An Auto-MILP Model for Flexible Job Shop Scheduling Problem

Liping Huang, Rong Su

Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798;

(e-mail: liping.huang@ntu.edu.sg, rsu@ntu.edu.sg).

Abstract: An auto mixed integer linear programming (Auto-MILP) model is proposed to tackle the flexible job shop scheduling problem. The Auto-MILP model allows the precedence between operations of a job to be given by an arbitrary directed acyclic graph rather than a linear order. The goal is the minimization of the makespan. By incorporating the operation assignment to resources, the Auto-MILP model also allows multiple production lines in the workshop, which means more than one machine could complete the same operation, and the machine assignment is determined by the algorithm. We utilize time separation to define the machine capacity, which makes it adaptive to various kinds of job scheduling scenarios. The effectiveness and efficiency of the Auto-MILP model are evaluated using instances from simulated scheduling case and real data from an industrial job shop.

Keywords: flexible job-shop scheduling, task planning, mixed-integer linear programming, makespan minimization.

1. INTRODUCTION

The job-shop scheduling problem (JSP) can be stated as follows. Consider a set of machines and a set of jobs, each job consists of a sequence of operations to be processed in a given order, and each operation must be processed individually on a specific machine without preemption. The objective is to find a processing sequence for each machine that minimizes the makespan, which is the completion time of the last operation to be processed (Birgin et al., 2014) The flexible job-shop scheduling problem (FJSP) is an extension of JSP in that it incorporates the routing flexibility, which implies the availability of alternative machines for each job operation (Mahmoodjanloo et al., 2020; Özgüven et al., 2010). Either JSP or FJSP is basically a combinatorial optimization problem, which is usually modeled as a mixed-integer linear programming (MILP) and solved by some well-developed solvers, such as GUROBI, or solved by other optimization algorithms, such as the genetic algorithm (Zhang et al., 2020a), meta-heuristic (Ahmadian et al., 2021), memetic algorithm (Luo et al., 2020; Zhang et al., 2020b), or other hybrid algorithms (Gao et al., 2020; Li et al., 2020).

Though, the FJSP is more generic for task planning in real manufacturing scheduling systems as it allows the operation routing flexibility, the linear order of job operation, such as the research in (Fan et al., 2021), does not allow non-linear operation seuquences, such as the assembling and dissembling operations. A dissembling operation means it is followed by several subsequent operations, and the assembling operation means it is executed if and only if several previous operations are finished. Figure 1 shows a general type of job, where the job operations are presented by a directed acyclic graph. Each node represents an operation in the figure, and the directed arcs represent precedence constraints. The black nodes are assembling operations and the gray nodes are dissembling operations. Utilizing a graph structure to define the operation routing flexibility is a good choice as it allows an arbitrary precedence relation over the set of operations. Here in this paper, we adopt a directed acyclic graph for representing the executing sequence of job operations.

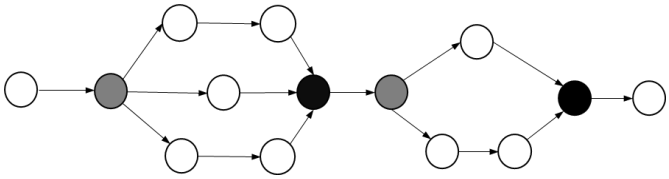

Figure 1 A representation of a more general type of job

Facing the high production rate requirements, many production systems are established to allow large-scale manufacturing. Therefore, an efficient algorithm for solving FJSP is of great significance in real job-shop scheduling systems. From the aspect of computational complexity, job-shop scheduling formulations are complex because of the large number of decision variables and constraints (Yan et al., 2018). Compared with the classical JSP, the FJSP is closer to a real production environment and has more practical applicability. However, it is more complicated than the classical JSP, because of its additional decision to assign each operation to the appropriate machine. It has been proved that the FJSP is a strongly NP-hard problem even if each job has at most three operations and there are two machines (Yuan et al., 2013; Garey et al., 1976; Lawler et al., 1993). The scheduling problem requires sophisticated techniques to produce reasonable results in an acceptable time. It is usually preferable to get good sub-optimal results quickly than to wait for days to get an optimal solution. In addition to the scheduling flexibility, solving time is an important factor for measuring the performance of FJSP formulations, which helps make the algorithm adaptive to large-scale job scheduling (Lawler et al., 1993).

Additionally, the resource capacity should be also considered in the formulation according to the real industrial producing scenarios. Hence, we propose a new MILP model that allows efficient scheduling with multiple flexibilities, including the arbitrary precedence relation over the operations and the resource capacity configurations, which is named Auto-MILP due two its multiple kinds of flexibility based on MILP. Using a generated instance and a dataset from a real industrial case, we have validated the effectiveness and efficiency of the Auto-MILP model.

The organization of this paper is as follows. In Section 2, we describe the preliminaries of the Auto-MILP model for the job-shop scheduling problem. In Section 3, we present the mathematical formulation of the Auto-MILP model, where we also describe the conversion of the logical constraints into a MILP formulation. In Section 4, we evaluate the effectiveness of the model with a simulated case and a real industrial case. Section 5 concludes the contributions of this paper.

2. PRELIMINARY

We extend the usual definition of flexible job shop scheduling problem to allow a job to be a set of operations with an arbitrary precedence relation, thus including general types of jobs. Subsequently, we describe the preliminaries of the Auto-MILP model for the job shop scheduling problem.

For the job scheduling in a workshop, a job is composed of a series of operations that can be executed on one of the machines in the workshop. Let $\boldsymbol{J} = \{J_1, J_2, \dots\}$ be a finite set of jobs, and we use j to denote a job. Let $\boldsymbol{O} = \{o_1, o_2, \dots\}$ be a finite set of operations that are allowed in the workshop.

The graph-based model describes the operational sequence of a job as a graph. Let $\mathcal{G}_j = (O_j, \mathcal{E}_j)$ be a directed acyclic graph to describe the logical sequence of operations for job j. The vertices $o_{ji}, o_{ji} \in O_j$ represent the operation set of job j. The arcs $\mathcal{E}_j$ represent precedence constraints, i.e. the operation sequence for job j. Specifically, for a job j, the directional edge $e_j = (o_{ji}, o_{ji'}) \in \mathcal{E}_j,\ o_{ji}, o_{ji'} \in O_j,\ i' \neq i$, denotes the operational sequence between two operations o_{ji} and $o_{ji'}$. It means $o_{ji'}$ is started if and only o_{ji} has been finished. Following assumption describes the operational sequence of a job.

Assumption 1: For a given job j, each pair of operations $(o_{ji}, o_{ji'}) \in \mathcal{E}_j,\ o_{ji}, o_{ji'} \in O_j,\ i' \neq i$, then $o_{ji'}$ must be started after o_{ji} has been finished.

Such graph-based sequential operations allow the precedence between operations of a job to be given by an arbitrary directed acyclic graph rather than a linear order.

Furtherly, for a given workshop, let $R = \{r_1, r_2, \dots\}$ be a finite set of resources, e.g., machines, conveyors, AGVs, storage buffers, etc. We use r to denote a resource. For each job j and a corresponding operation o_{ji}, we use $R(o_{ji})$ to denote the set of all resources that either one will allow operation o_{ji} to be completed. In addition, let $M(o_{ji})$ be required materials for the operation o_{ji}. Additionally, the following is the assumption for the resource assignment.

Assumption 2: For each job j and an operation o_{ji}, one and only one resource is needed for executing the operation. We further assume that the input operations for a given job j contains the connection operations, such as the transportation by a conveyor. The following is the assumption for the operation set inputs.

Assumption 3: For each job j and operation $(o_{ji}, o_{ji'}) \in \mathcal{E}_j,\ o_{ji}, o_{ji'} \in O_j,\ i' \neq i$, both $R(o_{ji})$ and $R(o_{ji'})$ are physically connected or can be reached from $R(o_{ji})$ to $R(o_{ji'})$, which means the transportation or connection operations by AGVs or conveyors are involved in the operation set.

3. MATHEMATICAL FORMULATION

3.1 Variables

For each job j and each operation o_{ji}, the following quantities are involved:

- An operation starting time: $T(o_{ji})$.
- The required material readiness time: $T(M_{O_{ji}})$.

For the material readiness time, following is the assumption.

Assumption 4: material readiness time for o_{ji} is assumed known, which may be derived in a separate material handling problem, e.g., an AGV fleet management problem.

- A processing time with respect to assigned resource: $\Delta(r_{o_{ji}})$, where $r_{o_{ji}} \in R(o_{ji})$.

For the processing time, following is the assumption.

Assumption 5: all processing times are known in advance.

- The changeover time for operations $o_{ji}, o_{j'i'}$ in resource r: $\delta(r, j, i, j', i')$.
- A specific resource chosen for operation o_{ji}, captured by a Boolean assignment to each resource in $R(o_{ji})$, i.e., a Boolean function $S[o_{ji}]: R(o_{ji}) \to \{0,1\}$ such that $\sum_{r \in R(o_{ji})} S[o_{ji}](r) = 1$. From now on, we write $S[o_{ji}](r)$ as $S\left(r_{o_{ji}}\right)$.

Such process plan flexibility implies the alternative machines for each operation.

At this point, we get the output decision variables $S\left(r_{o_{ji}}\right)$ and $T(o_{ji})$ for formulating the job scheduling. Besides, an auxiliary variable, $y^r_{jij'i'}$, is used to describe the operation execution order when two operations $o_{ji}, o_{j'i'}$ are allocated to the same resource r. The Boolean function is defined as: $y^r_{jij'i'} = 1$ if and only if o_{ji} started before $o_{j'i'}$, $S\left(r_{o_{ji}}\right) + S\left(r_{o_{j'i'}}\right) = 2$, otherwise $y^r_{jij'i'} = 0$.

3.2 Objective Function and Constraints

The objective of the task planning is to minimize maximum completion time (makespan), i.e. C_{max} value, which means the

ultimate time for finishing all job operations. So, the objective function is defined to be minimizing the maximum completion time of all job operations.

$$\min \max_{j,i} \left(T(o_{ji}) + \sum_{r} \left(S\left(r_{o_{ji}}\right) \times \Delta\left(r_{o_{ji}}\right) \right) \right) \quad (1)$$

To realize the job scheduling, three categories of constraints are constructed to make the scheduling feasible, including the operational time sequence, the operational resource assignment, and the resource capacity constraint. Each of the constraint is described as followings.

- Operational time sequencing of each job:

$$(\forall j \in J)\left(\forall (o_{ji}, o_{ji'}) \in \mathcal{E}_j\right) T(o_{ji'}) \geq T(o_{ji}) + \sum_{r_{o_{ji}} \in R(o_{ji})} S\left(r_{o_{ji}}\right) \Delta\left(r_{o_{ji}}\right) \quad (2)$$

As explained in the preliminary part, operation sequence of each job is described by a topological graph, which allows multiple sequential types that are elucidated in the preliminary part. $E(j)$ denotes the operation sequence set of job j. Specifically, the edge can be constructed as a finite set, where each element is the pairwise operation sequence. For example, the pairwise edge $(o_{ji}, o_{ji'})$ denotes that, for the given job j, the operation $o_{ji'}$ should only started only after the operation o_{ji} being finished. The item $\sum_{r_{o_{ji}} \in R(o_{ji})} S\left(r_{o_{ji}}\right) \Delta\left(r_{o_{ji}}\right)$ denotes the real processing time where the resource assignment is contained. We also assume that, for each job j and operation o_{ji}, one and only one resource is needed. The resource assignment constraint will be described later in this section. Hence the right part denotes the complete time of the operation o_{ji}.

Another operational sequencing constraint is about the operation starting time and the material readiness time, which is denoted as

$$(\forall j \in J)\left(\forall o_{ji} \in O_j\right) T(o_{ji}) \geq T\left(M(o_{ji})\right) \quad (3)$$

This constraint is constructed to ensure that the start time of each job operation is later than the material readiness time $T\left(M(o_{ji})\right)$. The material time for each operation is assumed to known in advance, where $T\left(M(o_{ji})\right)$ denotes the time when all required materials are ready for operation o_{ji}.

- Operational resource assignment for each job:

We assume that, for each job j and operation o_{ji}, one and only one resource is needed. The constraint is then mathematically described as

$$(\forall j \in J)\left(\forall o_{ji} \in O_j\right) \sum_{r_{o_{ji}} \in R(o_{ji})} S\left(r_{o_{ji}}\right) = 1 \quad (4)$$

- Resource capacity constraint:

We use separation time to define the resource capacity. Specifically, for each resource r, we assume that the minimum time separation is q_r, then the following constraint defines the resource capacity.

$$\begin{aligned}(\forall r \in R)\left(\forall o_{ji}, o_{j'i'} \in O\right) & S\left(r_{o_{ji}}\right) + S\left(r_{o_{j'i'}}\right) = 2 \\ & \Rightarrow \left|T(o_{ji}) - T(o_{j'i'})\right| \\ & \geq y^r_{jij'i'} \Delta\left(r_{o_{ji}}\right) + \left(1 - y^r_{jij'i'}\right) \Delta\left(r_{o_{j'i'}}\right) \\ & + \delta(r, j, j', i, i') + q_r \end{aligned} \quad (5)$$

3.3 Transform logical constraint to linear programming

Note that above resource capacity constraint is a propositional calculus constraint, i.e., a logical constraint. The antecedent part of the constraint, i.e., $S\left(r_{o_{ji}}\right) + S\left(r_{o_{j'i'}}\right) = 2$, means two operations o_{ji} and $o_{j'i'}$ are assigned to the same resource r. As described in the preliminary part, the Boolean function $y^r_{jij'i'} = 1$ means the operation o_{ji} is executed before $o_{j'i'}$ on machine r. $y^r_{jij'i'} \Delta\left(r_{o_{ji}}\right) + (1 - y^r_{jij'i'}) \Delta\left(r_{o_{j'i'}}\right)$ means the processing time of either operation that is executed first. Therefore, the inequality denotes that, when two operations are assigned to the same machine, the differential start time between two operations should be larger than the sum of the processing time of the former one $y^r_{jij'i'} \Delta\left(r_{o_{ji}}\right) + (1 - y^r_{jij'i'}) \Delta\left(r_{o_{j'i'}}\right)$, the changeover time $\delta(r, j, j', i, i')$, and the separation time q_r.

Such propositional calculus constraint should be transformed to the linear integer constraint to utilize the mixed-integer optimization techniques to obtain the solution. We further transform it to the MILP format then it could be coded and solved by the MILP solver, such as the GUROBI.

Because $S\left(r_{o_{ji}}\right) + S\left(r_{o_{j'i'}}\right) = 2$ is true if an only if $S\left(r_{o_{ji}}\right) S\left(r_{o_{j'i'}}\right) = 1$, we can replace the antecedent part of the constraint (5). Further, the product term of logical variables $S\left(r_{o_{ji}}\right) S\left(r_{o_{j'i'}}\right)$ can be replaced by an auxiliary logical variable $\theta(r, o_{ji}, o_{j'i'}) \triangleq S\left(r_{o_{ji}}\right) S\left(r_{o_{j'i'}}\right)$. Then, $\theta(r, o_{ji}, o_{j'i'}) = 1$ is true if and only if $S\left(r_{o_{ji}}\right) = 1 \wedge S\left(r_{o_{j'i'}}\right) = 1$ is true. $\theta(r, o_{ji}, o_{j'i'}) \triangleq S\left(r_{o_{ji}}\right) S\left(r_{o_{j'i'}}\right)$ can be transformed to the following constraint (6).

$$(\forall r \in R)\left(\forall o_{ji}, o_{j'i'} \in O\right) \begin{cases} -S\left(r_{o_{ji}}\right) + \theta(r, o_{ji}, o_{j'i'}) \leq 0 \\ -S\left(r_{o_{j'i'}}\right) + \theta(r, o_{ji}, o_{j'i'}) \leq 0 \\ S\left(r_{o_{ji}}\right) + S\left(r_{o_{j'i'}}\right) - \theta(r, o_{ji}, o_{j'i'}) \leq 1 \end{cases} \quad (6)$$

Hence, the constraint (5) is transformed to the following constraint (7):

$$\begin{aligned}(\forall r \in R)\left(\forall o_{ji}, o_{j'i'} \in O\right) & \theta(r, o_{ji}, o_{j'i'}) = 1 \\ & \Rightarrow \left|T(o_{ji}) - T(o_{j'i'})\right| \\ & \geq y^r_{jij'i'} \Delta\left(r_{o_{ji}}\right) + \left(1 - y^r_{jij'i'}\right) \Delta\left(r_{o_{j'i'}}\right) \\ & + \delta(r, j, j', i, i') + q_r \end{aligned} \quad (7)$$

Given the proposition $[\delta = 1] \rightarrow [f(x) \leq 0]$, where $\delta \in \{0,1\}$ is a logical variable and $f: \mathbb{R}^n \mapsto \mathbb{R}$ is linear, then the

proposition is equivalent to the linear inequality $f(x) \leq M(1-\delta)$, where $M \triangleq \max_{x \in \chi} f(x)$ according to the research in (Bemporad et al., 1999). But we should note that the function $f\left(y^r_{jij'i'}, T(o_{ji}), T(o_{j'i'})\right) = y^r_{jij'i'}\Delta\left(r_{o_{ji}}\right) + \left(1 - y^r_{jij'i'}\right)\Delta\left(r_{o_{j'i'}}\right) + \delta(r,j,j',i,i') + q_r - \left|T(o_{ji}) - T(o_{j'i'})\right|$ in constraint (7) is not a linear function because it contains the absolute term $\left|T(o_{ji}) - T(o_{j'i'})\right|$. We should transform the function to a linear one before solving the problem by the linear programming. So, we transform constraint (7) to the following two constraint (8) and (9) by considering two scenarios, i.e., $T(o_{ji}) > T(o_{j'i'})$ and $T(o_{ji}) > T(o_{j'i'})$.

$$(\forall r \in R)\left(\forall o_{ji}, o_{j'i'} \in O\right)$$

$$\begin{cases} y^r_{jij'i'}\Delta\left(r_{o_{ji}}\right) + \left(1 - y^r_{jij'i'}\right)\Delta\left(r_{o_{j'i'}}\right) + \delta(r,j,j',i,i') + q_r - \left(T(o_{ji}) - T(o_{j'i'})\right) \leq \mathrm{M}_1\left(1 - \theta\left(r,\, o_{ji},\, o_{j'i'}\right)\right) \\ T(o_{ji}) \geq T(o_{j'i'}) + \Delta\left(r_{o_{j'i'}}\right) - \mathrm{bigM}\left(y^r_{jij'i'} + \left(1 - S\left(r_{o_{ji}}\right)\right) + \left(1 - S\left(r_{o_{j'i'}}\right)\right)\right) \end{cases} \tag{8}$$

$$(\forall r \in R)\left(\forall o_{ji}, o_{j'i'} \in O\right)$$

$$\begin{cases} y^r_{jij'i'}\Delta\left(r_{o_{ji}}\right) + \left(1 - y^r_{jij'i'}\right)\Delta\left(r_{o_{j'i'}}\right) + \delta(r,j,j',i,i') + q_r - \left(T(o_{j'i'}) - T(o_{ji})\right) \leq \mathrm{M}_2\left(1 - \theta\left(r,\, o_{ji},\, o_{j'i'}\right)\right) \\ T(o_{j'i'}) \geq T(o_{ji}) + \Delta\left(r_{o_{ji}}\right) - \mathrm{bigM}\left(\left(1 - y^r_{jij'i'}\right) + \left(1 - S\left(r_{o_{ji}}\right)\right) + \left(1 - S\left(r_{o_{j'i'}}\right)\right)\right) \end{cases} \tag{9}$$

where M_1, M_2 are the maximum values of the corresponding linear function, i.e., the left part of the inequality, and $bigM$ is a specific value chosen sufficiently big.

3.4 Model analysis

We finally have the model with constraints (2-4,6,8,9). In this section, we analyze the computing complexity and the characteristics of the Auto-MILP model.

Let $\alpha \coloneqq \sum_j \sum_i \left|\{o_{ji}\}\right|$ denote the total operation number, $\beta \coloneqq \sum_j \mathcal{E}_j$ is the total edge number of all jobs' graph, $\gamma = \sum_{o_{ji} \in O} \sum_{,o_{j'i'} \in O} \sum_r \left|\{r | r \in R(o_{ji}) \cap R(o_{j'i'})\}\right|$ denotes the total executing order selection for all resources, and $\varphi \coloneqq \sum_{o_{ji} \in O} \left|\{R(o_{ji})\}\right|$ denote the total resource assignment number. Then the Auto-MILP model has $2\alpha + \varphi + 7\gamma$ constraints where 2α is the total constraint number of constraint (2) and (3), φ is the constraint number of constraint (4), and 7γ is the total constraint number of constraint (6-9). The model contains $\alpha + \gamma + \varphi$ variables, of which $\gamma + \varphi$ are binary.

Characteristics of the Auto-MILP model are concluded as a few of the following aspects. Firstly, the Auto-MILP model allows the precedence between operations of each job to be given by an arbitrary directed acyclic graph rather than a linear order of job operations. Secondly, time separation is proposed as the resource capacity, which is flexibly applicable in multiple industrial scenarios. Additionally, we make the formulation a generic one by adopting the changeover time $\delta(r,j,i,j',i')$ for two sequencial operations $o_{ji}, o_{j'i'}$ in resource *r*:

4. SIMULATIONS

In this simulation part, we firstly test the model on a generated case which are shown in Figure A1, then we utilize a data set from a real industrial job shop for evaluating the effectiveness of the Auto-MILP model. In the experiment, we utilize GUROBI as the solver.

4.1 Testing on simulated instance

We first test the model on a general simulated case. Three generated jobs are shown in the Figure A1, and the processing time information is shown in Table A1. The result for this instance is shown by the following Gantt chart shown in the following Figure 2. The makespan for this case is 620 and it consumes only 0.17 seconds of CPU time.

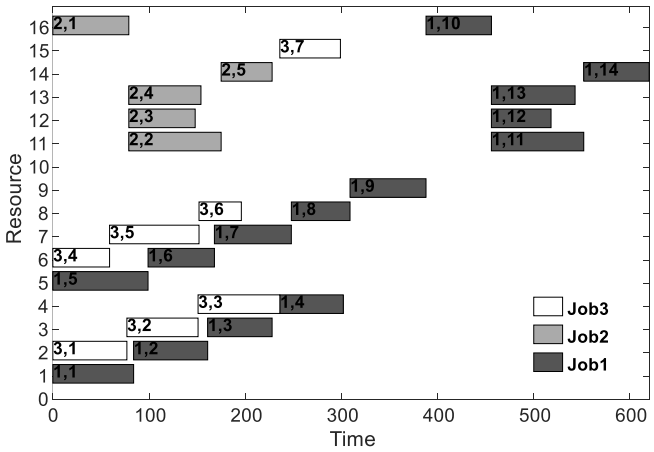

Figure 2. Gantt chart of the simulated instance.

4.2 Testing on real instance

The job information for the real instances is described in Table A2 in the Appendix, which gives the operations' directed acyclic graph and the processing time for each job. In this part, we use $J_K = \cup_{j=1}^{K}\{J_j\}$ to denote the first *K* jobs in Table A2, where *j* is the "Job ID" in Table A2. As the material time,

changeover time and the resesource capacity information is not provided in this case, we set $T\big(M(o_{ji})\big) = 0, \delta(r, j, j', i, i') = 0, q_r = 0$ for all resource and job operations. When enumerating the K value from 1 to 20, the calculated results of the CPU time and the makespan are shown in Table 1, from which we can see that even testing on a total of 20 jobs, the CPU time is just 8.27 seconds.

Table 1. Simulation results of the Auto-MILP model

K	CPU Time (sec)	C_{max}	K	CPU Time (sec)	C_{max}
1	0.01	116	**2**	0.01	159
3	0.03	194	**4**	0.05	236
5	0.08	269	**6**	0.14	295
7	0.22	330	**8**	0.39	376
9	0.52	407	**10**	0.73	407
11	0.96	407	**12**	1.19	407
13	1.50	422	**14**	2.11	486
15	2.46	521	**16**	3.54	557
17	4.54	574	**18**	5.67	622
19	6.35	655	**20**	8.27	689

5. CONCLUSIONS

We extend the definition of the FJS problem by allowing an arbitrary precedence relation over the set of operations for each job, and we presented an Auto-MILP model for the extended FJS problem. Using a simulated case and real data from an industrial job shop, the effectiveness, and efficiency of the Auto-MILP model are evaluated. The flexibility of such a general formulation exists in the following three aspects. One is that an arbitrary directed acyclic graph is applied to define the operational sequence rather than the linear sequence of job operations. The model also incorporates the operation assigning to resources. Such a job operation description allows several machines, not necessarily identical, to be capable of processing an operation, and the machine assignment is determined by the algorithm. Such flexible formulation makes the algorithm self-adaptive to the job scheduling scenarios where parallel producing is allowed. Finally, the formulation utilizes time separation to define the machine capacity. Such capacity definition makes it adaptive to various kinds of job scheduling scenarios, and to the best of our knowledge, such formulation has not been utilized in any academic research and industrial applications.

ACKNOWLEDGMENT

This study is supported under the RIE2020 Industry Alignment Fund - Industry Collaboration Projects (IAF-ICP) Funding Initiative, as well as cash and in-kind contribution from the industry partner(s), via Smart Mechatronic Lab for Industrial Collaborative Robotics in Manufacturing.

REFERENCES

Ahmadian, M., Salehipour, A., and other. (2021). A meta-heuristic to solve the just-in-time job-shop scheduling problem. *European Journal of Operational Research,* 288(1), 14-29.

Bemporad, A., and Morari, M. (1999). Control of systems integrating logic, dynamics, and constraints. *Automatica,* 35 (3), 407-427.

Birgin, E. G., Feofiloff, P., and other. (2014). A MILP model for an extended version of the flexible job shop problem. *Optimization Letters,* 8 (4), 1417-143.

Fan, H., Xiong, H., and Goh, M. (2021). Genetic Programming-based Hyper-heuristic Approach for Solving Dynamic Job Shop Scheduling Problem with Extended Technical Precedence Constraints. *Computers & Operations Research*, 134, 105401.

Gao, D., Wang, G., and Pedrycz, W. (2020). Solving fuzzy job-shop scheduling problem using DE algorithm improved by a selection mechanism. *IEEE Transactions on Fuzzy Systems,* 28 (12), 3265-3275.

Garey, M., Johnson, D., and Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of operations research,* 1 (2), 117-129.

Lawler, E., Rinnooy J., Shmoys, A. (1993). Sequencing and scheduling: algorithms and complexity. *Handbooks in operations research and management science,* 4 (9), 445–522.

Li, J., Deng, J., and other. (2020). An improved Jaya algorithm for solving the flexible job shop scheduling problem with transportation and setup times. *Knowledge-Based Systems,* 200, 106032.

Luo, Q., Deng, Q., and other. (2020) An efficient memetic algorithm for distributed flexible job shop scheduling problem with transfers. *Expert Systems with Applications* 160, 113721.

Mahmoodjanloo, M., Tavakkoli-Moghaddam R., and other. (2020). Flexible job shop scheduling problem with reconfigurable machine tools: An improved differential evolution algorithm. *Applied Soft Computing,* 94, 106416.

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), 1539-1548.

Yan, B., Mikhail A., and Peter B. (2018). Novel formulation and resolution of job-shop scheduling problems. *IEEE Robotics and Automation Letters,* 3(4), 3387-3393.

Yuan, Y., Xu, H. (2013). An integrated search heuristic for large-scale flexible job shop scheduling problems. *Computers & Operations Research,* 40(12), 2864-2877.

Zhang, G., Hu, Yu., and other. (2020a). An improved genetic algorithm for the flexible job shop scheduling problem with multiple time constraints. *Swarm and Evolutionary Computation,* 54, 100664.

Zhang, G., Sun, J., and other. (2020b). An improved memetic algorithm for the flexible job shop scheduling problem with transportation times. *Measurement and Control,* 53 (7-8), 1518-1528.

Appendix A. JOB INFORMATION

We generated an instance, which contains 3 jobs with a total of 14 kinds of operations executed on 16 machines. The directed acyclic graph of each job is described in Figure A1.

The material available time $T(M(o_{ji}))$, changeover time δ and time separation q_r for this case are all set to be zero. Processing time of the operations is given in Table A1.

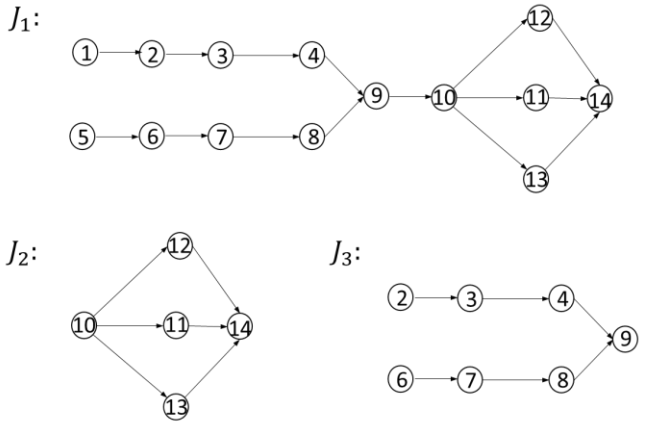

Figure A1. Directed acyclic graph for two simulated jobs.

We also used as benchmarks the 20 instances that comes from a real industrial job shop. The linear precedence of operations is a special case of presenting operation sequence as a graph.

We present the directed acyclic graph of the operation sequence and the processing time $\Delta\left(r_{o_{ji}}\right)$ in Table A2.

Table A1 Processing Time Information

J_1			J_2		
o_i	r	$\Delta\left(r_{o_{ji}}\right)$	o_i	r	$\Delta\left(r_{o_{ji}}\right)$
1	1	84	10	10,16	76,79
2	2	77	11	11	96
3	3	67	12	12	69
4	4	66	13	13	75
5	5	99	14	14	53
6	6	69	J_3		
7	7	80	o_i	r	$\Delta\left(r_{o_{ji}}\right)$
8	8	61	2	2	77
9	9,15	79,83	3	3	74
10	10,16	86,68	4	4	85
11	11	96	6	6	59
12	12	62	7	7	93
13	13	87	8	8	44
14	14	68	9	9,15	82,63

Table A2 Job Operation Information of an Industrial Case

Job ID	**directed acyclic graph**	**Processing time** $\Delta\left(r_{oji}\right)$				
		o_1, r_1	o_2, r_2	o_3, r_3	o_4, r_4	o_5, r_5
1	$o_1 \rightarrow o_3 \rightarrow o_4 \rightarrow o_5$	30	-	40	30	16
2	$o_1 \rightarrow o_3 \rightarrow o_4 \rightarrow o_5$	40	-	25	48	18
3	$o_1 \rightarrow o_3$	35	-	30	-	-
4	$o_1 \rightarrow o_2 \rightarrow o_4 \rightarrow o_5$	42	50	-	14	23
5	$o_1 \rightarrow o_3 \rightarrow o_4 \rightarrow o_5$	33	-	46	28	16
6	$o_1 \rightarrow o_2 \rightarrow o_3 \rightarrow o_4 \rightarrow o_5$	24	46	30	28	16
7	$o_1 \rightarrow o_4 \rightarrow o_5$	35	-	-	20	12
8	$o_1 \rightarrow o_2 \rightarrow o_3 \rightarrow o_4 \rightarrow o_5$	27	78	30	40	8
9	$o_1 \rightarrow o_3 \rightarrow o_5$	31	-	20	-	26
10	$o_3 \rightarrow o_4 \rightarrow o_5$	0	-	30	12	20
11	$o_3 \rightarrow o_4 \rightarrow o_5$	0	-	10	16	46
12	$o_3 \rightarrow o_4$	0	-	48	34	-
13	$o_1 \rightarrow o_5$	15	-	-	-	18
14	$o_1 \rightarrow o_2 \rightarrow o_3 \rightarrow o_4 \rightarrow o_5$	37	66	20	42	26
15	$o_1 \rightarrow o_3$	35	-	16	-	-
16	$o_1 \rightarrow o_2 \rightarrow o_3 \rightarrow o_4 \rightarrow o_5$	30	78	20	42	22
17	$o_1 \rightarrow o_3 \rightarrow o_4 \rightarrow o_5$	17	-	12	20	24
18	$o_1 \rightarrow o_2 \rightarrow o_4$	29	66	-	14	-
19	$o_1 \rightarrow o_4$	33		-	34	-
20	$o_1 \rightarrow o_2 \rightarrow o_3 \rightarrow o_4 \rightarrow o_5$	34	52	34	31	20