuristics to solve the problem. Although some metaheuristics are successful at escaping from local optima to some extent, they are commonly stuck in local optima at the end of the search, especially when there are two independent decision dimensions like the ones existent in the IPPS problem (i.e., assignment and sequencing). In this case, the encoding scheme has two independent parts, one for each decision, and this increases the probability of getting into a local optimum. Another weak point of metaheuristics is that they provide no theoretical optimality gap for their performance.

3. The IPPS problem is decomposable into two more trackable subproblems. These two subproblems, master and slave problems, can be solved more effectively than solving the large original IPPS problem which is commonly done by metaheuristics.

Therefore, we decided to develop a Benders' decomposition algorithm to overcome these shortcomings. In the Benders decomposition, the slave problem needs to be a continuous linear problem, where in IPPS problem decomposition, it is more effective to have the slave problem as an integer program. Thus, the logic-based Benders' decomposition is applied. Besides the algorithm, we provide a comprehensive review of the related literature on benchmark sets and compare the performance of the proposed algorithm with the ones from the literature. The proposed algorithm either results in the optimal solutions or improves the best-known solutions for all the benchmark sets; hence, it provides the state-of-the-art results for the type-1 IPPS problem.

## 3. Problem definition and formulation

A typical IPPS problem can be described as follows. There are *n* jobs where each requires a set of operations for completion. There are *m* machines to process these operations. There are alternative manufacturing process plans for each job, and the machines are flexible and can process different operations. The objective is to select one process plan for each job, assign each operation associated to that process plan to one of eligible machines, and to sequence operations of all jobs on the machines to minimize the makespan while meeting the precedence relations among operations of each job. Moreover, the following assumptions are commonly established. Jobs are independent. Operation preemption is not allowed, and each machine can handle only one job at a time. The operations within a process plan of a job are sequentially ordered. All jobs and machines are available at time zero. After a job is processed on a machine, it is immediately transported to the next machine on its process, and the transmission time is negligible. Setup time for the operations on the machines is independent of the operation sequence and is included in the processing times. This problem is in fact the flexible job shop scheduling with process plan flexibility as named by Özgüven et al. (2010). But, it is conventionally called type-1 IPPS problem.

The mathematical model of the type-1 IPPS problem is given as follows (Jin et al., 2015).

Notations and parameters

| | |
|---|---|
| $i, i'$ | index for jobs, |
| $j, j'$ | index for operations, |
| $k, k'$ | index for machines, |
| $l, l'$ | index for process plans, |
| $O_{i,l,j}$ | *j*-th operation of job *i* on the job's *l*-th process plan |
| $N$ | set of all the jobs |
| $M$ | set of all the machines |

| | |
|---|---|
| $T_i$ | set of process plans of job *i* |
| $NO_{i,l}$ | set of operations for *l*-th process plan of job *i* |
| $R_{i,l,j}$ | set of available machines for $O_{i,l,j}$ |
| $p_{i,j,k,l}$ | processing time of $O_{i,l,j}$ on machine *k* |
| $A$ | a large positive integer |

<u>Decision variables</u>

| | |
|---|---|
| $C_{max}$ | makespan. |
| $X_{i,l}$ | binary variable taking value 1, if the *l*-th process plan of job *i* is selected; 0, otherwise. |
| $Z_{i,j,k,l}$ | binary variable taking value 1, if $O_{i,l,j}$ is processed on machine *k*, 0, otherwise. |
| $Y_{i,j,l,i',j',l'}$ | binary variable taking value 1, if $O_{i,l,j}$ is processed directly or indirectly after $O_{i',l',j'}$; 0, otherwise. |
| $C_{i,j}$ | continuous variable for the completion time of the *j*-th operation of job *i* |

<u>Objective</u>

$$min \quad C_{max} \tag{1}$$

subject to:

$$\sum_{l\in T_i} X_{i,l} = 1 \qquad \forall_{i\in N} \tag{2}$$

$$\sum_{k\in R_{i,l,j}} Z_{i,j,k,l} + \left(1 - X_{i,l}\right) = 1 \qquad \forall_{i\in N, l\in T_i, j\in NO_{i,l}} \tag{3}$$

$$C_{i,0} = 0 \qquad \forall_{i\in N} \tag{4}$$

$$C_{i,j} \geq C_{i,j-1} + \sum_{k\in R_{i,l,j}} p_{i,j,k,l} Z_{i,j,k,l} \qquad \forall_{i\in N, l\in T_i, j\in NO_{i,l}, j\geq 1} \tag{5}$$

$$C_{i,j} \geq C_{i',j'} + p_{i,j,k,l} - A(3 - Y_{i,j,l,i',j',l'} - Z_{i,j,k,l} - Z_{i',j',k',l'}) \qquad \forall_{\substack{i\in N, i<|n|, i'>i, l\in T_i, l'\in T_i', j\in NO_{i,l} \\ j'\in NO_{i',l'}, k\in R_{i,l,j}\cap R_{i',l',j'}}} \tag{6}$$

$$C_{i',j'} \geq C_{i,j} + p_{i',j',k',l'} - A(2 + Y_{i,j,l,i',j',l'} - Z_{i,j,k,l} - Z_{i',j',k',l'}) \qquad \forall_{\substack{i\in N, i<|n|, i'>i, l\in T_i, l'\in T_i', j\in NO_{i,l} \\ j'\in NO_{i',l'}, k\in R_{i,l,j}\cap R_{i',l',j'}}} \tag{7}$$

$$C_{max} \geq C_{i,j} \qquad \forall_{i\in N, l\in T_i, j\in NO_{i,l}} \tag{8}$$

$$C_{i,j} \geq 0 \qquad \forall_{i\in N, l\in T_i, j\in NO_{i,l}} \tag{9}$$

$$X_{i,l}, Z_{i,j,k,l}, Y_{i,j,l,i',j',l'} \in \{0,1\} \qquad \forall_{i\in N, l\in T_i, k, j\in NO_{i,l}} \tag{10}$$

Equation (1) is the objective function that minimizes the makespan. Constraint (2) ensures that exactly one process plan is selected for every job. Constraint (3) allocates operations to their corresponding machines according to the selected process plan of that job. Constraints (4) and (5) guarantee that two operations of the same job follow a correct sequence by determining completion time of operations for each job. Constraints (6) and (7) ensure that the precedence relationship between two operations on the same machine. Constraint (8) calculates the makespan and Constraints (9) and (10) define the continuous and binary variables, respectively.

## 4. Logic-based Benders decomposition algorithms

In solving optimization problems, one important feature is computational time required for the problem at hand. In practice, the computational time is highly affected by the problem size (i.e., number of variables and constraints). The standard mathematical programming approaches develop a monolithic model simultaneously considering all variables and constraints. As a result, they usually become intractable with increased number of variables and constraints. One direction to overcome this weakness is to use the concept of "divide-and-conquer" which decomposes a big model into smaller sub-models because it is usually faster to solve smaller problems repeatedly than solving the big problem at once. Benders decomposition is a solution approach utilizing this idea (Benders, 1962).

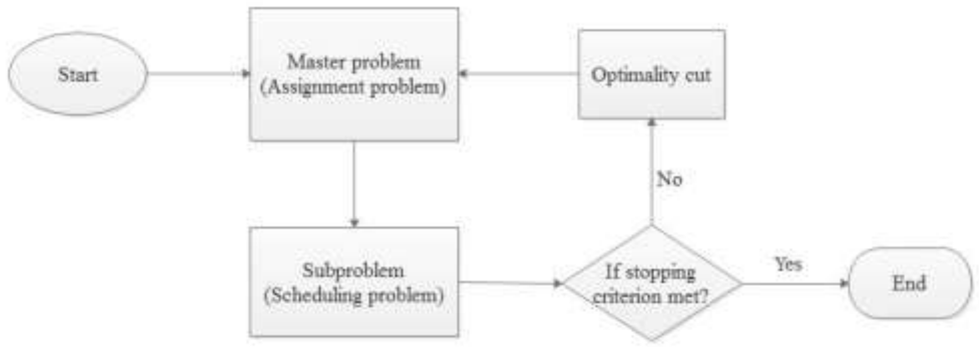
It partitions the variables of the big problem into two smaller sets of primary and secondary variables, called master-problem (MP) and sub-problem (SP). The algorithm iterates between MP and SP until their solutions converge. Since the MP is a relaxation of the original problem, its optimal solution at each iteration is a lower bound for the original model (for a minimization problem); whereas the solution of the SP, if feasible, is the best upper bound for the MP solution (Roshanaei et al., 2017; Mariel and Minner, 2017). The optimality of the (original) big problem is obtained if these two bounds converge. At each iteration, if not optimal, the SP solution is passed to the MP by optimality and feasibility cuts. The classical Benders decomposition requires the sub-problem to be a continuous linear program since it uses the concept of linear duality. The logic-based Benders decomposition algorithm, an extension of Benders, does not require the sub-problem to be a linear program. Recent successful applications of this algorithm are the inventory-location problem by Wheatley et al., (2015) and network interdiction models by Enayaty-Ahangar et al. (2018).

### 4.1. IPPS problem decomposition

The IPPS problem under consideration includes three decision dimensions:

1- Decision variables for process plan assignment ($X$)
2- Decision variables for machine assignment ($Z$)
3- Decision variables for scheduling ($Y$).

The MP includes the first two assignment decisions of $X$ and $Z$, and the SP contains sequencing decision of $Y$. For a given assignment of the MP, the schedule is determined by the SP. The SP is always feasible, but it may not be optimal. If the SP solution is equal to the master-problem solution, then the sub-problem is called optimal. Otherwise, the sub-problem is called suboptimal. If the sub-problem is labeled as optimal, then the solution of the MP is optimal. Otherwise, the algorithm adds Benders optimality cut to the MP and repeats. In our problem there is no feasibility Benders cut since the SP is always feasible. Figure 1 shows the general scheme of Benders decomposition algorithm for the IPPS. Next, the implementation of the algorithm for the IPPS is discussed in detail.


Start
Master problem (Assignment problem)
Optimality cut
No
Subproblem (Scheduling problem)
If stopping criterion met?
Yes
End
Figure 1. The general scheme of the proposed Benders decomposition algorithm.

## 4.2. Master-problem

The MP determines two decisions (process plan assignment and machine assignment). Therefore, the MP consists of variables $X$, $Z$ and $C_{max}$. The MP in its simplest form only includes Constraints (2) and (3). It is well-known that the algorithm usually converges faster if it is enhanced by the relaxation of the sub-problem. In our case, the relaxation can be any lower bound for the makespan of the SP over the solution of the MP.

The simplest lower bound for shop scheduling problems is that makespan is greater than the total processing of each job and total processing time of operations assigned to each machine. Yet, it is expected that the algorithm becomes even more efficient with a better lower bound. A tighter lower bound for the classical job shop problem is presented by Carlier and Pinson (1989). This lower bound is known for one-machine problem since it obtains a lower bound based on release and tail times of operations assigned to the machine. The release time of an operation is its earliest possible starting time while the tail time is the lowest possible completion of the corresponding job after this operation. The adaptation of this lower bound for our problem is as follows.

**Proposition 1.** $LB$ is a lower bound of the optimal makespan for the IPPS model.

$$LB = \max_{k}\{H_k\}$$

where $H_k$ is the lower bound of the optimal makespan for the one-machine problem:

$$H_k = \min_{O_{i,l,j} \in G_k} \left\{ \sum_{j' < j} \sum_{k' | O_{i,l,j'} \in G_{k'}} p_{i,j',k',l} \right\} + \sum_{i \in N} \sum_{l \in T_i} \sum_{j \in NO_{i,l} | O_{i,l,j} \in G_k} p_{i,j,k,l} + \min_{O_{i,l,j}} \left\{ \sum_{j' > j} \sum_{k' | O_{i,l,j'} \in G_{k'}} p_{i,j',k',l} \right\}$$

**Proof.** Similar to the one-machine case, we can add the minimum release and tail times among all operations of the same machine to the total processing time of operations on the same machine. To calculate the release time of operation $O_{i,l,j}$, it is required to know the machines to which the preceding operations of this operation are assigned. Thus, the release time of each operation $O_{i,l,j}$ is calculated as follows:

$$\sum_{j' < j} \sum_{k' | O_{i,l,j'} \in G_{k'}} p_{i,j',k',l}$$

Hence, the minimum release time of operations assigned to machine $k$ is

$$\min_{O_{i,l,j} \in G_k} \left\{ \sum_{j' < j} \sum_{k' | O_{i,l,j'} \in G_{k'}} p_{i,j',k',l} \right\}$$

The same procedure is used to calculate the tail time of operation $O_{i,l,j}$ (this time with successor of the operation). Adding the minimum of these two values (release and tails times) for the operations of the same machine to the total processing times gives us a lower bound for the makespan. ■

To incorporate this lower bound into the master-problem, the following constraint is used.

$$C_{max} \geq \min_{O_{i,l,j}} \left\{ Z_{i,j,k,l} \left( \sum_{j' < j} \sum_{k' \in R_{i,l,j'}} p_{i,j',k',l} Z_{i,j',k',l} \right) \right\} + \qquad \forall_{k \in M}$$
$$\sum_{i \in N} \sum_{l \in T_i} \sum_{j \in NO_{i,l} | k \in R_{i,l,j}} p_{i,j,k,l} Z_{i,j,k,l} + \min_{O_{i,l,j}} \left\{ Z_{i,j,k,l} \left( \sum_{j' > j} \sum_{k' \in R_{i,l,j'}} p_{i,j',k',l} Z_{i,j',k,l} \right) \right\}$$

In this case, the master-problem ends up with a constraint that is a mixed integer nonlinear program. We can linearize this constraint by defining new auxiliary variables. $E_k^1$ and $E_k^2$ are two continuous variables for each machine $k$ that calculate the minimum release and tail times, respectively. $Q_{i,j,k,l}^1$ ($Q_{i,j,k,l}^2$) is a binary variable that determines which operation $O_{i,j,l}$ generates the minimum release (tail) time for machine $k$. Using variables $Q_{i,j,k,l}^1$ and $Q_{i,j,k,l}^2$ and constraints (12)-(15), $E_k^1$ and $E_k^2$ take a value equal to the minimum release and tail times of operations assigned to machine $k$. Constraint (12) selects an operation for each of release and tail times on machine $k$. Constraint (13) limits the selection to operations assigned to machine $k$. Constraints (14) and (15) ensures that $E_k^1$ and $E_k^2$ are greater than the release and tail times of the selected operations. Note that since both $E_k^1$ and $E_k^2$ are positively correlated with $C_{max}$, the model tends to minimize them. Thus, Constraint (12) selects the operations with minimum release and tail times to minimize $E_k^1$ and $E_k^2$. The linear version of the master problem with this tighter lower bound is:

(MP) Objective

$min \quad C_{max}$

Subject to:

Constraint sets (2) and (3)

$$C_{max} \geq \sum_{l\in T_i}\sum_{j\in NO_{i,l}}\sum_{k\in R_{i,l,j}} p_{i,j,k,l}Z_{i,j,k,l} \qquad \forall_{i\in N} \qquad (11)$$

$$\sum_{i\in N}\sum_{l\in T_i}\sum_{j\in NO_{i,l}|k\in R_{i,l,j}} Q^h_{i,j,k,l} = Y_k \qquad \forall_{k\in M,h=1,2} \qquad (12)$$

$$Q^h_{i,j,k,l} \leq Z_{i,j,k,l} \qquad \forall_{i\in N,l\in T_i,j\in NO_{i,l},k\in R_{i,l,j},h=1,2} \qquad (13)$$

$$E^1_k \geq \sum_{j'>j}\sum_{k'\in R_{i,l,j'}} p_{i,j',k',l}Z_{i,j',k',l} - M\left(1 - Q^1_{i,j,k,l}\right) \qquad \forall_{i\in N,l\in T_i,j\in NO_{i,l},k\in R_{i,l,j}} \qquad (14)$$

$$E^2_k \geq \sum_{j'<j}\sum_{k'\in R_{i,l,j'}} p_{i,j',k',l}Z_{i,j',k,l} - M\left(1 - Q^2_{i,j,k,l}\right) \qquad \forall_{i\in N,l\in T_i,j\in NO_{i,l},k\in R_{i,l,j}} \qquad (15)$$

$$C_{max} \geq E^1_k + \sum_{i\in N}\sum_{l\in T_i}\sum_{j\in NO_{i,l}|k\in R_{i,l,j}} p_{i,j,k,l}Z_{i,j,k,l} + E^2_k \qquad \forall_{k\in M} \qquad (16)$$

$$Z_{i,j,k,l} \leq Y_k \qquad \forall_{i\in N,l\in T_i,j\in NO_{i,l},k\in R_{i,l,j}} \qquad (17)$$

$$E^1_k, E^2_k \geq 0 \qquad (18)$$

$$Q^1_{i,j,k,l}, Q^2_{i,j,k,l} \in \{0,1\} \qquad (19)$$

$$X_{i,l}, Y_k, Z_{i,j,k,l} \in \{0,1\} \qquad (19)$$

Constraints (11)-(19) are the relaxation of the sequencing decisions to estimate a lower bound for the makespan in the SP. Constraint (11) specifies that the makespan is always greater than the total processing time of operations assigned to each job. While Constraints (12)-(19) are linearized version of the $LB$ presented in proposition 1 into the MP. Binary variable $Y_k$ is also defined to calculate the minimum release and tail times for the machines to which at least one operation is assigned.

Incorporating this tighter relaxation of the SP into the MP expedites the convergence of the algorithm (Table 4 and 5). Yet, since the linearized model requires more binary variables and constraints, solving the master-problem at each iteration needs higher computational time. Since the MP is not a hard problem to solve, it does not significantly affect overall computational time.

## 4.3. Sub-problem

After solving the MP and specifying the assignment, we have a set of values for $X$ and $Z$ variables. Now, considering these values as fixed parameters in the IPPS model, the problem is reduced a classical job shop. The SP's model is as follows:

(SP) Objective

$min \quad C_{max}$

Subject to:

Constraint sets (5) and (9)

$$C_{i,j} \geq C_{i,j-1} + p_{i,j,k,l} \qquad \forall_{i \in N, l \in T_i, j \in NO_{i,l} | Z^{MP}_{i,j,k,l}=1, j \geq 1} \tag{21}$$

$$C_{i,j} \geq C_{i',j'} + p_{i,j,k,l} - A(1 - Y_{i,j,l,i',j',l'}) \qquad \forall_{\substack{i \in n, i<|n|, i'>i, l \in T_i, l' \in T_i', j \in NO_{i,l}, \\ j' \in NO_{i',l'}, k \in R_{i,l,j} \cap R_{i',l',j'} | Z^{MP}_{i,j,k,l} = Z^{MP}_{i',j',k',l'} = 1}} \tag{22}$$

$$C_{i',j'} \geq C_{i,j} + p_{i',j',k',l'} - A.Y_{i,j,l,i',j',l'} \qquad \forall_{\substack{i \in n, i<|n|, i'>i, l \in T_i, l' \in T_i', j \in NO_{i,l}, \\ j' \in NO_{i',l'}, k \in R_{i,l,j} \cap R_{i',l',j'} | Z^{MP}_{i,j,k,l} = Z^{MP}_{i',j',k',l'} = 1}} \tag{23}$$

$$C_{max} \geq C_{i,j} \qquad \forall_{i, j \in NO_{i,l} | X^{MP}_{i,l} = 1} \tag{24}$$

$$Y_{i,j,l,i',j',l'} \in \{0,1\} \tag{25}$$

where $Z^{MP}_{i,j,k,l}$ and $X^{MP}_{i,l}$ are the values of the corresponding variables in the MP at this iteration.

Constraint (21) is to meet the precedence relation among operations of each job. Constraints (22) and (23) sequence operations assigned to the same machine. Constraint (24) obtains the makespan, and Constraint (25) defines the binary variables.

## 4.4. Optimality and Benders cut

After solving the SP, the algorithm ends up with either optimality or sub-optimality. When the makespan obtained by the MP and the SP are the same, the algorithm obtains optimality and it terminates. When the makespan of the SP is greater than that of the MP, the algorithm is sub-optimal. Hence, a Benders optimality cut is added to the MP and the algorithm repeats. Since the SP is feasible for any solution of the MP, there is no need to add any Benders feasibility cut during search. The makespan obtained by the MP is a lower bound and the makespan obtained by the SP is an upper bound for the original IPPS problem.

The following Benders optimality cut is developed.

$$C_{max} \geq C^{SP}_{max}\left(1 - \sum_{i \in N} \sum_{j \in NO_{i,l}} \sum_{k \in R_{i,l,j}} \sum_{l \in T_i | Z^{MP}_{i,j,k,l}=1} (1 - Z_{i,j,k,l})\right) \tag{26}$$

where $C^{SP}_{max}$ is the makespan of the SP at the current iteration. The optimal solution of the MP is in fact a set of values for variable $Z$, and this set of values for $Z$ is the optimum for the relaxation used to estimate the makespan. When the SP is solved using these values, a feasible makespan for the original problem is obtained. The idea of this cut is to limit the makespan of the same solution in the MP to the feasible makespan obtained by the SP.

**Proposition 2.** Inequality (26) is a valid Benders optimality cut.

**Proof.** A valid Benders optimality cut must hold two properties; the cut should not remove any integer solution and the objective of the incumbent solution is bounded by the optimal solution of the SP. As clear, Equation (26) does not cut off any integer solution and only limits the makespan of the incumbent solution of the MP (i.e., a solution with the same set of assignments) to the optimal makespan of its associated SP. ■

## 4.5. Further improvements

After our initial experiments and careful analysis of the results, it turns out that the proposed algorithm is slow. Two main reasons for this slow convergence are the long computational time required to solve the SP and the weak optimality cut.

One weak point for the Benders algorithm is that it becomes slow when the SP is hard to solve. In our problem, the SP is itself a hard problem to solve. To speed up the SP, we can heuristically solve the SP and obtain a near-optimal solution. Different heuristics and approaches are tested. We find that solving the SP with branch and bound in a commercial optimization software (CPLEX) with the following stopping criterion works the best, a computational time of 5 seconds or an optimality gap of 5%, whichever comes first. After analyzing the convergence pattern of the SP, we realize that our heuristic approach obtains a good upper bound (optimal or near optimal solution) for the SP only after few seconds, and it requires a significant amount of time to increase the lower bound and to guarantee the optimality. That is, the upper bound of the SP converges after a short computational time while its lower bound convergences at a slower rate. Figure 2 shows a typical SP convergence over time in the proposed algorithm. Therefore, the stopping criterion is set to a fixed computational time and a fixed optimality gap (whichever comes first).

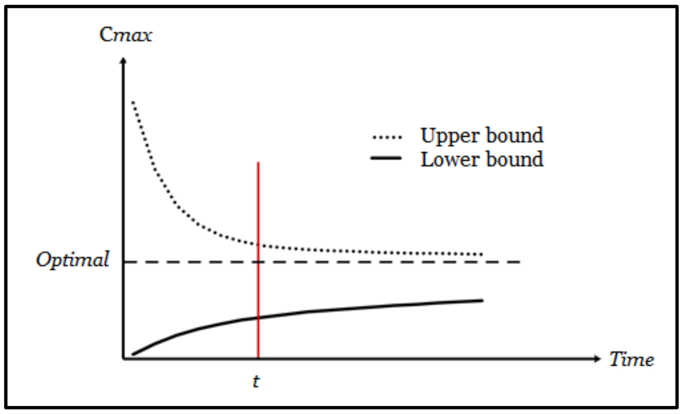


Figure 2. The convergence pattern of the sub-problem (upper and lower bounds)

In this case, the proposed algorithm converges faster, although heuristically. Note that in this case, the algorithm may stop before optimality. The upper bound is valid because the SP solution is feasible; yet, the lower bound may not be proper since the cuts added to MP are not valid.

The optimality cut is essential in convergence of the algorithm. Unfortunately, the above-mentioned Benders cut (26) does not perform well in practice since the cut only affects a small number of solutions in the MP and limits the makespan of only those solutions to $C_{max}^{SP}$. The cut can perform better if each Benders cut, instead of excluding few solutions, affect many solutions (i.e., none of which can be better than the solutions already obtained). In the proposed Benders cut (26), all operations with $Z$ value of 1 in the MP are considered. Thus, this cut can be strengthened by identifying a smaller set of operations that result in the same makespan. One way to do this end is to track which operations play a role in the determining makespan.

In scheduling, it is well-known that makespan is not improved as long as the operations in the critical path remain untouched. In other words, the makespan of any other schedule keeping the same critical path would not have lower the makespan. We implement the idea of the critical path into the cut to reduce the number of operations used in the cut. Note that the critical path concept holds only when both the assignment and the sequence of operations in the path are kept. In our problem, the MP only includes the assignment decisions. Therefore, the defined cut only holds the assignment while we may find a schedule with a lower makespan for the same assignment. Hence, the critical path-based cut is not a valid cut and the algorithm using this idea theoretically does not guarantee the optimality. In fact, this cut may remove an optimal solution.

Let us describe the idea of critical path with a numerical example with 5 jobs and 5 machines. Suppose that the schedule obtained by the SP is the one shown by Figure 3. This schedule has 17 operations. As a result, the basic no-good cut includes all these 17 operations. In this example, there are 3 different critical paths. We show $O_{ab}$ as the $b^{th}$ operation of job $a$.

Path 1) $O_{31} \rightarrow O_{32} \rightarrow O_{12} \rightarrow O_{13} \rightarrow O_{14}$

Path 2) $O_{11} \rightarrow O_{32} \rightarrow O_{12} \rightarrow O_{13} \rightarrow O_{14}$

Path 3) $O_{41} \rightarrow O_{21} \rightarrow O_{22} \rightarrow O_{34} \rightarrow O_{44}$

For this example, 3 Benders cuts are added to the MP, one cut for each critical path. Therefore, instead of one cut with 17 operations, three cuts are defined where each cut has as many operations as it has.

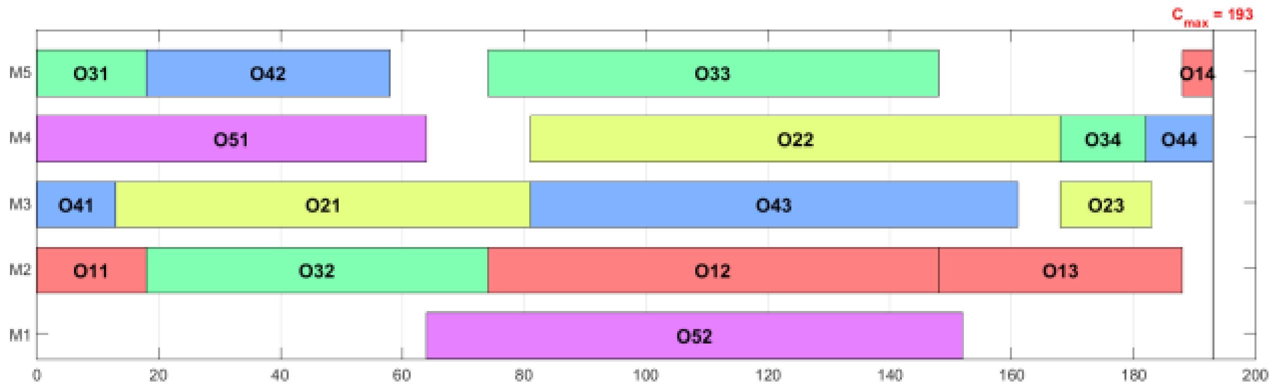


Figure 3. The schedule with three critical paths.

## 5. Experimental evaluation

This section evaluates the performance of the proposed algorithm and improvements against the IPPS model (MIP) and available algorithms from the literature of the IPPS problem. The model and algorithm are coded in C++ and IBM ILOG CPLEX 12.6.3 using Concert technology. All the experiments are run on a computer with core™ i7-5500U CPU @ 2.40 GHz Intel processor and 8.0 GBRAM. The time limit was set to the maximum 3600 seconds. An experimental study across available benchmark sets is conducted.

We test two different versions of the proposed algorithm, exact and enhanced Benders decomposition algorithms. In the exact one (EBD), both the MP and the SP are optimally solved (optimality gap of zero) and the Benders' optimality cut (26) is used. In the heuristic one (HBD) the MP is optimally solved, yet the SP is solved for at most 5 seconds or 5% optimality gap (whichever comes first). In HBD, the critical path-based optimality cut is used. In the literature of the IPPS problem, there are different benchmark sets as reviewed earlier (in Table 1). There are 10 available benchmark sets for the type-1 IPPS consisting of total of 16 instances. The performance of the proposed solution method is compared with all the 16 available algorithms given in Table 2. Note that, as discussed before, we cannot solve type-2 IPPS benchmark sets since this model is designed for the type-1 IPPS, and type-1 and type-2 problems are quite different from each other in terms of mathematical modelling.

Benchmark sets B1-B7 include small instances that are optimally solved by EBD, HBD and some of the available algorithms in a very short computational time (commonly less than one second). Their results are presented in Appendix 1. Table 4 shows the results obtained from all the algorithms over benchmark sets B8, B9, and B11 where none of instances in these benchmark sets are optimally solved by the available algorithms. For the instance in benchmark set B8, the best-known solution is 28 found by SA (Mohammadi et al., 2012). Our algorithm solved this benchmark instance optimally and the optimal solution is 27 (Appendix 2). Among the six instances of benchmark set B9, HBD improves the best-known solutions for all the six instances. Both EBD and HBD solve four instances (1, 2, 5, and 6) to optimality. The best-known solutions for the first and second instances are 507 and 586, respectively; while the optimal solutions are 492 and 582. The best-known solutions for the fifth and sixth instances are 888 and 954, respectively; while the optimal solutions are 874 and 944. For the other two instances, the optimality is not proven, the best-known solutions are improved from 670 to 665 for the third instance, and from 787 to 774 for the fourth instance. Appendix 3 presents the solutions obtained by HBD for the six instances in B9. As of benchmark set B11 with a large instance of 100 jobs and 10 machines, the best-known solution is 165 by AB (Li et al., 2010a). HBD decreases the best-known solution to 161 even for this large instance. Appendix 3 presents the Gantt chart of the schedule with makespan of 161. For this large instance, the optimality gap of the MP is set to 1% and the parameter *MIPemphasis* in CPLEX to one, and the SP stopping criterion to 1000 seconds.

Algorithms from the literature were coded with different programming languages and ran on different computers over different time frames; therefore, their computational times are not comparable. For example, Lian et al. (2012) code their algorithm with C++ and run on a computer with 2.0 GHz Intel Core2 Duo CPU. Mohammadi et al. (2012) code with MATLAB and use a computer with 2.20GHz dual-core processor and 2 GB RAM.

HBD solves all the benchmark instances within a reasonable computational time. It optimally solves the instances of benchmark sets B1, B2, B3, B5, B6, and B7 in less than one second, benchmark sets B4 and B8 in less than 25 seconds, benchmark set B9 in less than 150 seconds on average, and benchmark set B11 in less than 3450 seconds. Benchmark set B9 is discussed later in this section.

In brief, HBD either results in the optimal solution or improves the best-know solution for all the 16 available benchmark instances. In 13 instances, HBD approves and obtains the optimal solution. Among these 13 instances, five instances are solved to optimality for the first time. In the three other instances, it improves the best-known solutions. Therefore, HBD provides the state-of-the-art results for the type-1 IPPS problem.

Table 4. The results (makespan) of all the tested algorithms

| Benchmark | *n* | *m* | Algorithms | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| B8 | | | MIP | EBD | HBD | IGA | PSO | SA | GA2 |
| | 10 | 10 | **27** | **27** | **27*^** | 30 | 29 | 28 | 29 |
| B9 | | | MIP | EBD | HBD | EA | ICA | HBMO | |
| | 8 | 4 | 520 | **492** | **492*^** | 520 | 499 | 507 | |
| | 10 | 4 | 603 | **582** | **582*^** | 621 | 586 | 596 | |
| | 12 | 4 | 701 | 694 | **665^** | 724 | 679 | 670 | |
| | 14 | 4 | 854 | 785 | **774^** | 809 | 803 | 787 | |
| | 16 | 4 | 950 | **874** | **874*^** | 921 | 900 | 888 | |
| | 18 | 4 | 1013 | **944** | **944*^** | 994 | 976 | 954 | |
| B11 | | | MIP | EBD | HBD | ICA | AB | SGA | GADG |
| | 100 | 10 | 318 | 184 | **161^** | 169 | 165 | 267 | 229 |

*Optimal solution

^ The state of the art solution

To further evaluate the proposed improvements over the algorithm, four different versions are tested, HBDs 1-4. HBD1 and HBD2 use Benders' optimality cut (26) and HBD3 and HBD4 use the derived critical path-based cut. HBD1 and HBD3 use simple relaxation of SP (i.e., excluding the tight relaxation in (12)-(18) form the MP). All four HBDs may terminate solving the SP before optimality. We solve the six instances of benchmark set B9 by Jain et al. (2006) and report makespan, optimality gap, and computational times. The optimality gap is calculated with the following formula.

$$Gap = \frac{Upper\ bound - lower\ bound}{Upper\ bound} \times 100$$

Note that as of the lower bound for HBDs, the makespan of the first iteration is used. Since no cut is added to the MP yet, the makespan of the MP at the first iteration is still a valid lower bound for the makespan.

Table 5 show the results of three different versions of the proposed algorithm. From Table 5, incorporating the stronger relaxation significantly improves the performance since HBD2 and HBD4 are faster than HBD1 and HBD3. The difference between HBDs and EBDs becomes clear when the SP is hard, i.e., the idea of non-optimal solution effectively works. In terms of cut's effectiveness, HBD4 is slightly better than HBD2 and this shows that in these instances, the critical path-based optimality cut partially improves the performance. However, we observe that the critical path-based cut can be much more effective, for example, in the instance by Dong and Sun (2007). HBD2 requires 430 iterations to solve this instance while HBD4 needs only 42 iterations.

Table 5. The results by four HBDs.

| Ins. | HBD1 | | | HBD2 | | | HBD3 | | | HBD4 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $C_{max}$ | Gap (%) | Time (sec) | $C_{max}$ | Gap (%) | Time (sec) | $C_{max}$ | Gap (%) | Time (sec) | $C_{max}$ | Gap (%) | Time (sec) |
| 1 | 492 | 3.3 | 263 | 492 | 2.0 | 101 | 492 | 2.6 | 205 | 492 | 2.0 | 122 |
| 2 | 582 | 2.6 | 2815 | 582 | 2.2 | 567 | 582 | 2.6 | 2888 | 582 | 2.2 | 564 |
| 3 | 665 | 1.8 | 1294 | 665 | 0.6 | 98 | 665 | 1.8 | 1266 | 665 | 0.6 | 164 |
| 4 | 774 | 0.8 | 491 | 774 | 0.1 | 19 | 774 | 0.8 | 468 | 774 | 0.1 | 19 |
| 5 | 874 | 0.0 | 5 | 874 | 0.0 | 5 | 874 | 0.0 | 5 | 874 | 0.0 | 5 |
| 6 | 944 | 0.0 | 10 | 944 | 0.0 | 10 | 944 | 0.0 | 10 | 944 | 0.0 | 10 |

Next, the convergence behavior of the proposed algorithm is examined. We solve the six instances of benchmark set B9 with MIP, EBD, and HBD4 with 80 and 3600 seconds of the computational time. Table 6 provides the results. It is observed that HBD4 is fast to solve the IPPS problem compared to MIP and EBD. It obtains an average optimality gap of less than 2.1% in 80 seconds. EBD is also effective with the average optimality gap of 3.5%, while MIP yields the average gap of 58%. All these results strongly support the high performance of the proposed algorithm. They outperform all the existing algorithms and can obtain small optimality gap in a reasonable short computational time.

Table 6. The results of MIP, EBD, and HBD4 in two different computational times.

| Ins. | 80 seconds | | | | | | 3600 seconds | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MIP | | EBD | | HBD4 | | MIP | | EBD | | HBD4 | |
| | $C_{max}$ | Gap | $C_{max}$ | Gap | $C_{max}$ | Gap | $C_{max}$ | Gap | $C_{max}$ | Gap | $C_{max}$ | Gap |
| 1 | 520 | 34% | 492 | 1% | 492 | 2% | 504 | 32% | 492 | 0% | 492 | 2% |
| 2 | 680 | 50% | 637 | 10% | 592 | 4% | 603 | 43% | 592 | 3% | 582 | 2% |
| 3 | 821 | 58% | 696 | 6% | 681 | 4% | 701 | 51% | 696 | 6% | 665 | 1% |
| 4 | 1006 | 66% | 806 | 4% | 793 | 3% | 854 | 60% | 806 | 5% | 774 | 1% |
| 5 | 1082 | 68% | 874 | 0% | 874 | 0% | 950 | 64% | 874 | 0% | 874 | 0% |
| 6 | 1265 | 73% | 944 | 0% | 944 | 0% | 1013 | 66% | 944 | 0% | 944 | 0% |

## 6. Conclusion and future research

This paper studies the integrated process planning and scheduling problem. Although there are several papers in the literature, they all solve the problem with algorithms based on heuristics and metaheuristics over different benchmark sets. We use 16 different algorithms and 10 benchmark sets from the literature and provide a comprehensive performance comparison of the proposed method with existing ones. An enhanced logic-based Benders decomposition algorithm is proposed. We first develop an exact algorithm where the master-problem involves two decisions of process plan selection and operation to machine assignment, and the sub-problem is a pure scheduling problem. Two strong sub-problem relaxations are incorporated into the master-problem for faster convergence. Since the sub-problem is always feasible, the algorithm only requires the Benders optimality cut.

After a careful analysis of the results obtained, it turns out that the algorithm is still slow due to the long computational time for the sub-problem and the weak optimality cut. Considering the convergence pattern of the sub-problem, we give limited computational time to solve the sub-problem to speed up the algorithm. Moreover, to have a more effective cut, we use the concept of critical path and operations on the same critical path produce a separate cut. We comprehensively test the performance of our proposed algorithm on 10 different available benchmark sets (including a total of 16 instances) against 16 existing algorithms. The proposed algorithm is effective to solve the IPPS problem by resulting in the optimal solution or improving the best-known solution from the literature. The proposed algorithm finds the optimal solution in most cases (13 out of 16 instances). We present all the solutions for the improved benchmark instances in the appendices.

Productivity (i.e., resource utilization) and makespan minimization in scheduling are highly correlated (Pinedo 2008; Deidel and Arndt 1998). Regarding cost-intensive resources in manufacturing systems, every single idle time unit of resources means the capital waste. Removing unnecessary idle time results in a lot of saving for manufacturers. A lower makespan implies good resource utilization (i.e., the same production with less resource). Therefore, what this paper contributes to the production management in a broader sense is an effective planning tool that improves the productivity of production resource. For example, for the benchmark instance by Jin et al. (2006), our method improves the makespan over the average results in the literature by 3.3% which means 2% improvement over the-state-of-art result in productivity with zero-dollar cost.

This paper studies the IPPS problem with a given number of process plans, which is called type-1 IPPS. This problem is not compatible with a network graph-based instances. In fact, in this type, we must pre-count all the possible process plans from the network graph of jobs in advance. It is known that the type-1 IPPS is not effective when the number of process plans rises. One interesting future research is to extend the results to the type-2 IPPS where the algorithms and models must deal with the network graph. Another useful future research is to develop better solution methods for the SP. The SP is in fact a reentrant job shop problem where a job may visit a machine multiple times, each time for one of its operations. In this study, branch and bound method of commercial software CPLEX is used to solve the sub-problem. This is effective and fast for medium or large-sized instances. However, for very large instances like benchmark set

B11 with 100 jobs, the branch and bound becomes slow. It may be possible to adapt an algorithm from the job shop scheduling to speed up the SP.

An interesting future research direction is to use constraint programming (CP) technique to solve the subproblem since it has been reported in the literature that CP models work well to solve pure scheduling problems like the subproblem (Bukchin, and Raviv, 2018). Another idea is to consider model-based metaheuristic algorithms which combine metaheuristics and mathematical programming techniques (Leggieri and Haouari, 2018; Hernández-Leandro et al., 2018). The algorithm designed in this paper is only applicable to the type-1 IPPS problem since the problem specifications of type-1 and type-2 are significantly different. One study can extend the proposed algorithm to the type-2 IPPS problem.

## Appendixes

Appendix 1. Table 7 shows the makespan obtained by algorithms over benchmark sets B1-B7.

Table 7. The results (makespan) obtained by the algorithms over benchmarks B1-B7.

| Benchmark | *n* | *m* | Algorithms | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| B1 | | | MIP | EBD | HBD | ICA | EA | SBGA | | | |
| | 4 | 6 | **17** | **17** | **17*** | **17** | **17** | **17** | | | |
| B2 | | | MIP | EBD | HBD | GA | EA | PBH | ICA | HBMO | SBGA |
| | 5 | 3 | **33** | **33** | **33*** | **33** | **33** | **33** | **33** | **33** | **33** |
| B3 | | | MIP | EBD | HBD | GA | PBH | ACO | AB | GADG | |
| | 5 | 3 | **350** | **350** | **350*** | 360 | **350** | 380 | **350** | 360 | |
| B4 | | | MIP | EBD | HBD | HGA | GA | ICA | ESA | | |
| | 5 | 5 | **14** | **14** | **14*** | **14** | **14** | **14** | 16 | | |
| B5 | | | MIP | EBD | HBD | HGA | EA | ICA | HBMO | | |
| | 6 | 5 | **27** | **27** | **27*** | **27** | **27** | 27 | **-** | | |
| | 6 | 5 | **90** | **90** | **90*** | **-** | 92 | **90** | **90** | | |
| B6 | | | MIP | EBD | HBD | ICA | | | | | |
| | 8 | 5 | **24** | **24** | **24*** | **24** | | | | | |
| B7 | | | MIP | EBD | HBD | HGA | GA1 | SBGA | | | |
| | 8 | 6 | **23** | **23** | **23** | **23** | 34 | **23** | | | |

*Optimal solution

Appendix 2. The optimal solution for instance by Dong and Sun (2007) is presented in Figure 4.

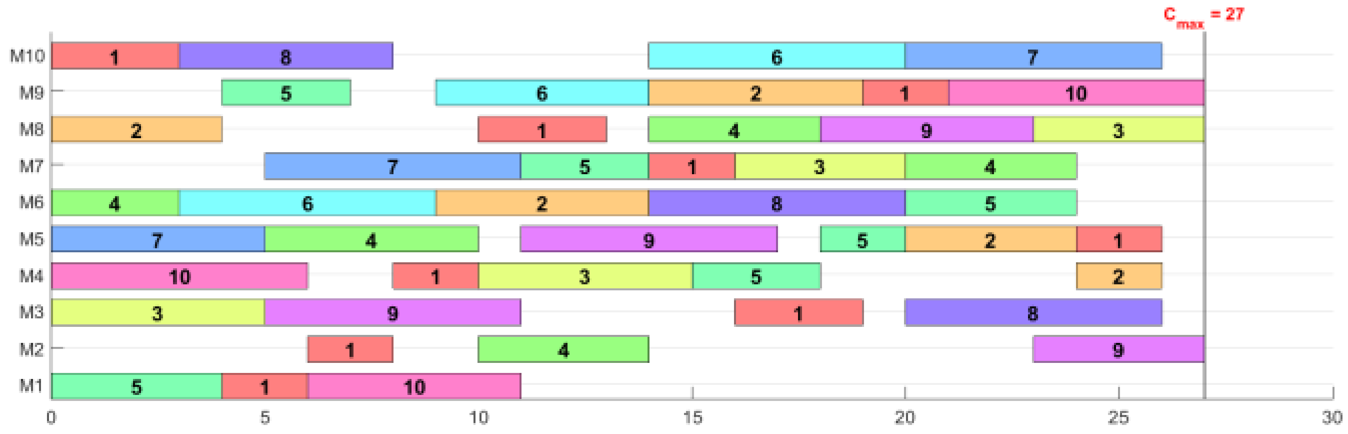


Figure 4. The optimal schedule of the instance with 10 jobs in by Dong and Sun (2007).

Appendix 3. The solutions for the six instances by Jain et al. (2006) are presented in Figures 5-10.

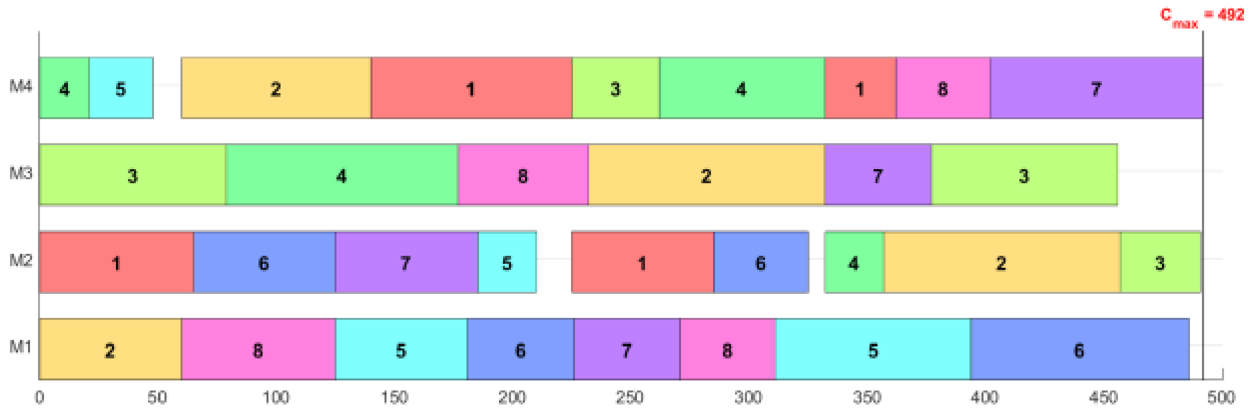


Figure 5. The optimal schedule of Instance 1 with 8 jobs by Jain et al. (2006).

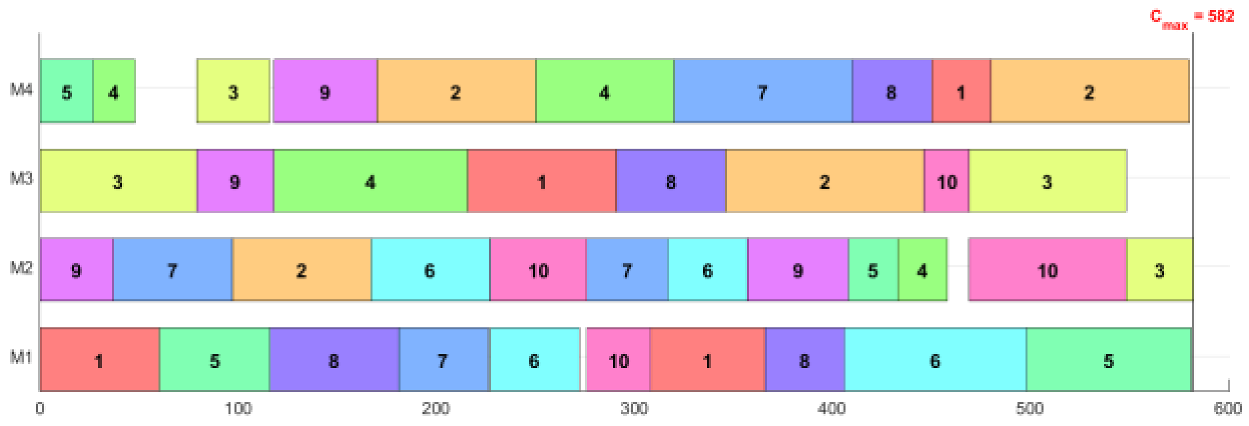


Figure 6. The optimal schedule of Instance 2 with 10 jobs by Jain et al. (2006).

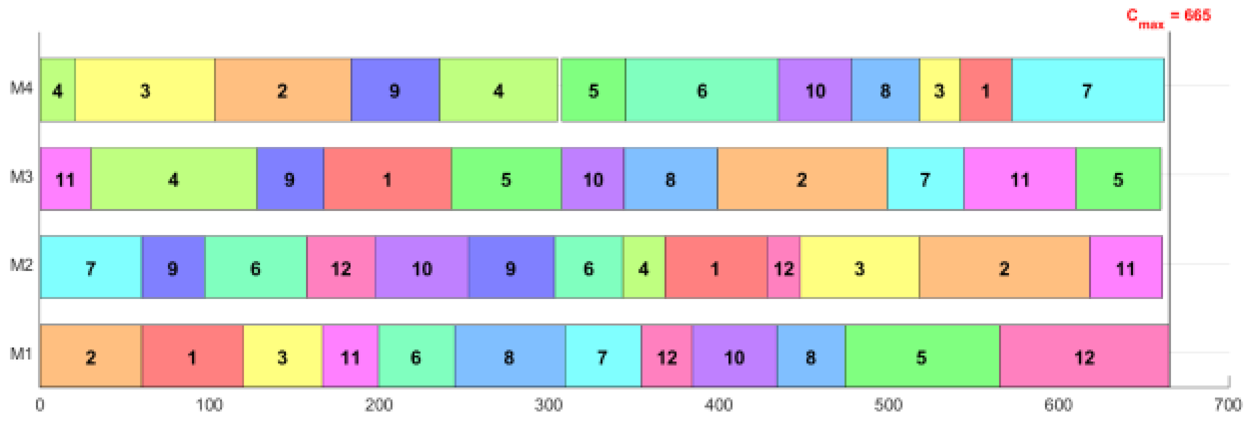


Figure 7. The schedule of Instance 3 with 12 jobs by Jain et al. (2006).

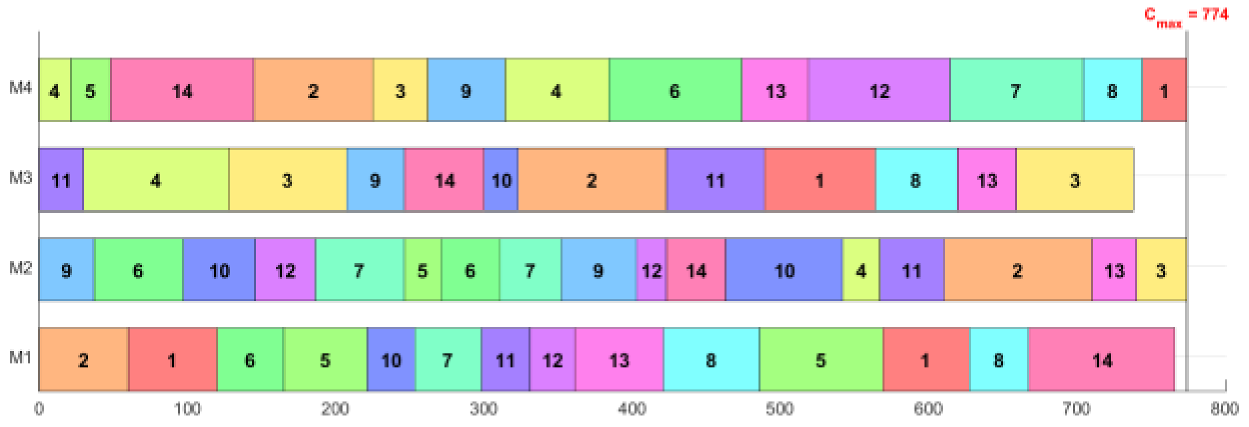


Figure 8. The schedule of Instance 4 with 14 jobs by Jain et al. (2006).

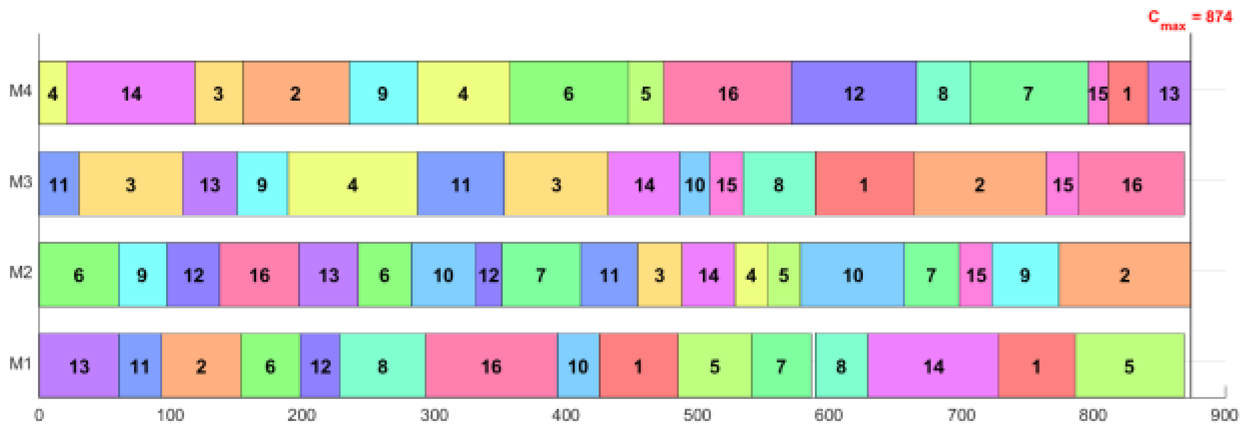


Figure 9. The optimal schedule of Instance 5 with 16 jobs by Jain et al. (2006).

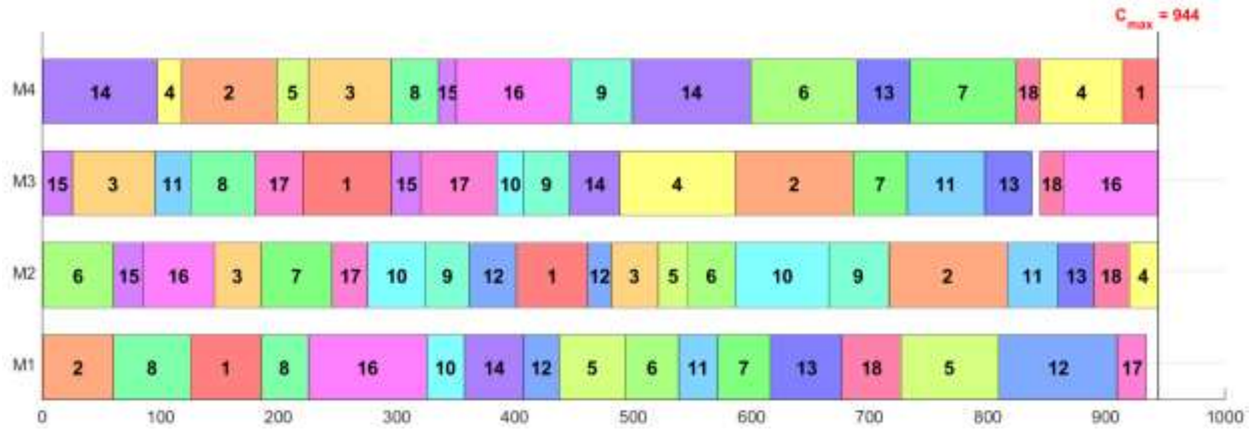


Figure 10. The optimal schedule of Instance 6 with 18 jobs by Jain et al. (2006).

Appendix 4. The solution for the instance by Chan et al. (2008) is presented in Figure 11.

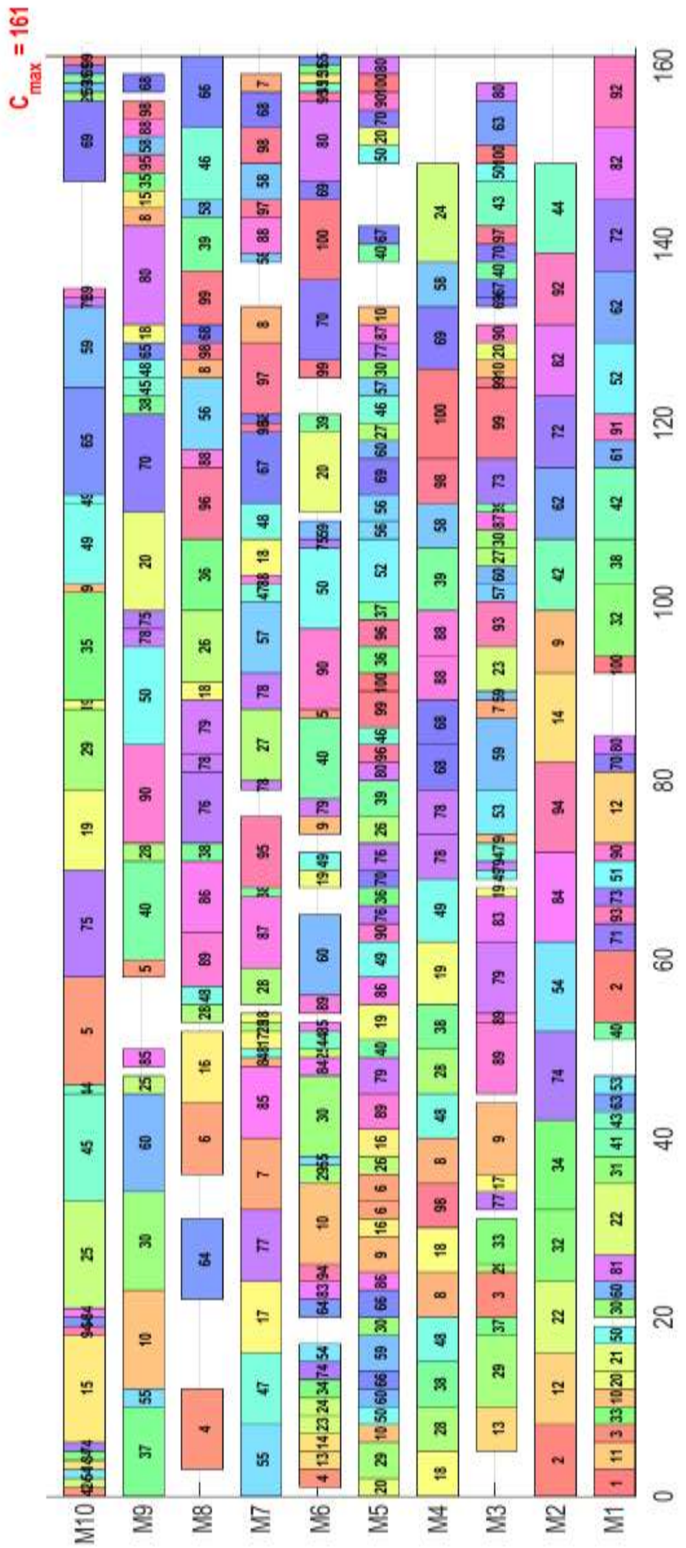


Figure 11. The optimal schedule of the instance with 8 jobs by Chan et al. (2008).

## References

Ausaf, M.F., Gao, L., Li, X. and Al Aqel, G., 2015. A priority-based heuristic algorithm (PBHA) for optimizing integrated process planning and scheduling problem. *Cogent Engineering*, *2*(1), p.1070494.

Altarazi, S. and Yasin, O., 2005. Integration of process planning and scheduling with sequence dependent setup time: a case study from electrical wires and power cable industry. *Advances in Intelligent Systems and Computing,* Springer publication, pp.283-294.

Amin-Naseri, M.R. and Afshari, A.J., 2012. A hybrid genetic algorithm for integrated process planning and scheduling problem with precedence constraints. The International Journal of Advanced Manufacturing Technology, 59(1), pp.273-287.

Benders, J.F., 1962. Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik*, *4*(1), pp.238-252.

Bukchin, Y., Raviv, T., 2018. Constraint programming for solving various assembly line balancing problems, Omega, 78, 57-68.

Chaudhry, I.A. and Usman, M., 2017. Integrated process planning and scheduling using genetic algorithms. *Tehnički Vjesnik-Technical Gazette*, 24(5), pp.1401-1409.

Chryssolouris, G., Chan, S. and Cobb, W., 1984. Decision making on the factory floor: an integrated approach to process planning and scheduling. *Robotics and Computer-Integrated Manufacturing*, *1*(3-4), pp.315-319.

Dong, C.Y., and Sun, S.D., 2007. Immune genetic algorithm job scheduling process and collaborative optimization. *Mech Sci Technol*, *26*, pp.761-766.

Enayaty-Ahangar, F., Rainwater, C.E., and Sharkey T.C., 2018. A logic-based decomposition approach for multi-period network interdiction models, *Omega*, DOI: org/10.1016/j.omega.2018.08.006.

Gu, P., Balasubramanian, S. and Norrie, D.H., 1997. Bidding-based process planning and scheduling in a multi-agent system. *Computers & Industrial Engineering*, *32*(2), pp.477-496.

Guo, Y.W., Li, W.D., Mileham, A.R. and Owen, G.W., 2009. Applications of particle swarm optimisation in integrated process planning and scheduling. *Robotics and Computer-Integrated Manufacturing*, *25*(2), pp.280-288.

Haddadzade, M., Razfar, M.R. and Zarandi, M.H.F., 2016. Multipart setup planning through integration of process planning and scheduling. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 230(6), pp.1097-1113.

Hoitomt, D.J., Luh, P.B. and Pattipati, K.R., 1993. A practical approach to job-shop scheduling problems. *IEEE transactions on Robotics and Automation*, *9*(1), pp.1-13.

Hernández-Leandro, N.A., Boyer, V., Salazar-Aguilar, M.A., Rousseau, L.M. (2018). A matheuristic based on Lagrangian relaxation for the multi-activity shift scheduling problem, *European Journal of Operational Research*, doi.org/10.1016/j.ejor.2018.07.010.

Jain, A., Jain, P.K. and Singh, I.P., 2006. An integrated scheme for process planning and scheduling in FMS. *The International Journal of Advanced Manufacturing Technology*, *30*(11), pp.1111-1118.

Jin, L., Zhang, C. and Shao, X., 2015. An effective hybrid honey bee mating optimization algorithm for integrated process planning and scheduling problems. International Journal of Advanced Manufacturing Technology, 80.

Jin, L., Zhang, C., Shao, X., Yang, X. and Tian, G., 2016a. A multi-objective memetic algorithm for integrated process planning and scheduling. *The International Journal of Advanced Manufacturing Technology*, 85(5-8), pp.1513-1528.

Jin, L., Zhang, C., Shao, X. and Tian, G., 2016b. Mathematical modeling and a memetic algorithm for the integration of process planning and scheduling considering uncertain processing times. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 230(7), pp.1272-1283.

Jin, L., Zhang, C., Shao, X. and Yang, X., 2017. A study on the impact of periodic and event-driven rescheduling on a manufacturing system: An integrated process planning and scheduling case. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 231(3), pp.490-504.

Joo, J., Park, S., Cho, H., 2001. Adaptive and dynamic process planning using neural networks. Int. J. Prod. Res. 39 (13), 2923-2946.

Khoshnevis, B. and Chen, Q.M., 1991. Integration of process planning and scheduling functions. *Journal of Intelligent Manufacturing*, *2*(3), pp.165-175.

Kim, Y.K., Park, K. and Ko, J., 2003. A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling. *Computers & operations research*, *30*(8), pp.1151-1171.

Kis, T., 2003. Job-shop scheduling with processing alternatives. *European Journal of Operational Research*, *151*(2), pp.307-332.

L. Ozbakır., 2004 Modeling, Analyzing and Solution of Multiple Objective Flexible Job Shop Scheduling Problems by Using Meta-heuristic Algorithms, Ph.D. Dissertation, Erciyes University, Social Sciences Institute, Kayseri, Turkey (in Turkish).

Lee, D.Y. and DiCesare, F., 1992, May. FMS scheduling using Petri nets and heuristic search. In *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*(pp. 1057-1062). IEEE.

Lee, Y.H., Jeong, C.S. and Moon, C., 2002. Advanced planning and scheduling with outsourcing in manufacturing supply chain. *Computers & Industrial Engineering*, 43(1-2), pp.351-374.

Lee, H. and Kim, S.S., 2001. Integration of process planning and scheduling using simulation based genetic algorithms. *The International Journal of Advanced Manufacturing Technology*, *18*(8), pp.586-590.

Leggieri, V., Haouari, M., (2018). A matheuristic for the asymmetric capacitated vehicle routing problem, *Discrete Applied Mathematics*, 234, 139-150.

Leung, C.W., Wong, T.N., Mak, K.L. and Fung, R.Y., 2010. Integrated process planning and scheduling by an agent-based ant colony optimization. *Computers & Industrial Engineering*, *59*(1), pp.166-180.

Li, W.D. and McMahon, C.A., 2007. A simulated annealing-based optimization approach for integrated process planning and scheduling. *International Journal of Computer Integrated Manufacturing*, *20*(1), pp.80-95.

Li, W.D., Gao, L., Li, X.Y. and Guo, Y., 2008, April. Game theory-based cooperation of process planning and scheduling. In *Computer Supported Cooperative Work in Design, 2008. CSCWD 2008. 12th International Conference on* (pp. 841-845). IEEE.

Li, X., Gao, L. and Li, W., 2012. Application of game theory based hybrid algorithm for multi-objective integrated process planning and scheduling. *Expert Systems with Applications*, *39*(1), pp.288-297.

Li, X., Gao, L. and Shao, X., 2012. An active learning genetic algorithm for integrated process planning and scheduling. *Expert Systems with Applications*, *39*(8), pp.6683-6691.

Li, X., Gao, L., Shao, X., Zhang, C. and Wang, C., 2010b. Mathematical modeling and evolutionary algorithm-based approach for integrated process planning and scheduling. *Computers & Operations Research*, *37*(4), pp.656-667.

Li, W.D., Ong, S.K. and Nee, A.Y.C., 2002. Hybrid genetic algorithm and simulated annealing approach for the optimization of process plan for prismatic parts. *International Journal of Production Research*, 40, 1899–1922.

Li, X., Zhang, C., Gao, L., Li, W. and Shao, X., 2010a. An agent-based approach for integrated process planning and scheduling. *Expert Systems with Applications*, *37*(2), pp.1256-1264.

Li, X., Shao, X., Gao, L. and Qian, W., 2010c. An effective hybrid algorithm for integrated process planning and scheduling. *International Journal of Production Economics*, *126*(2), pp.289-298.

Lian, K., Zhang, C., Gao, L. and Li, X., 2012. Integrated process planning and scheduling using an imperialist competitive algorithm. *International Journal of Production Research*, *50*(15), pp.4326-4343.

Lihong, Q. and Shengping, L., 2012. An improved genetic algorithm for integrated process planning and scheduling. *The International Journal of Advanced Manufacturing Technology*, *58*(5), pp.727-740.

Lim, M.K. and Zhang, D.Z., 2004. An integrated agent-based approach for responsive control of manufacturing resources. *Computers & Industrial Engineering*, *46*(2), pp.221-232.

Lim, M.K. and Zhang, Z., 2003. A multi-agent based manufacturing control strategy for responsive manufacturing. *Journal of Materials Processing Technology*, *139*(1), pp.379-384.

Liu, X., Ni, Z. and Qiu, X., 2016. Application of ant colony optimization algorithm in integrated process planning and scheduling. *The International Journal of Advanced Manufacturing Technology*, 84(1-4), pp.393-404.

Liu, M., Yi, S. and Wen, P., 2018. Quantum-inspired hybrid algorithm for integrated process planning and scheduling. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 232(6), pp.1105-1122.

Luo, G., Wen, X., Li, H., Ming, W. and Xie, G., 2017. An effective multi-objective genetic algorithm based on immune principle and external archive for multi-objective integrated process planning and scheduling. The International Journal of Advanced Manufacturing Technology, 91(9-12), pp.3145-3158.

Mariel, K., and Minner, S., 2017. Benders decomposition for a strategic network design problem under NAFTA local content requirements, *Omega*, 68, 62-75.

Mohammadi, G., Karampourhaghghi, A. and Samaei, F., 2012. A multi-objective optimisation model to integrating flexible process planning and scheduling based on hybrid multi-objective simulated annealing. *International Journal of Production Research, 50*(18), pp.5063-5076.

Nasr, N. and Elsayed, E.A., 1990. Job shop scheduling with alternative machines. *The international journal of production research*, *28*(9), pp.1595-1609.

Özgüven, C., Özbakır, L. and Yavuz, Y., 2010. Mathematical models for job-shop scheduling problems with routing and process plan flexibility. *Applied Mathematical Modelling*, *34*(6), pp.1539-1548.

Petrović, M., Vuković, N., Mitić, M. and Miljković, Z., 2016. Integration of process planning and scheduling using chaotic particle swarm optimization algorithm. *Expert Systems with Applications*, *64*, pp.569-588.

Pinedo, M.L. (2008). Scheduling, Theory, Algorithms, and Systems, *Springer Science+Business Media*, LLC (Page 17).

Roshanaei, V., Luong, C., Aleman, D., Urbach, D., (2017). Propagating logic-based Benders' decomposition approaches for distributed operating room scheduling, *European Journal of Operational Research*, *257(2)*, pp. 439–455.

Seidel R.H.A., Arndt G., (1998). Productivity Improvement in Job Shop Production, *CIRP Annals*, 37, 421-424.

Shao, X., Li, X., Gao, L. and Zhang, C., 2009. Integration of process planning and scheduling—a modified genetic algorithm-based approach. *Computers & Operations Research*, *36*(6), pp.2082-2096.

Sugimura, N., Hino, R., & Moriwaki, T., 2001. Integrated process planning and scheduling in holonic manufacturing systems. In Proceedings of IEEE international symposium on assembly and task planning soft research park (Vol. 4, pp. 250–254).

Sundaram, R.M. and Fu, S.S., 1988. Process planning and scheduling—a method of integration for productivity improvement. *Computers & Industrial Engineering*, *15*(1-4), pp.296-301.

Usher, J.M., 2003. Evaluating the impact of alternative plans on manufacturing performance. *Computers & Industrial Engineering*, *45*(4), pp.585-596.

Weintraub, A., Cormier, D., Hodgson, T., King, R., WIlson, J. and Zozom, A., 1999. Scheduling with alternatives: a link between process planning and scheduling. *IIE Transactions*, 31(11), pp.1093-1102.

Wheatley, D., Gzara, F. and Jewkes, E., 2015. Logic-based Benders decomposition for an inventory-location problem with service constraints, *Omega*, 55, 10–23.

Xia, H., Li, X. and Gao, L., 2016. A hybrid genetic algorithm with variable neighborhood search for dynamic integrated process planning and scheduling. *Computers & Industrial Engineering*, *102*, pp.99-112.

Yin, L., Gao, L., Li, X. and Xia, H., 2017. An improved genetic algorithm with rolling window technology for dynamic integrated process planning and scheduling problem. In *Computer Supported Cooperative Work in Design (CSCWD), 2017 IEEE 21st International Conference* on (pp. 414-419). IEEE.

Yu, M., Zhang, Y., Chen, K. and Zhang, D., 2015. Integration of process planning and scheduling using a hybrid GA/PSO algorithm. *The International Journal of Advanced Manufacturing Technology*, *78*(1-4), pp.583-592.

Zhang, W., Gen, M. and Jo, J., 2014. Hybrid sampling strategy-based multi-objective evolutionary algorithm for process planning and scheduling problem. *Journal of Intelligent Manufacturing*, *25*(5), pp.881-897.

Zhang, Z., Tang, R., Peng, T., Tao, L. and Jia, S., 2016. A method for minimizing the energy consumption of machining system: integration of process planning and scheduling. *Journal of cleaner production*, 137, pp.1647-1662.

Zhang, L. and Wong, T.N., 2013. Distributed genetic algorithm for integrated process planning and scheduling based on multi agent system. *IFAC Proceedings Volumes*, *46*(9), pp.760-765.

Zhang, L., and Wong, T.N., 2015. An Object-coding genetic algorithm for integrated process planning and scheduling. *European Journal of Operational Research*, *244(2)*, pp. 434-444.

Zhang, L., and Wong, T.N., 2016. Solving integrated process planning and scheduling problem with constructive meta-heuristics. *Information Sciences*, *340*, pp. 340-341.

Zhao, C. and Wu, Z., 2001. A genetic algorithm approach to the scheduling of FMSs with multiple routes. *International Journal of Flexible Manufacturing Systems*, *13*(1), pp.71-88.

Zhang, S., Yu, Z., Zhang, W., Yu, D. and Xu, Y., 2016. An extended genetic algorithm for distributed integration of fuzzy process planning and scheduling. *Mathematical Problems in Engineering*, 3763512.

Zhang, S., Xu, Y., Yu, Z., Zhang, W. and Yu, D., 2017. Combining Extended Imperialist Competitive Algorithm with a Genetic Algorithm to Solve the Distributed Integration of Process Planning and Scheduling Problem. *Mathematical Problems in Engineering*, 9628935.

Zhu, H., Ye, W., Bei, G., 2009. A particle swarm optimization for integrated process planning and scheduling. In: *IEEE 10th International Conference on Computer aided Industrial Design & Conceptual Design*, Wenzhou, China, pp. 1070-1074.