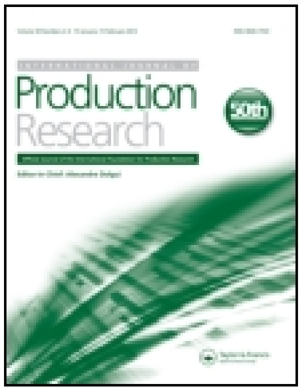

International Journal of Production Research
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tprs20

Flexible job-shop scheduling problems with 'AND'/'OR' precedence constraints

Sanghyup Lee [a], Ilkyeong Moon [b], Hyerim Bae [b] & Jion Kim [a]
[a] Automation Research Department, Industrial Research Institute, Hyundai Heavy Industries Co. Ltd, Ulsan, Korea
[b] Department of Industrial Engineering, Pusan National University, Busan, Korea

Available online: 30 Aug 2011

To cite this article: Sanghyup Lee, Ilkyeong Moon, Hyerim Bae & Jion Kim (2012): Flexible job-shop scheduling problems with 'AND'/'OR' precedence constraints, International Journal of Production Research, 50:7, 1979-2001

To link to this article: http://dx.doi.org/10.1080/00207543.2011.561375

International Journal of Production Research
Vol. 50, No. 7, 1 April 2012, 1979–2001

Flexible job-shop scheduling problems with 'AND'/'OR' precedence constraints

Sanghyup Lee[a], Ilkyeong Moon[b*], Hyerim Bae[b] and Jion Kim[a]

[a]*Automation Research Department, Industrial Research Institute, Hyundai Heavy Industries Co. Ltd, Ulsan, Korea;* [b]*Department of Industrial Engineering, Pusan National University, Busan, Korea*

(*Received 27 February 2010; final version received 24 January 2011*)

The purpose of this research is to solve flexible job-shop scheduling problems with 'AND'/'OR' precedence constraints in the operations. We first formulate the problem as a Mixed-Integer Linear Program (MILP). The MILP can be used to compute optimal solutions for small-sized problems. We also developed a heuristic algorithm that can obtain a good solution for the problem regardless of its size. Moreover, we have developed a representation and schedule builder that always produces a legal and feasible solution for the problem, and developed genetic and tabu search algorithms based on the proposed schedule builder. The results of the computational experiments show that the developed meta-heuristics are very effective.

Keywords: flexible job-shop scheduling; 'AND'/'OR' precedence constraints; heuristic; genetic algorithm; Tabu search

1. Introduction

Process planning and scheduling in a shop are considered to be major problems that must be addressed in manufacturing systems. This is necessary because these systems must systematically deal with production items and assign the resources for those items. The resources for completing the operations for a job are usually determined during process planning. The chosen resources are assigned to the operations on a time horizon at the scheduling stage. In light of resource constraints, the solution to these problems requires a two-stage procedure that includes both process planning and scheduling. First, the resources that are required for a job are decided by a two-stage process (process planning). Each operation is assigned to a resource according to the sequence of operations for jobs at this stage; only one resource is considered for each operation. Accordingly, an operation route for the job is also fixed. Then, in the second stage (scheduling), the schedule for the assigned operations is developed for each resource (Figure 1).

Process planning considers that only one resource is available for carrying out each operation in a traditional two-stage approach for fixing job routes. However, in the real world, it is common for an operation not to be dependent on a single resource, which means that an operation can be performed by one of several alternative resources. Only one resource is normally deemed to be suitable for any given operation in order to avoid the increase in scheduling complexity that arises from the consideration of alternative routes. When an operation is uniquely assigned to a resource, the scheduling problem is one of job-shop scheduling. In this problem, there are I jobs that must be processed on a group of m resources. Each job consists of a sequence of several operations, where each operation should be processed on a predefined resource during its processing time. The job-shop scheduling problem searches for a sequence of operations that are specified for each resource in order to satisfy the given objectives. When an operation has alternative resources, the scheduling problem is deemed to be a flexible job-shop scheduling problem, which is an extension of the traditional job-shop scheduling problem. In a flexible job-shop, an operation can be processed on any resource among a set of alternative resources. One resource among all alternatives for each operation is determined in the scheduling problem. A job route and a sequence of operations on resources are chosen to meet the objectives. This means that the resource assignment of operations and the scheduling of resources for the assigned operations are considered simultaneously during the problem-solving stages.

The flexible job-shop scheduling problem is an NP-hard problem, since it is an extension of the traditional job-shop scheduling problem, which has been proven to be NP-hard. Furthermore, there are some precedence constraints among the operations for completing a job. The precedence constraints are divided into 'AND' and 'OR'

*Corresponding author. Email: ikmoon@pusan.ac.kr

ISSN 0020–7543 print/ISSN 1366–588X online
© 2012 Taylor & Francis
http://dx.doi.org/10.1080/00207543.2011.561375
http://www.tandfonline.com

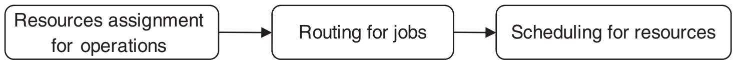

Figure 1. Two-stage procedure for process planning and scheduling.

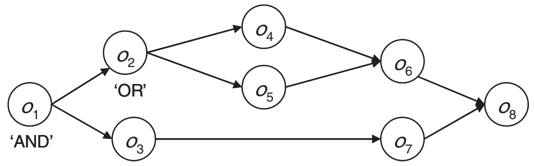

Figure 2. Example of a network containing 'AND' and 'OR' precedence constraints.

types. The 'AND' precedence constraint means that an operation can be started only after all of its immediate predecessors have been completed. The 'OR' precedence constraint means that an operation can start if (and only if) one of its immediate predecessors has been completed. We can represent the precedence constraints of a job through a network-based graph, as shown in Figure 2. Both o_2 and o_3 operations can be started after o_1 is completed. When o_2 is completed, either o_4 or o_5 can be started. If one among o_4 and o_5 is finished, o_6 can be started. Only if both o_6 and o_7 are completed can o_8 be started.

Extant scheduling approaches have considered all splits, such as the following branch of either o_1 or o_2 as 'AND' in a precedence-constraint network, which means that all branches in the network should be performed. Since the alternative operation routes have not been considered, the process planner tries to develop processes by including only 'AND' relations. To simplify scheduling problems, 'OR' relations are excluded from a job route. Consequently, an optimum schedule is optimal only for a fixed process plan. It is very difficult to choose one resource out of the alternative resources for each operation in real manufacturing environments. It is also difficult to fix the most suitable process for a job in an environment where multiple processes and resources exist. The exclusion of 'OR' relations in operations may result in missed opportunities for more effective job scheduling. To handle cases where alternative resources are given for each operation, recent studies have sought to simultaneously develop both a route for each job and a schedule for each resource.

When the operations of a job have alternative resources, there are several routes for completing the job. If alternative routings of jobs are possible, a route for each job should be determined, and then a schedule should be developed for the given operations on each resource. Generally, all of the precedence constraints among the operations of a job are regarded as 'AND' relations in these studies. However, this study addresses 'OR' relations between operations, which ultimately deals with a scheduling problem where each operation of a job is allowed to secure alternative resources, and both 'AND' and 'OR' precedence constraints of operations are present. A routing decision for each job and a scheduling problem for each resource are chosen simultaneously, rather than step by step. To find an optimal solution for the scheduling problem, an MILP mathematical formation is provided. A heuristic method and meta-heuristic algorithms are also developed. Usually, there are I jobs that consist of a sequence of operations with precedence constraints. Each operation is executed on one of the alternative resources. Furthermore, both 'AND' and 'OR' precedence constraints are allowed among the operations of a job in this paper.

This paper is organised as follows. The relevant literature on the subject is reviewed in Section 2. An MILP formulation to minimise the various objective functions is introduced in Section 3. A heuristic method for minimising the total flow time or makespan for the problem is introduced in Section 4. Chromosome and schedule builders using the meta-heuristics are proposed in Section 5. We also introduce a genetic algorithm and a tabu search algorithm for the problem in Section 5. Two types of computational experiments are performed for analysing the effectiveness of the heuristic and meta-heuristics in Section 6. We conclude in Section 7.

2. Literature review

Wilhelm and Shin (1985) conducted a study to investigate the effectiveness of alternative operations in flexible manufacturing systems. Through computational experiments, they showed that alternative operations can reduce the flow time while increasing machine utilisation. Nasr and Elsayed (1990) developed an efficient heuristic to minimise the mean flow time in a general job-shop type machining system with alternative machine tool routings. However, they did not provide an algorithm to solve the problem optimally. Liang and Dutta (1990) developed a mixed-integer programming formulation to consider simultaneously the problems of process planning and machine loading. This formulation has been used to provide an optimal process plan for each part and an optimal load for each machine by including the machining cost, material handling cost, setup cost, and machine idle cost. Hutchison *et al.* (1991) provided an optimal solution procedure to investigate the effect of routing flexibility on job-shop flexible manufacturing systems (FMS).

Meta-heuristics, such as the genetic algorithm and tabu search algorithm, have been adopted to solve the FJSP, as proven by the many papers on the topic. Brandimarte (1993) developed a tabu search algorithm for solving the problem. Genetic algorithms have proven to be effective for job-shop scheduling problems; see Cheng *et al.* (1999) for a review of the relevant studies. Park *et al.* (1998) applied a genetic algorithm to a job-shop system with alternative process plans. Mastrolilli and Gambardella (2000) proposed two neighbourhood functions for the FJSP. Zhou *et al.* (2001) developed a hybrid genetic algorithm for job-shop scheduling problems. To devise a hybrid genetic algorithm, they applied priority rules, such as the shortest processing time, for the genetic search. Kacem *et al.* (2002) used a chromosome representation that combines both routing and sequencing information and developed an approach through localisation to find promising initial assignments; then they applied dispatching rules for sequencing the operations. Kim *et al.* (2003a) presented a new evolutionary algorithm for solving FMS loading problems with machines, tools, and process flexibilities. Jia *et al.* (2003) proposed a modified GA for solving distributed scheduling problems; their GA can be adapted to the FJSP. Kim *et al.* (2004) introduced an asymmetric, multi-level, symbiotic evolutionary algorithm and applied it to the integrated problem of process planning and scheduling in FMS. Wu and Weng (2005) proposed a multi-agent scheduling method for the problem with earliness and tardiness objectives. Zhang and Gen (2005) proposed a multistage, operation-based, genetic algorithm to handle the FJSP from the viewpoint of dynamic programming. Gao *et al.* (2007) proposed a hybrid of a genetic algorithm and the bottleneck shifting method for a multi-objective FJSP. Moon *et al.* (2008) proposed MILP formulations for obtaining optimal solutions to the FJSP and a genetic algorithm using a chromosome that was composed of a priority-sequence part and an operations-assignment part.

The methods for selecting a process path and scheduling operations to complete a job are identical to those for the integrated process planning and scheduling (IPPS) in a flexible job shop. Several studies have focused on IPPS. These studies suggest that integrated process planning and scheduling methods provide better solutions than methods that solve process planning and scheduling problems separately. Moon *et al.* (2002) proposed an IPPS model for the multi-plant supply-chain and developed a genetic algorithm based heuristic approach. Lee *et al.* (2002) presented a model for advanced planning and scheduling (APS), in which each customer order has a due date and outsourcing is possible, and developed a genetic-algorithm-based heuristic approach. Kim *et al.* (2003b) introduced a symbiotic evolutionary algorithm for the IPPS problem in flexible job shop manufacturing systems. Li *et al.* (2010) developed a hybrid algorithm for IPPS. Leung *et al.* (2010) introduced an ant colony optimisation (ACO) algorithm in an agent-based system to solve IPPS problems.

3. Mathematical modelling

Some notation and definitions are required. We define a set of jobs, $J = \{j_i | i = 1, 2, 3, \ldots, N\}$, where j_i is a job, and N is the number of jobs in the problem. A job j_i is defined as a dyad of $\langle O_i, L_i \rangle$ and a labelling function f, each of which is defined below.

- $O_i = \{o_{ij} | \, j = 1, \ldots, J_i\}$ is the set of operations, where o_{ij} is the jth operation of j_i and J_i is the total number of operations in j_i.
- $L_i \subseteq \{(o_{ij^-}, o_{ij^+}) | \; o_{ij^-}, o_{ij^+} \in O_i\}$ is the set of links where an element of the form (o_{ij^-}, o_{ij^+}) indicates that o_{ij^-} immediately precedes o_{ij^+}.
- For a split operation $o_{\cdot j}$, such that $|S(\bullet_j)| > 1$, where $S(\bullet_j) = \{o_{\bullet j^+} | \; (o_{\bullet j}, o_{\bullet j^+}) \in L_\bullet\}$, $f(o_{\bullet j}) =$ 'AND' if all $o_{\bullet j^+}$'s should be executed; $f(o_{\bullet j}) =$ 'OR' if at most one of $o_{\bullet j^-}$'s can be executed.
- For a merge operation $o_{\bullet j}$, such that $|P(\bullet_j)| > 1$, where $P(\bullet_j) = \{o_{\bullet j^-} | \; (o_{\bullet j^-}, o_{\bullet j}) \in L_\bullet\}$, $f(o_{\bullet j}) =$ 'AND' if all $o_{\bullet j^-}$'s should be executed; $f(o_{\bullet j}) =$ 'OR' if at most one of $o_{\bullet j^-}$'s can be executed.
- For all operations of the form o_{ij}, if o_{ij} has no immediate predecessor, its level l_{ij} is 1; otherwise, $l_{ij} = \sum \frac{l_{i\bullet}}{|S(i\bullet)|}$ for all its immediately precedent operations of the form $o_{i\bullet}$ ($|S(i^\bullet)|$ is the number of immediately subsequent operations for $o_{i\bullet}$).
- For a merge operation $o_{\bullet j}$, the function depends on the function of a precedent split operation. If the function of a precedent split operation with equal or greater than the level of the merge function is 'AND', then that of the merge operation is 'AND'; otherwise, the function of the merge operation is 'OR'. A merge

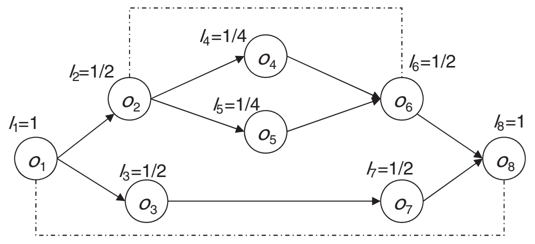

Figure 3. Relationship between the 'Split' and 'Merge' operations.

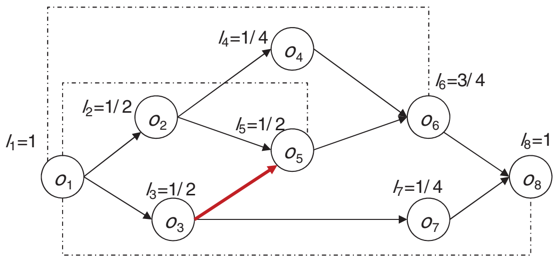

Figure 4. Distinction between 'AND' and 'OR' merge functions.

operation can be executed only if the corresponding split operation of the merge operation has been executed.

The level of an 'AND' merge operation is equal to or less than the sum of levels for its immediately precedent operations. However, the level of an 'OR' merge operation should be equal to the sum of levels for its immediately precedent operations. This means that an 'AND' merge operation can have a split operation as its immediately precedent operation. However, a split operation cannot be the immediately precedent operation of an 'OR' merge operation.

For example, as the level of the split operation o_2 is identical to that of the merge operation o_6, the corresponding split operation of o_6 is o_2 in Figure 3. Consequently, the function of o_6 is dependent on the function of o_2. If o_2 is an 'AND' split operation, then o_6 is an 'AND' merge operation. Otherwise, o_6 is an 'OR' merge operation. If o_2 is not executed, it is impossible to execute o_6; the relationship between o_8 and o_1 is similar to that between o_2 and o_6 in Figure 3.

When o_5 is the immediately subsequent operation of o_3 in Figure 3, the levels of merge operations (o_5, o_6, o_8) and their corresponding split operation are changed as in Figure 4. The corresponding operation of o_5, o_6, and o_8 is o_1. The level of both o_6 and o_8 is the same to the sum of levels of their immediately precedent operations. However, the level of o_5 is less than the sum of levels of its immediately precedent operations. In this situation, the functions of o_5, o_6, and o_8 depend on the function of o_1. The 'AND' function of both o_5 and o_1 is possible, but the 'OR' function of o_5 is not allowed in this paper. As described in the last definition, the level of an 'OR' merge operation is identical to the sum of levels of its immediately precedent operations. An 'OR' function strictly means an exclusive 'OR' in this paper. When either o_2 or o_3 is activated, o_5 can be activated. Although one of o_2 and o_3 is activated, the activation of o_5 is not guaranteed under the situation in Figure 4 because its level is lower than the sum of levels of its immediately precedent operations. In other words, a split operation is not allowed as an immediately subsequent operation for an 'OR' merge operation.

An MILP formulation for flexible job-shop scheduling with 'AND' and 'OR' precedence constraints is introduced in this section. The following notation is used for the mathematical formulation of the problem.

i	index for jobs ($i = 1, 2, \ldots, N$).
j	index for operations ($j = 1, 2, \ldots, J_i$).
k	index for resources ($k = 1, 2, \ldots, M$).
p	index for the order of processing by k-th resource ($p = 1, 2, \ldots, Q_k$).
Q_k	number of operations that belong to set $G(k)$.
$G(k)$	a set of operations that are assigned to resource k.
J_i	number of component operations for ith job.
o_{ij}	jth operation of ith job.
o_{iJ_i}	last operation of ith job. Only one last operation is included in ith job ($i = 1, 2, \ldots, N$).
S_{ij}	starting time of operation o_{ij}.
F_{ij}	finishing time of operation o_{ij}.
R_{kp}	release time of resource k after it processes its pth operation in $G(k)$.
t_{ijk}	processing time of operation o_{ij} on the kth resource.
r_i	ready time of ith job.
H	a very large positive number.

$ST_k(o_{i'j'}, o_{ij})$ setup time for o_{ij} after processing $o_{i'j'}$ on resource k

$$\text{Min} \sum_{i=1}^{N} F_{iJ_i}$$

$$\text{subject to } S_{iJ_i} + \sum_{\{k:o_{iJ_i} \in G(k)\}} \sum_{p=1}^{Q_k} t_{iJ_ik} X_{iJ_ikp} = F_{iJ_i}, \quad \forall i \tag{1}$$

$$S_{ij^-} + \sum_{\{k:o_{ij^-} \in G(k)\}} \sum_{p=1}^{Q_k} t_{ij^-k} X_{ij^-kp} \le S_{ij^+}, \quad \forall i, \quad \forall\, (o_{ij^-}, o_{ij^+}) \in L_i, \quad j^- = 1, 2, \ldots, J_i, \quad j^+ = 1, 2, \ldots, J_i \tag{2}$$

$$r_i \le S_{ij}, \quad \forall i, \quad (o_{ij^-}, o_{ij}) \notin L_i, \quad \forall j, \quad j^- = 1, 2, \ldots, J_i \tag{3}$$

$$S_{ij} + (t_{ijk} + H) X_{ijkp} - R_{kp} \le H, \quad \{(i,j) : o_{ij} \in G(k)\}, p = 1, 2, \ldots, Q_k - 1, \quad \forall k \tag{4}$$

$$R_{kp} + \sum ST_k(o_{i'j'}, o_{ij}) X_{i'j'kp} + HX_{ijkp+1} - S_{ij} \le H, \quad \{(i,j) : o_{ij} \in G(k)\}, (i', j') \ne (i,j), \quad p = 1, 2, \ldots, Q_k - 1, \quad \forall k \tag{5}$$

$$R_{kp} - R_{kp+1} \le 0, \quad p = 1, 2, \ldots, Q_k - 1, \quad \forall k \tag{6}$$

$$\sum_{\{(i,j):o_{ij} \in G(k)\}} X_{ijkp} - \sum_{\{(i,j):o_{ij} \in G(k)\}} X_{ijkp+1} \ge 0, \quad p = 1, 2, \ldots, Q_k - 1, \quad \forall k \tag{7}$$

$$\sum_{\{(i,j):o_{ij} \in G(k)\}} X_{ijkp} \le 1, \quad \forall k, \quad p = 1, 2, \ldots, Q_k \tag{8}$$

$$\sum_{\{k:o_{ij} \in G(k)\}} \sum_{p=1}^{Q_k} X_{ijkp} = 1, \quad \forall i, \quad (o_{ij^-}, o_{ij}) \notin L_i, \quad \forall j, \quad j^- = 1, 2, \ldots, J_i \tag{9}$$

$$\sum_{\{k:o_{ij^-} \in G(k)\}} \sum_{p=1}^{Q_k} X_{ij^-kp} - \sum_{\{k:o_{ij^+} \in G(k)\}} \sum_{p=1}^{Q_k} X_{ij^+kp} = 0, \quad \forall i, \quad \forall\, (o_{ij^-}, o_{ij^+}) \in L_i, \quad f(o_{ij^-}) \ne' OR', \quad f(o_{ij^+}) \ne' OR', \quad f(o_{ij^+}) \ne' AND' \text{ and } j^-, j^+ = 1, 2, \ldots, J_i \tag{10}$$

$$\sum_{\{j^+|(o_{ij^-}, o_{ij^+}) \in L_i\}} \sum_{\{k:o_{ij^+} \in G(k)\}} \sum_{p=1}^{Q_k} X_{ij^+kp} - \sum_{\{k:o_{ij^{s-}} \in G(k)\}} \sum_{p=1}^{Q_k} X_{ij^{s-}kp} = 0, \quad \forall i, \quad \forall j^-$$

$$\text{such that } |\{o_{ij^+} | (o_{ij^{s-}}, o_{ij^+}) \in L_i\}| > 1, \quad f(o_{ij^{s-}}) =' OR' \tag{11}$$

$$\sum_{\{k:o_{ij^{s-}} \in G(k)\}} \sum_{p=1}^{Q_k} X_{ij^{s-}kp} - \sum_{\{k:o_{ij^{m+}} \in G(k)\}} \sum_{p=1}^{Q_k} X_{ij^{m+}kp} = 0, \quad \forall i, \quad \forall\, (o_{ij^-}, o_{ij^+}) \in L_i,$$

$$\text{such that } |\{o_{ij^-} | (o_{ij^-}, o_{ij^{m+}}) \in L_i\}| > 1, \quad |\{o_{ij^+} | (o_{ij^{s-}}, o_{ij^+}) \in L_i\}| > 1, \quad l_{ij^{s-}} = l_{ij^{m+}} \tag{12}$$

$$\sum_{\{j^-|(o_{ij^-}, o_{ij^{m+}}) \in L_i\}} \sum_{\{k:o_{ij^-} \in G(k)\}} \sum_{p=1}^{Q_k} X_{ij^-kp} - \sum_{\{k:o_{ij^{m+}} \in G(k)\}} \sum_{p=1}^{Q_k} X_{ij^{m+}kp} = 0, \quad \forall i, \quad \forall\, (o_{ij^-}, o_{ij^+}) \in L_i,$$

such that $|\{o_{ij^-}|(o_{ij^-}, o_{ij^{m+}}) \in L_i\}| > 1$, $|\{o_{ij^+}|(o_{ij^{s-}}, o_{ij^+}) \in L_i\}| > 1$,

$$f(o_{ij^{s-}}) = f(o_{ij^{m+}}) = 'OR', \quad l_{ij^{s-}} > l_{ij^{m+}} \tag{13}$$

$$X_{ijkp} = 0 \text{ or } 1, \text{ for all } i, j, k, p \tag{14}$$

The constraints are constructed as follows.

Constraint (1) represents the flow time of the ith job. Constraint (2) imposes that if two operations in a job have an immediate precedence relationship (o_{ij^-}, o_{ij^+}), o_{ij^+} can be started only after o_{ij^-} has been completed. Constraint (3) states that the starting time of the first operation of a job must be greater than or equal to its ready time. If an operation is assigned to a resource, the release time of the resource is greater than the completion of the operation. The relationship between the resource release time and the completion time of the operation that is assigned to the resource is reflected in Constraint (4). If an operation is not assigned to a resource (i.e. $X_{ijkp} = 0$), Constraint (4) becomes inactive. Constraint (5) stipulates that the starting time of the $(p+1)$th operation that is assigned to resource k must be greater than or equal to the release time of the resource plus setup time for jobs after it processes the pth operation. The release time of resource k after it finishes the $(p+1)$th operation must be greater than or equal to that of its pth operation; this is reflected in Constraint (6). Constraint (7) implies that the $(p+1)$th operation belonging to $G(k)$ can be assigned after the pth operation has been assigned to resource k. The assumption that only one operation can be assigned to a resource at a time has been reflected in Constraint (8). Constraint (9) guarantees that every operation with no precedent operations in a job should be conducted. Constraint (10) implies that for the two operations with immediate precedence relations (o_{ij^-}, o_{ij^+}), if the function of operation o_{ij^+} is not a merge operation, operation o_{ij^+} can be assigned to a resource only after the immediately precedent operation has been assigned to that resource. Constraint (11) states that if the function of a split operation is 'OR', only one operation from its immediately subsequent operations can be conducted. Regardless of its function, a merge operation o_{ij^+} can be assigned to a resource among its alternatives after its corresponding split operation has been assigned to a resource; this is reflected in Constraint (12). Constraint (13) states that if o_{ij^+} is an 'OR' merge operation, and its level is less than that of its corresponding split operation, o_{ij^+} can be assigned to a resource after one of its immediately precedent operations has been conducted.

The levels of the merge operation o_7 and its corresponding split operation o_1 are 1, as depicted in Figure 5. Regardless of their function, the activation of o_7 is dependent on that of o_1, i.e. $\sum\sum X_{i1kp} - \sum\sum X_{i7kp} = 0$, which is reflected in Constraint (12). If the level of a merge operation is lower than that of its corresponding split operation, the condition of activation for the merge operation varies depending on the function of the operation. The level of the merge operation o_5 is lower than that of the corresponding split operation o_1. If the function of o_5 is 'AND', its activation is dependent on the activation of o_1, i.e. $\sum\sum X_{i1kp} - \sum\sum X_{i5kp} = 0$. This condition is expressed in Constraint (12). If the function of o_5 is 'OR', the activation of o_5 is dependent on the activation of its immediately precedent operations o_2 and o_3, i.e. $\sum\sum X_{i2kp} + \sum\sum X_{i3kp} - \sum\sum X_{i5kp} = 0$. This condition is expressed in Constraint (13). If o_{ij} is the pth operation to be processed on resource k, $X_{ijkp} = 1$. Otherwise, $X_{ijkp} = 0$. Constraint (14) imposes that an operation can be assigned to only one of its alternative resources.

The objective function of this MILP formulation can be changed according to the purpose. In case the objective of the scheduling is to minimise the makespan, the previous MILP formulations are changed as follows:

$$\text{Min } Z$$
$$F_{iJi} \leq Z \text{ for all } i$$

subject to (1)~(14).

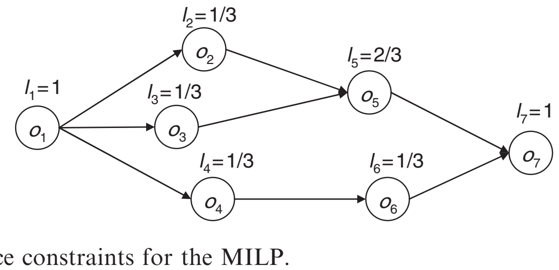

Figure 5. Example of the precedence constraints for the MILP.

In case the objective is to minimise the maximum lateness of the jobs, the MILP formulation is changed as follows:

$$\text{Min } LT_{\max}$$

$$S_{iJ_I} + \sum_{\{k:O_{iJ_I}\in G(k)\}} \sum_{p=1}^{Q_k} t_{iJ_Ik} X_{iJ_Ikp} - LT_i = D_i, \quad \text{for all } i$$

$$LT_i \le LT_{\max} \text{ for all } i$$

subject to (2)~(14).

4. Heuristic for minimising the mean flow time and makespan

A useful heuristic algorithm to produce a near-optimal solution is introduced in this section. The proposed heuristic is to minimise the mean flow time or makespan of jobs. Under the circumstances of operations with alternative resources and 'AND' or 'OR' precedence constraints between operations in a job, the following decisions are made in the scheduling problems:

- Which operation is to be chosen first from the operations that satisfy the precedence constraints?
- Which one among the alternative resources is in charge of the chosen operation?
- If the function of an operation is 'OR', which one among the immediately subsequent operations is to be chosen as the immediately subsequent operation? The following notation is used in the proposed heuristic.

l_{ij}	level of o_{ij}
To_{ij}	evaluated value of the operation time of o_{ij}
$Tl^i_{jj^+}$	evaluated value of the operation time of link (o_{ij}, o_{ij^+}) in job i
Rr_k	release time of resource k (time at which resource k becomes available)
As_{ij}	possible starting time of o_{ij}
$R(ij)$	set of alternative resources for o_{ij}
$P(ij)$	set of predecessor operations of o_{ij}.
$S(ij)$	set of successor operations of o_{ij}.
Lo_k	latest operation processed on resource k
b_{ij}	total minimum processing time of o_{ij^+}s, which is the sum of the estimated processing times of the serial successive operations and links (o_{ij^+}) to the complete job i. That is, $b_{ij} = \sum_{o_{ij^+}} t^{\min}_{ij^+}$,

(i) If o_{ij^+} is a serial or merge operation, $t^{\min}_{ij^+} = To_{ij^+}$;
(ii) else if o_{ij^+} is a split operation and $f(o_{ij^+})$='AND', $t^{\min}_{ij^+} = To_{ij^+} + \max_{o_{ip}\in S(ij^+)}(Tl^i_{j^+p})$;
(iii) else if o_{ij^+} is a split operation and $f(o_{ij^+})$='OR', $t^{\min}_{ij^+} = To_{ij^+} + \min_{o_{ip}\in S(ij^+)}(Tl^i_{j^+p})$.

Lf_{ijk} lower bound on the flow time of o_{ij}, when o_{ij} is processed on resource k.

$$Lf_{ijk} = \max\{Rr_k + ST_k(Lo_k, o_{ij}), As_{ij}\} + t_{ijk} + b_{ij}$$

t	time
$\bar{W}$	set containing all unscheduled operations.
W	set containing all schedulable operations, i.e. those for which all the preceding operations are completed $(P(ij) = \phi)$.
C	conflict set containing all the operations that can take a resource that is also required by one or more operations of a different job.

- ***Operation assignment and scheduling procedure***

Step (0): Set level value for each operation.

$$l_{ij} = 1, \quad p(ij) = \phi, \quad \forall o_{ij}$$

$l_{ij^+} = l_{ij} \div |S(ij)|$, $o_{ij^+} \in S(ij)$, where o_{ij} is a split or serial operation, $\forall o_{ij^+}$
$l_{ij^+} = \sum l_{ij}$, $o_{ij} \in P(ij^+)$, where o_{ij^+} is a merge operation, $\forall o_{ij^+}$

- Set evaluated operation time for each operation

$$To_{ij} = \frac{\sum \sqrt{Q_k} \times t_{ijk}}{|R(ij)|} \text{ or } To_{ij} = \min\left\{\sqrt{Q_k} \times t_{ijk}\right\}, \quad \forall o_{ij}, \quad k \in R(ij)$$

$|R(ij)|$: number of alternative resources of operation o_{ij}

- Evaluate $Tl^i_{jj^+}$ at a split operation o_{ij}. This step returns the result $Tl^i_{jj^+}$ by executing the following recursive procedure, EVAL_CL(j,j^+).

```
***EVAL_CL*** ($j,j^+$)
  Time eval $\leftarrow 0$
    Operation $o_{ic} \leftarrow o_{ij^+}$
      Do While ($l_{ic} < l_{ij}$)
      If $|S(ij^+)| > 1$ then
      If $f(o_{ij^+}) =$ 'OR' then
      For all $j^*$ such that $(o_{ij^+}, o_{ij^*}) \in L_i$, return min{***EVAL_CL***($j^+, j^*$)}
      Else if $f(o_{ij^+}) =$ 'AND'
        For all $j^*$ such that $(o_{ij^+}, o_{ij^*}) \in L_i$, return max{***EVAL_CL***($j^+, j^*$)}
      Else /* $|S(ij^+)| = 1$ */
        eval $\leftarrow$ eval $+ To_{ij^+}$
        $o_{ic} \leftarrow o_{ij'}$ such that $(o_{ij^+}, o_{ij'}) \in L_i$
      return eval;
```

Step (1): Let $t = 0$, $\bar{W}$ includes all operations.

Set $As_{ij} = r_i$, $\forall i$, o_{ij} has no preceding operations ($P(ij) = \phi$).

Step (2): If there is any $[o_{ij} \in \bar{W} | As_{ij} \leq t]$ in a set W, go to Step (5); otherwise, set $t = \min_{o_{ij} \in W \text{ and } P(ij) = \phi}(As_{ij})$.

Step (3): Place all $[o_{ij} \in \bar{W} | As_{ij} \leq t]$ in a set W.

Step (4): Find a resource k with the minimum finishing time among all the alternative resources for each $o_{ij} \in W$.

$$\text{Define } mC_{ij} = \min_{k \in R(ij)} \{\max[Rr_k + ST_k(Lo_k, o_{ij}), As_{ij}] + t_{ijk}\}$$

Step (5): Find an operation that requires resource k and for which resource k is required by one or more operations for each $o_{ij} \in W$. Put the operation in the conflict set C.

Step (6): Find an operation that requires resource k and for which resource k is one of the alternative resources for one or more operations in the conflict set C for each $o_{ij} \in W$. Add that operation to the conflict set C.

Step (7): For each operation in C,

(i) Calculate the lower bound on the flow time of the corresponding job

$$Lf_{ijk} = \max\{Rr_k + ST_k(Lo_k, o_{ij}), As_{ij}\} + t_{ijk} + b_{ij},$$

when o_{ij} is processed on resource k.

(ii) Select an operation such as $[o_{ij} | \min\{Lf_{ijk}\}]$ or $[o_{ij} | \max\{Lf_{ijk}\}]$ according to a predefined option.

(iii) Assign the selected operation to a resource with the minimum (or maximum) Lf_{ijk}.
(iv) Apply ***'Schedule setting and operation removal procedure'*** to o_{ij}.
(v) Remove o_{ij} from C.

If $C = \phi$, go to Step (8); else, go to (i).

Step (8): For each operation $o_{ij} \in W$ that requires resource k, which is not required by any other operation in W, assign the resource to the operation.

Step (9): Apply ***'Schedule setting and operation removal procedure'*** for each operation in $[o_{ij} \in W | o_{ij} \notin C]$,

Step (10): If $\bar{S} \neq \phi$, go to Step (2); otherwise, stop.

- ***Schedule setting and operation removal procedure***

The following procedure is applied to a selected operation o_{ij}.

(i) Determine:
$S_{ij} = \max\{Rr_k + ST_k(Lo_k, o_{ij}), As_{ij}\}, \quad F_{ij} = S_{ij} + t_{ijk}$
$As_{ij^+} = \max\{As_{ij^+}, F_{ij}\}$, for all $o_{ij^+} \in \underline{S}(ij)$
(ii) Remove operation o_{ij} from set $\bar{W}$.
(iii) If $|\ \{o_{ij^+}|(o_{ij}, o_{ij^+}) \in L_i\}| > 1$ and $f(o_{ij}) = 'OR'$, then select an immediately subsequent operation with the minimum $Tl^i_{jj^+}$ and remove the others by '*Operation removal procedure*'.

For all $o_{i\cdot} \in S(ij)$ where $o_{i\cdot}$ is not a selected operation, do the following
Operation removal procedure (o_{ic}, l_{ij})

(1) Remove o_{ic} from $\bar{W}$
(2) Remove o_{ic} from $P(id)$ and $S(ic)$, for all $o_{ic} \in P(id)$ and $o_{id} \in S(ic)$
(3) If $l_{id} < l_{ij}$ and $P(id) = \phi$, then perform ***Operation removal procedure*** (o_{id}, l_{ij})
(4) Remove o_{ij} from $P(ij^+)$ for all $o_{ij^+} \in S(ij)$.

(iv) Set $Rr_k = F_{ij}$ and $Lo_k = o_{ij}$.

An example of this heuristic is shown below. The precedence constraints for each job are represented in Figure 6. The processing time and alternative resources for each operation are represented in Table 1. Sequence-dependent setup times are shown in Table 2.

Step (0): Set the level value for each operation (see Figure 7).

$$l_{11}, l_{12} = 1, \quad l_{13}, l_{14} = 1/2, \quad l_{15} = 1/2, \quad l_{16} = l_{13} + l_{15} = 1, \quad l_{17} = 1$$
$$l_{21} = 1, \quad l_{22}, l_{23} = 1/2, \quad l_{24}, l_{25} = 1/4, \quad l_{26} = 1/4, \quad l_{27} = l_{23} + l_{25} + l_{26} = 1, \quad l_{28} = 1.$$

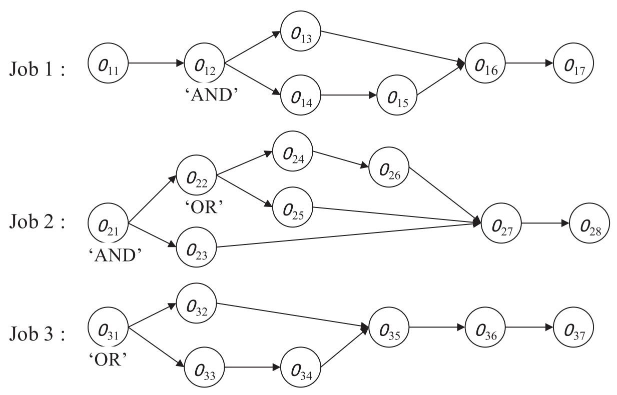

Figure 6. Precedence-constraint network for jobs.

Table 1. Processing times and alternative resources of operations.

Job \ Resource		R_1	R_2	R_3	R_4	R_5
Job	O_{11}	6	5			8
	O_{12}			9	9	
	O_{13}		8	5	7	
Job 1	O_{14}	8				6
	O_{15}		8		9	
	O_{16}	7		5		
	O_{17}				7	5
Job 2	O_{21}		5			7
	O_{22}	7		9	8	
	O_{23}	8			7	5
	O_{24}		2		3	
	O_{25}	9				10
	O_{26}		6	5	4	
	O_{27}	8		6	5	
	O_{28}			7		5
Job 3	O_{31}	6	8			5
	O_{32}	6		6		7
	O_{33}				3	5
	O_{34}		3	4		
	O_{35}	5			6	6
	O_{36}		10		7	
	O_{37}	5		7		

Table 2. Sequence-dependent setup times.

From \ To		Job		
		1	2	3
Job	–	0	0	0
	1	1	3	2
	2	2	1	2
	3	3	2	1

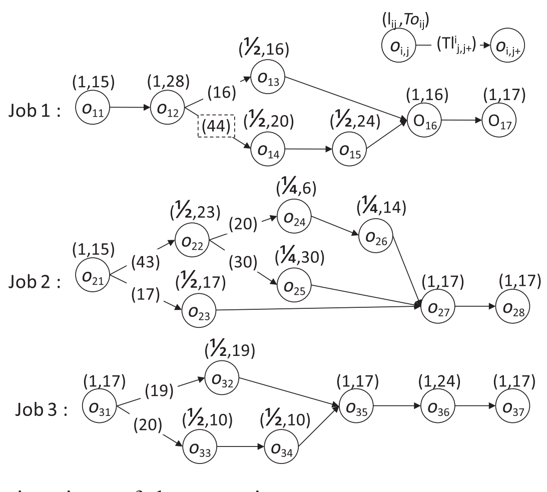

Figure 7. Levels and evaluated operation times of the operations.

- Set the evaluated operation time for each operation (see Figure 7).

$$
\begin{aligned}
To_{ij} &= \min\left\{\sqrt{Q_k} \times t_{ijk}\right\}, \quad Q_1 = 11, \quad Q_2 = 9, \quad Q_3 = 10, \quad Q_4 = 12, \quad Q_5 = 11 \\
To_{11} &= \min\left\{\sqrt{11} \times 6, \sqrt{9} \times 5, \sqrt{11} \times 8\right\} = \{19.9, 15, 26.5\} = 15, \\
To_{12} &= \min\left\{\sqrt{10} \times 9, \sqrt{12} \times 9\right\} = \{28.4, 31.2\} = 28.4 \approx 28, \ldots
\end{aligned}
$$

- Evaluate $Tl^{i}_{ij^+}$ for each split operation o_{ij} (see Figure 7).

$$
\begin{aligned}
& Tl^1_{23} = To_{13} = 16, \quad Tl^1_{24} = To_{14} + To_{15} = 20 + 24 = 44 \\
& Tl^2_{24} = To_{24} + To_{26} = 6 + 14 = 20, \quad Tl^2_{25} = To_{25} = 30, \\
& Tl^2_{12} = To_{22} + \min\{Tl^2_{24}, Tl^2_{25}\} = 23 + \min\{20, 30\} = 43, \quad Tl^2_{13} = To_{23} = 17 \\
& Tl^3_{12} = To_{32} = 19, \quad Tl^3_{13} = To_{33} + To_{34} = 20
\end{aligned}
$$

Step (1): Let $t = 0$, $\bar{W} = \{o_{11}, o_{12}, \ldots, o_{17}, o_{21}, \ldots, o_{28}, o_{31}, \ldots, o_{37}\}$, $As_{11} = r_1 = 0$, $As_{21} = r_2 = 0$, $As_{31} = r_3 = 0$

Step (2): $P(11) = P(21) = P(31) = \phi, \quad t = \min\{As_{11}, As_{21}, As_{31}\} = 0$

Step (3): $W = \{o_{11}, o_{21}, o_{31}\}$

Step (4):

$$
\begin{aligned}
& mC_{11} = \min_{k \in R(11)}\{\max[Rr_k + ST(-, o_{1\cdot}), As_{11}] + t_{11k}\} = \min\{\max\{0 + 0, 0\} + 6, 5, 8\} = 5 \\
& mC_{21} = \min\{5, 7\} = 5, \quad mC_{31} = \min\{6, 8, 5\} = 5
\end{aligned}
$$

o_{11} requires R_2. o_{21} requires R_2. o_{31} requires R_5.

Step (5): o_{11} and o_{21} are requiring R_2 at the same time.
Therefore, initialize the Conflict set with o_{11} and o_{21}, i.e. $C = \{o_{11}, o_{21}\}$.

Step (6): Since R_5 that is required by o_{31} is the alternative resource for o_{11} and o_{21}, add o_{31} to Conflict set C. $C = \{o_{11}, o_{21}, o_{31}\}$

Step (7): Solve the conflict set.

$$
\begin{aligned}
Lf_{ijk} &= \max\{Rr_k + ST(Lo_k, o_{i\cdot}), As_{ij}\} + t_{ijk} + b_{ij}, \quad b_{ij} = \sum_{o_{ij^+}} t^{\min}_{ij^+} \\
b_{11} &= 28 + \max\{16, 44\} + 16 + 17 = 105, \\
Lf_{111} &= \max\{0 + 0, 0\} + 6 + 105 = 111, \quad Lf_{112} = 110, \quad Lf_{115} = 113 \\
b_{21} &= \max\{43, 17\} + 17 + 17 = 77, \\
Lf_{212} &= \max\{0 + 0, 0\} + 5 + 77 = 82, \quad Lf_{215} = 84. \\
b_{31} &= \min\{19, 20\} + 17 + 24 + 17 = 77, \\
Lf_{311} &= \max\{0 + 0, 0\} + 6 + 77 = 83, \quad Lf_{312} = 85, \quad Lf_{315} = 82.
\end{aligned}
$$

Randomly select o_{31} from o_{21}, o_{31} with the minimum value Lf Select o_{31} and assign o_{31} to R_5.

Apply the ***schedule setting and operation removal Procedure*** to o_{31}.

$$S_{31} = 0, \quad F_{31} = 5, \quad As_{32} = As_{33} = 5,$$

Remove o_{31} from set $\bar{W}$. $\bar{W} = \{o_{11}, o_{12}, \ldots, o_{17}, o_{21}, \ldots, o_{28}, o_{32}, o_{33}, \ldots, o_{37}\}$.

Since o_{31} has an 'OR' split function, select o_{32} and remove o_{33} by the

Operation removal procedure (o_{33}, l_{31})

Remove o_{33} from sets $\bar{W}$, $S(31) = \{o_{32}, o_{33}\}$, and $P(34) = \{o_{33}\}$.

Since $l_{34} < l_{31}$ and $P(34) = \phi$, remove o_{34} from set $\bar{W}$ and $P(35) = \{o_{32}, o_{34}\}$.

Remove o_{31} from $P(32) = \{o_{31}\}$. $\bar{W} = \{o_{11}, o_{12}, \ldots, , o_{17}, o_{21}, \ldots_{28}, o_{32}, o_{35}, \ldots, , o_{37}\}$. Set $Rr_5 = F_{31}(= 5)$, $Lo_5 = o_{31}$.

Remove o_{31} from set C.

$$Lf_{111} = 111, \quad Lf_{112} = 110, \quad Lf_{115} = \max\{5 + 3(= ST(o_{3\cdot}, o_{1\cdot})), 0\} + 105 + 8 = 121,$$
$$Lf_{212} = 82, \quad Lf_{215} = \max\{5 + 2(= ST(o_{3\cdot}, o_{2\cdot})), 0\} + 7 + 77 = 91,$$

Select o_{21} and assign o_{21} to R_2.

$$S_{21} = 0, \quad F_{21} = 5, \quad As_{22} = As_{23} = 5,$$

Remove o_{21} from set $\bar{W}$. $\bar{W} = \{o_{11}, o_{12}, \ldots, o_{17}, o_{22}, \ldots, o_{28}, o_{32}, o_{35}, \ldots, o_{37}\}$. Set $Rr_2 = F_{21}(= 5)$, $Lo_2 = o_{21}$. Remove o_{21} from set C.

$$Lf_{111} = 111, \quad Lf_{112} = \max\{5 + 2, 0\} + 5 + 105 = 117, \quad Lf_{115} = 121$$

Select o_{11} and assign o_{11} to R_1.

$$S_{11} = 0, \quad F_{11} = 0 + 6 = 6, \quad As_{12} = \max\{0, 6\} = 6.$$

Remove o_{11} from set $\bar{W}$. $\bar{W} = \{o_{12}, \ldots, o_{17}, o_{22}, \ldots, o_{28}, o_{32}, o_{35}, \ldots, o_{37}\}$.

Remove o_{11} from $P(12) = \{o_{11}\}$.

Set $Rr_1 = F_{11}(= 6)$, $Lo_1 = o_{11}$.

Step (2):

$$P(12) = P(22) = P(23) = P(32) = \phi, \quad t = \min\{As_{12}, As_{22}, As_{23}, As_{32}\} = 5$$

Step (3):

$$W = \{o_{22}, o_{23}, o_{32}\}$$

Step (4):

$$mC_{22} = \min\{\max\{6 + 3, 5\} + 7,\ \max\{0, 5\} + 9,\ \max\{0, 5\} + 8\} = 13.$$
$$mC_{23} = \min\{\max\{6 + 3, 5\} + 8,\ \max\{0, 5\} + 7,\ \max\{5 + 2, 5\} + 5\} = 12.$$
$$mC_{32} = \min\{\max\{6 + 2, 5\} + 6,\ \max\{0, 5\} + 6,\ \max\{5 + 1, 5\} + 7\} = 11.$$

o_{22} requires R_4. o_{23} requires R_4. O_{32} requires R_3.

Step (5): o_{22} and o_{23} are requiring R_4 at the same time.

Therefore, initialize the Conflict set with o_{22} and o_{23}, i.e. $C = \{o_{22}, o_{23}\}$.

Step (6): Since R_3 that is required by o_{32} is the alternative resource for o_{22}, add o_{32} to Conflict set C, $C = \{o_{22}, o_{23}, o_{32}\}$.

Step (7): Solve the conflict set as like the previous case.

The iterations are continued until $\bar{W} = \phi$. The final result of the scheduling of operations is shown in Figure 8. $\sum_{i=1}^{3} F_{i\cdot} = 117$, $F_{17} = 48$, $F_{28} = 37$, $F_{37} = 32$.

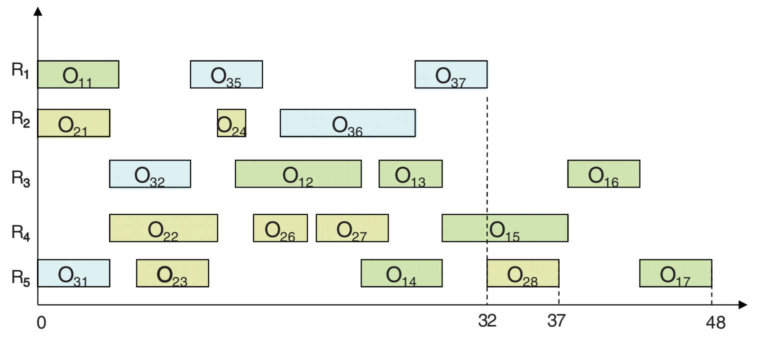

Figure 8. Gantt chart of the solution obtained by the heuristic.

5. Genetic and tabu search algorithms

5.1 *Individual representation and a schedule builder*

An individual representation that is to be applied in the genetic algorithm, tabu search, and a schedule building procedure for individuals is introduced in this section. The assignments of the operations and the processing sequences for operations are determined in an established schedule. The length of a chromosome is the total number of operations. A positive or negative integer can be the allele of the sequence portion.

The schedule-building procedure for a given chromosome is as follows. A schedule builder for a given sequence of operations is introduced in this section. The schedule builder for a priority sequence of operations makes use of the heuristic for resource assignment for operations to build a schedule.

While a priority sequence of the operations is given, there is no information on the assignment of operations and job routings in a chromosome in Figure 9. The assignment of resources for the operations and job routings are decided by a heuristic. According to the priority sequence in a chromosome, each operation is scheduled one after the other. The designated operation is dispatched to a resource with the minimum evaluated value, which is reflected by the processing time and the number of operations that can be assigned to the resources (mC_{ijk}) from the alternatives. Some of the notation is used in the schedule-building procedure. The scheduling procedures for a given priority sequence and a resource-assignment heuristic are as follows:

v_{ij}	sequence chromosome value of o_{ij}

$$f_{ijk} = \max\{Rr_k, As_{ij}\} + t_{ijk}$$

NQ_k	number of operations that can be assigned to resource k

$$mC_{ijk} = f_{ijk} + \alpha \times \sqrt{NQ_k}, \quad \alpha \geq 0.$$

Step (0): Set all operations to $\bar{W}$ and $W = \phi$.

Step (1): Set $o_{ij} \in W$ and remove o_{ij} from $\bar{W}$ for all $P(ij) = \phi$ and $o_{ij} \in \bar{W}$.

Step (2): If $W = \phi$, then stop. Else, let $m = \min_{o_{ij} \in W}\{v_{ij}\}$.

Step (3): Select o_{ia} from W according to the following rules, for all $m = v_{ij}$, $o_{ij} \in W$.

(i) The oldest operation remains in W.
(ii) The operation with the minimum processing time across its alternative resources.
(iii) A randomly selected operation.

Step (4): Calculate mC_{iak}, $\forall k \in R(ia)$.

Assign a resource k with the minimum (or maximum) value mC_{iak} to o_{ia}.
Schedule o_{ia}: $S_{ia} = \max\{Rr_k + ST_k(Lo_k, o_{ia}), As_{ia}\}$, $F_{ia} = S_{ia} + t_{iak}$.
Set $NQ_k = NQ_k - 1. \forall_{k \in R(ia)}$, $As_{ia^+} = \max\{As_{ia^+}, F_{ia}\}, o_{ia^+} \in S(ia)$.
$Rr_k = F_{ia}$, $Lo_k = o_{ia}$.

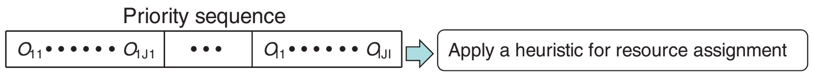
Priority sequence
O_{11} • • • • • • O_{1J1}
• • •
O_{I1} • • • • • • O_{JI}
Apply a heuristic for resource assignment

Figure 9. Structure of an individual and the assignment heuristic.

Remove o_{ia} from W.

Step (5): If $f(o_{ia}) \neq 'OR'$, then remove o_{ia} from $P(i\cdot)$ for all $o_{i\cdot} \in S(ia)$.

Else if $f(o_{ia}) = 'OR'$, select o_{ip} under the following rules.
Let $m = \min_{o_{i\cdot} \in S(ia)} \{v_{i\cdot}\}$.

(i) $m = v_{ip}$ and $o_{ip} \in S(ia)$.
(ii) Earliest completion time.
(iii) Random selection.

Remove o_{ia} from $P(ip)$ for all $o_{i\bullet} \in S(ia)$ and $o_{i\bullet} \neq o_{ip}$ and apply ***Operation removal procedure***(o_{ic}, l_{ia}) to o_{ia}.

(1) Remove o_{ic} from $\bar{W}$.
(2) Remove o_{ic} from $P(id)$ and $S(ic)$ for all $o_{ic} \in P(id),\ o_{id} \in S(ic)$.
(3) If $l_{id} < l_{ia}$ and $P(id) = \phi$, apply ***Operation removal procedure*** (o_{id}, l_{ia}) to o_{id}.

Go to Step (1).

An example is represented to give a full detail of the schedule builder for a given priority sequence as follows (Figure 10):

o_{11}	o_{12}					o_{17}	o_{21}	o_{22}						o_{28}	o_{31}	o_{32}					o_{37}
4	11	21	14	19	3	1	2	5	13	8	22	16	15	18	12	9	10	7	20	6	17

Figure 10. A chromosome representing priority sequence.

Let $\alpha = 1$.

Step (0): $\bar{W} = \{o_{11}, o_{12}, \ldots, o_{17}, o_{21}, \ldots_{28}, o_{31}, \ldots, o_{37}\},\ W = \phi$.

Step (1): Since $P(11) = P(21) = P(31) = \phi$,

set $o_{11}, o_{21}, o_{31} \in W$ and remove $o_{11},\ o_{21},\ o_{31}$ from $\bar{W}$.

Step (2): $W = \{o_{11}, o_{21}, o_{31}\}$. Let $m = \min\{v_{11}, v_{21}, v_{31}\} = \min\{4, 2, 12\} = 2$.

Step (3): Since v_{21} is a unique minimum value with m, select o_{21}.

Step (4): Calculate all of $mC_{21\bullet}$, for o_{21}.

$$NQ_1 = 11, \quad NQ_2 = 9, \quad NQ_3 = 10, \quad NQ_4 = 12, \quad NQ_5 = 11$$

$$mC_{212} = \max\{0, 0\} + 5 + \sqrt{9} = 8, \quad mC_{215} = \max\{0, 0\} + 7 + \sqrt{11} = 10.3.$$

Assign o_{21} to R_2, since mC_{212} is the minimum value.

Set $NQ_2 = 8$, $NQ_5 = 10$, $F_{21} = 5$, $As_{22} = As_{23} = 5$, $Rr_2 = 5$, $Lo_2 = o_{21}$.

Step (5): Remove o_{21} from $P(22) = \{o_{21}\}$ and $P(23) = \{o_{21}\}$.

Step (1): Since $P(22) = \phi$ and $P(23) = \phi$, set $o_{22},\ o_{23} \in W$ and remove o_{22},o_{23} from $\bar{W}$.

Step (2): Since $W = \{o_{11}, o_{31}, o_{22}, o_{23}\}$, let $m = \min\{v_{11}, v_{31}, v_{22}, v_{23}\} = \min\{4, 12, 5, 13\} = 4$.

Step (3): Since v_{11} is a unique minimum value with m, select o_{11}.

Step (4): Calculate all of $mC_{11\bullet}$, for o_{11}.

$$NQ_1 = 11, \quad NQ_2 = 8, \quad NQ_3 = 10, \quad NQ_4 = 12, \quad NQ_5 = 10$$

$$mC_{111} = \max\{0,0\} + 6 + \sqrt{11} = 9.3, \quad mC_{115} = \max\{0,0\} + 8 + \sqrt{10} = 11.2,$$

$$mC_{112} = \max\{5 + 2(= ST(Lo_2, o_{1\cdot})), 0\} + 5 + \sqrt{8} = 14.8.$$

Assign o_{11} to R_1, since mC_{111} is the minimum value.

Set $NQ_1 = 10$, $NQ_2 = 7$, $NQ_5 = 9$, $F_{11} = 6$, $As_{12} = 6$, $Rr_1 = 6$, $Lo_1 = o_{11}$.

Step (5): Remove o_{11} from $P(12) = \{o_{11}\}$.

Step (1): Since $P(12) = \phi$, set $o_{12} \in W$ and remove o_{12} from $\bar{W}$.

Step (2): Since $W = \{o_{31}, o_{22}, o_{23}, o_{12}\}$, let $m = \min\{v_{31}, v_{22}, v_{23}, v_{12}\} = \min\{12, 5, 13, 11\} = 5$.

Step (3): Since v_{22} is a unique minimum value with m, select o_{22}.

Step (4): Calculate all of $mC_{22\bullet}$, for o_{22}.

$$NQ_1 = 10, \quad NQ_2 = 7, \quad NQ_3 = 10, \quad NQ_4 = 12, \quad NQ_5 = 9,$$

$$mC_{221} = \max\{5 + 3, 5\} + 7 + \sqrt{10} = 18.2, \quad mC_{223} = \max\{0, 5\} + 9 + \sqrt{10} = 17.2,$$

$$mC_{224} = \max\{0, 5\} + 8 + \sqrt{9} = 16.$$

Assign o_{22} to R_4, since mC_{224} is the minimum value.

Set $NQ_1 = 9$, $NQ_3 = 9$, $NQ_4 = 11$,
$F_{22} = 13$, $As_{24} = As_{25} = 13$, $Rr_4 = 13$, $Lo_4 = o_{22}$.

Step (5): $f(o_{22}) = 'OR'$, $S(22) = \{o_{24}, o_{25}\}$, $m = \min\{v_{24}, v_{25}\} = \min\{8, 22\} = 8$.

Select o_{24} and remove o_{22} from $P(24) = \{o_{22}\}$.
Remove o_{25} by ***Operation removal procedure***(o_{25}, l_{22}).
Remove o_{25} from $\bar{W}$ and $P(27) = \{o_{25}, o_{26}\}$.

Step (1): Since $P(24) = \phi$, set $o_{24} \in W$ and remove o_{24} from $\bar{W}$.

Step (2): Since $W = \{o_{31}, o_{23}, o_{12}, o_{24}\}$, let $m = \min\{v_{31}, v_{23}, v_{12}, v_{24}\} = \min\{12, 13, 11, 8\} = 8$.

Step (4): Calculate all of $mC_{24\cdot}$, for o_{24}, As previous cases, assign o_{24} to R_2.

The above steps are continued until $W = \phi$. The final result is shown in Figure 11. $\sum_{i=1}^{3} F_{i\cdot} = 106$, $F_{17} = 43$, $F_{28} = 30$, $F_{37} = 33$.

5.2 *Genetic algorithm*

In this section, we present a genetic algorithm for solving the flexible job-shop scheduling problem with 'AND'/'OR' precedence constraints. Genetic algorithms are search algorithms that are developed to explain and simulate the mechanisms of natural systems. The algorithms that were first proposed by John Holland are optimisation techniques for functions that are defined over finite domains. Genetic algorithms have been widely applied for a variety of scheduling problems during the last few decades. In most cases, each gene represents an operation, while the individual represents the entire schedule. The overall structure of the proposed genetic algorithm is represented in Figure 12 and can be described as follows.

(1) *Representation*: A priority sequence is included in an individual. In this case, the resource assignment of each operation is decided based on the given sequence of operations.
(2) *Encoding*: The priority-sequence step of an individual is coded by any integer value within a predefined range. When the Partial Mapped Crossovers (PMX) operator is employed for crossover reproduction, the gene value of the priority sequence could be a unique integer from 1 to the total number of operations. The length of a chromosome is the total number of operations.

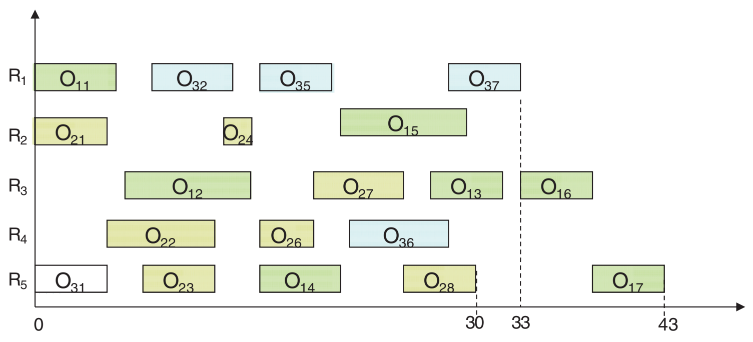

Figure 11. Gantt chart for scheduling the individual in Figure 10.

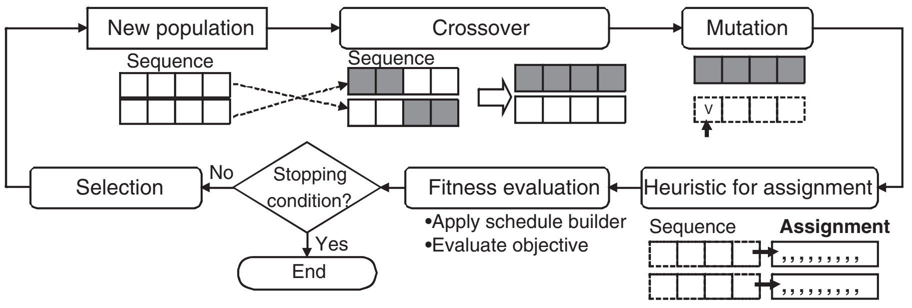

Figure 12. Structure of the genetic algorithm.

(3) *Crossover*: The crossover operates on two chromosomes at a time and generates offspring by combining features of both chromosomes. The priority sequence of the offspring made by a crossover operation should not conflict with the precedence constraints of the operations. However, one-point or two-point crossovers may yield an illegal or infeasible priority sequence for the operations. Therefore, partial mapped crossovers (PMXs) and order crossovers (OXs), etc. have been proposed for scheduling and always yield legal and feasible schedules with simple precedence constraints. However, the schedule builder in Section 5.1 does not care if illegal or infeasible sequences of operations in relation to any precedence constraints are yielded. Every crossover operator is applied for making a priority-sequence chromosome in the decoding procedures proposed in this section.

(4) *Mutation*: The mutation of the priority sequence of operations involves swapping the alleles of two randomly selected genes. Another mutation of the priority sequence of operations is an increasing or decreasing allele of the randomly selected gene.

(5) *Evaluation of fitness*: The decoding for the chromosome progresses according to the procedure of the schedule builder. The fitness of an individual is evaluated by the objective function, which may pertain to one of these measures: mean flow time, makespan, mean lateness, or mean tardiness.

(6) *Stopping condition*: The predefined number of generations is reached.

(7) *Selection*: Both fitness-proportional and tournament selection are employed for creating the new population.

5.3 *Tabu search algorithm*

The tabu search algorithm makes use of flexible memory, which is structured in accordance with strategies and aspiration conditions for exploiting search spaces. Tabu search is a meta-heuristic approach that is mainly used to find a near-optimal solution for combinatorial optimisation problems. It was introduced and formalised by Glover (1989). Tabu search starts from the current solution and creates the neighbourhoods of the current solution. At each step, these neighbourhoods are searched to find the best neighbour, and the new solution is selected as the current solution for the next step. In order to prevent cycling as well as to lead the search to good search regions in the

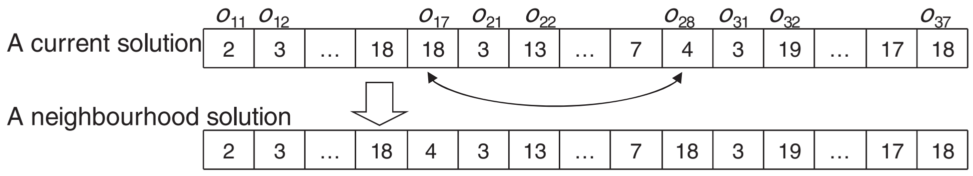

Figure 13. Neighbourhood solution by swapping two alleles.

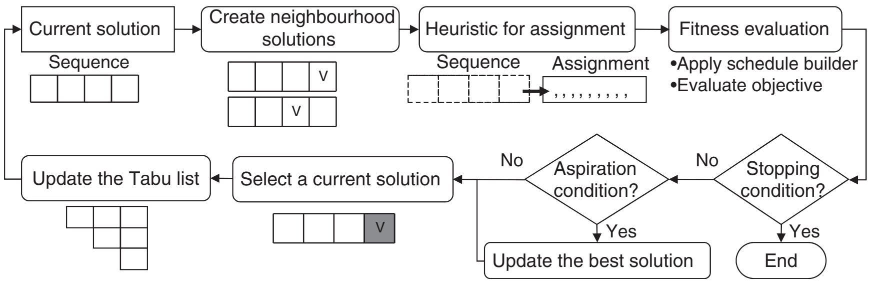

Figure 14. Structure of the tabu search algorithm.

search space, a history of searches is recorded in the tabu list. The tabu list contains the attributes of forbidden solutions that have already been the current solution in previous steps. The overall structures of the proposed tabu search algorithms are represented in Figure 14 and can be described as follows:

(1) *Representation*: The representation is identical to that of the genetic algorithm.
(2) *Initial current solution*: An initial current solution is randomly created or obtained from the heuristic method described in Section 4.
(3) *Neighbourhood*: The method of creating a neighbour is obtained by swapping the alleles of the two selected genes in the chromosome (see Figure 13).
(4) *Tabu attributes and tabu list*: A tabu list is used to prevent the search from cycling between solutions. A tabu attribute is a feature of the solution history that is maintained in the tabu list. When a neighbourhood is made by swapping the alleles of the two selected operations, the tabu list memorizes the two selected operations in the priority sequence.
(5) *Aspiration condition*: If the fitness of a neighbourhood is better than the best-so-far solution, the neighbourhood is chosen as the next current solution whether it is tabu or not.

6. Computational experiments

In this section, we have described the performances of the heuristic, genetic algorithm, and tabu search algorithms for solving several problems. Preliminary experiments were performed to tune the parameters of the proposed genetic and tabu search algorithms. The parameters of the genetic algorithm are set as follows: crossover rate $= \{0.6, 0.7, 0.8\}$ and mutation rate $= \{0.1, 0.2, 0.3\}$. The combination of a crossover rate of 0.7 and a mutation rate of 0.2 is the best setting for the genetic algorithm. In the case of tabu search, a tabu tenure of $\{10, 15, 20\}$ is set, and the best value is 15. All of the proposed methods are coded in Visual Basic, and a computer with an AMD Athlon dual-core processor 2.50 GHz was used to evaluate the performance of each method.

The first experiment evaluated the performance in terms of the total flow time of the jobs. The 30 problem instances were generated under the following conditions:

- number of jobs: 3;
- number of resources: 8~10;
- number of operations for each job: 8~10;
- number of total operations: 22~27;

- number of alternative resources for each operation: 2;
- processing times: 5~10.

The parameters of the genetic algorithm and the tabu search are listed in Table 3. The fitness for both the genetic algorithm and tabu search is the sum of the flow times of the three jobs. The best value for the objective function of the schedule for the two meta-heuristics was selected after five runs.

We examined the relative deviation with respect to each algorithm. The relative deviation is defined as $\text{dev}_{SF}(\%) = \frac{SF - SF_{MILP}}{SF_{MILP}} \times 100(\%)$, where SF is the sum of the flow times under each algorithm, and SF_{MILP} is the optimal objective value obtained using MILP solved by LINGO 10. The values of the minimum, average, and maximum relative deviations of the total flow time of jobs are listed for each method in Table 4.

In the case of the heuristic, the average relative deviation was less than 4%. The average relative deviation was less than 0.5% in the genetic algorithm and tabu search. Both the genetic algorithm and tabu search achieved optimum results for 21 and 23 instances out of 30 test problems, respectively.

The second experiment tested the performance of the methods in terms of the makespan of the test problems. Fifty problem instances were generated under the following conditions:

- number of jobs: 5;
- number of resources: 8~10;
- number of operations for each job: 10~15;
- number of total operations: 57~72;
- number of alternative resources for each operation: 2~3;
- processing times: 5~10;
- setup times are not considered.

The parameters of the genetic algorithm and tabu search were the same as those in the first experiment except that the number of iterations was 300. If the objective function of this research problem is to minimise the makespan of a schedule, the lower bound (LB) of the makespan without considering the setup times for resources is obtained by considering the following restrictions for each job:

- keeping the precedence constraints of operations in each job;
- all operations in a job are implemented on the resource with the smallest processing time;

Table 3. Parameters of the genetic and tabu search algorithms.

Genetic algorithm	Tabu search algorithm
Number of iterations: 100	Number of iterations: 100
Population size: 30	Neighbourhood operator: swapping
Selection operator: Fitness proportional	Neighbourhood size: 30
Crossover operator: PMX	Tabu tenure: 15
Crossover rate: 70%	
Mutation operator: swapping	
Mutation rate: 2%	

Table 4. Relative deviation for the total flow time of jobs.

	dev_{SF} (%)		
	Minimum	Average	Maximum
Heuristic	0	3.6	10.0
Genetic algorithm	0	0.5	3.0
Tabu search	0	0.3	2.0

Table 5. Relative deviation compared with LB for the makespan.

	Average CPU time(s)	dev_{MK} (%)		
		Minimum	Average	Maximum
Heuristic	0.03	2.00	10.96	19.00
Genetic algorithm	10.91	0	2.48	12.00
Tabu search	10.86	0	2.18	9.00

- compute b_{ij} (where b_{ij} is the total minimum processing time of o_{ij}: see Section 4) for all the first operations in jobs.

$$LB = \max_{i=1 \sim I}\{\min_{o_{ij}}\{r_i + \min\{t_{ij\cdot}\} + b_{ij}\}\}$$

where o_{ij} is the first operation in job i.

Note that the proposed lower bound may be looser as the number of resources relative to the number of operations becomes smaller. We used the following measure of the quality of the makespan for the problems; this measure was obtained as a percentage of the error related to the LB. $\text{dev}_{MK}(\%) = \{(MK - LB) / LB\} \times 100$ (%), where MK is the best makespan obtained by the five runs.

The value of the objective function of the schedule for the meta-heuristics was selected after five runs from different initial populations, which were based on the solution of the heuristic. The relative deviations of the makespan for each algorithm are shown in Table 5. The average relative deviation was less than 2.5% in the genetic algorithm and the tabu search.

The proposed methods are compared with other methods introduced thus far in the third experiment. Kim *et al.* (2003) introduced 24 test-bed problems involving 18 jobs and 15 resources. The specifications of each problem are listed in Table 6. These problems are adopted as benchmarking problems in some related papers. For each problem, five simulation runs are carried out for the heuristic, genetic algorithm and tabu search under the conditions listed in Table 7.

Table 6. Test-bed problems introduced by Kim (2003).

Problem	Number of jobs	Number of operations	Job number
1	6	79	1, 2, 3, 10, 11, 12
2	6	100	4, 5, 6, 13, 14, 15
3	6	121	7, 8, 9, 16, 17, 18
4	6	95	1, 4, 7, 10, 13, 16
5	6	96	2, 5, 8, 11, 14, 17
6	6	109	3, 6, 9, 12, 15, 18
7	6	99	1, 4, 8, 12, 15, 17
8	6	96	2, 6, 7, 10, 14, 18
9	6	105	3, 5, 9, 11, 13, 16
10	9	132	1, 2, 3, 5, 6, 10, 11, 12, 15
11	9	168	4, 7, 8, 9, 13, 14, 16, 17, 18
12	9	146	1, 4, 5, 7, 8, 10, 13, 14, 16
13	9	154	2, 3, 6, 9, 11, 12, 15, 17, 18
14	9	151	1, 2, 4, 7, 8, 12, 15, 17, 18
15	9	149	3, 5, 6, 9, 10, 11, 13, 14, 16
16	12	179	1, 2, 3, 4, 5, 6, 10, 11, 12, 13, 14, 15
17	12	221	4, 5, 6, 7, 8, 9, 13, 14, 15, 16, 17, 18
18	12	191	1, 2, 4, 5, 7, 8, 10, 11, 13, 14, 16, 17
19	12	205	2, 3, 5, 6, 8, 9, 11, 12, 14, 15, 17, 18
20	12	195	1, 2, 4, 6, 7, 8, 10, 12, 14, 15, 17, 18
21	12	201	2, 3, 5, 6, 7, 9, 10, 11, 13, 14, 16, 17
22	15	256	2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18
23	15	256	1, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18
24	18	300	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18

Table 7. Conditions of the third experiment for genetic algorithm/tabu search.

Problem	Number of iterations	Population/neighbourhood size
1∼9	400	50
10∼15	600	50
16∼21	800	50
22, 23	900	50
24	1000	50

Table 8. Comparison of best makespan.

Problem	SEA[a]	HA[b]	LB	Heuristic	Genetic algorithm	Tabu search	Genetic algorithm (Heu.)	Tabu search (Heu.)
1	428	427	200	285	203	200	200	200
2	343	343	244	286	244	244	244	244
3	347	345	196	212	205	196	196	196
4	306	306	244	285	244	244	244	244
5	319	322	198	219	201	201	201	201
6	438	429	159	202	209	198	189	173
7	372	372	244	276	244	244	244	244
8	343	343	190	209	190	190	190	190
9	428	428	198	212	198	198	198	198
10	443	430	200	276	259	260	254	232
11	369	369	244	301	260	256	247	244
12	328	327	244	264	264	250	244	244
13	452	436	167	261	286	273	248	232
14	381	380	244	320	289	272	273	246
15	434	427	198	264	231	220	224	210
16	454	446	244	360	336	329	319	292
17	431	423	244	345	351	339	315	289
18	379	377	244	316	316	308	288	260
19	490	476	198	316	352	339	298	291
20	447	432	244	357	362	345	325	297
21	477	446	198	291	335	318	284	270
22	534	518	244	424	438	418	374	353
23	498	470	244	414	426	401	380	343
24	587	544	244	451	522	494	430	398

Note: [a]Cited from Kim *et al*. (2003).
[b]Cited from Li *et al*. (2010).
(Heu.): initial solutions are composed from the solution of the heuristic.

Table 8 shows the best makespan among the 24 test-bed problems for each proposed method. The value listed in Table 8 is the minimum makespan obtained after five runs of each method for each problem. The results of the proposed methods are compared with those of SEA (symbiotic evolutionary algorithm) by Kim *et al*. (2003b) and HA (hybrid algorithm) by Li *et al*. (2010). For all 24 problems, the results of the proposed methods were superior to those of both SEA and HA. In particular, an optimum was found in cases of Problems 1, 2, 3, 4, 7, 8, 9, 11, and 12 while running the genetic algorithm and tabu search since their objective values are equal to LB.

Table 9 lists the average makespan of the 24 problems for each proposed method. The value for each method is the average makespan taken after five runs for each problem. The 'Best ever' column in Table 9 lists the best result for each problem among the SEA by Kim *et al*. (2003b), MAN (multi-agent negotiation) and HAN (hybrid-based agent negotiation) by Wong *et al*. (2006), and ANT (ant colony optimisation) by Leung *et al*. (2010). The average makespans for the proposed methods are smaller than those shown in the 'Best ever' column except for the heuristic for problems 1 and 4.

Table 10 lists both the average improvement rates and best improvement rates (figures in parentheses) for the proposed methods relative to those of the previous methods for the problem sets. The improvement rate for each proposed method is calculated as follows: {(result of previous approach – result of the proposed method)/result of previous approach} $\times$ 100 (%). The improvement rates of the two meta-heuristics are better than those of the heuristic for a small problem size. However, the improvement rates of the heuristic are better than those of the meta-heuristics for a large problem size. When the initial population of the genetic algorithm and the current solution of tabu search were created from the heuristic, the improvement rates were better than those of the general cases, and they remained relatively steady regardless of the problem size. Among all the methods described in this paper, the best option is a tabu search with the current solution from the heuristic.

Table 9. Comparison of the overall makespan.

Problem	Best ever[a]	Heuristic	Genetic algorithm	Tabu search	Genetic algorithm (Heu.)	Tabu search (Heu.)
1	282.5 (161)	*288.0* (0.1)	214.2 (20.1)	210.8 (19.2)	208.0 (19.9)	200.6 (19.3)
2	302.3 (134)	286.0 (0.1)	244.0 (20.3)	244.8 (19.7)	244.0 (20.6)	244.0 (20.1)
3	273.8 (137)	230.4 (0.1)	218.8 (22.7)	211.4 (22.2)	198.2 (23.1)	196.0 (22.6)
4	279.5 (179)	*285.0* (0.1)	247.4 (18.1)	244.8 (17.6)	244.0 (18.6)	244.0 (18.1)
5	247.0 (178)	227.8 (0.1)	201.0 (19.2)	210.6 (18.5)	206.4 (19.4)	204.2 (18.8)
6	268.8 (182)	203.2 (0.1)	211.2 (25.3)	217.4 (24.6)	190.2 (25.2)	184.8 (24.8)
7	327.5 (177)	281.4 (0.1)	248.4 (20.6)	250.0 (19.7)	249.6 (20.6)	248.8 (20.2)
8	250.8 (116)	227.4 (0.1)	197.6 (21.1)	202.8 (20.2)	192.8 (20.9)	191.2 (20.4)
9	236.2 (124)	221.4 (0.1)	198.6 (21.3)	203.0 (20.7)	198.0 (21.6)	198.0 (21.3)
10	318.8 (143)	287.8 (0.1)	286.6 (46.0)	287.6 (45.0)	268.0 (46.7)	246.2 (46.8)
11	378.9 (166)	308.8 (0.1)	289.0 (50.5)	280.2 (47.5)	254.4 (49.4)	248.4 (48.3)
12	332.8 (143)	280.0 (0.1)	278.2 (41.4)	263.6 (40.3)	254.8 (42.1)	253.2 (41.7)
13	323.3 (213)	269.0 (0.1)	296.8 (53.1)	290.2 (51.8)	248.4 (53.3)	235.4 (53.1)
14	417.2 (171)	336.8 (0.1)	307.2 (47.4)	290.4 (45.5)	290.4 (47.3)	272.0 (48.1)
15	309.4 (162)	264.0 (0.1)	250.8 (48.1)	259.6 (45.7)	229.2 (49.7)	224.0 (48.1)
16	401.5 (466)	360.6 (0.2)	360.6 (83.9)	353.8 (80.0)	329.0 (84.3)	305.0 (82.8)
17	420.8 (549)	363.0 (0.2)	382.8 (91.3)	347.0 (84.0)	325.4 (93.2)	298.8 (88.1)
18	389.6 (357)	324.8 (0.2)	351.2 (77.2)	344.2 (75.9)	294.8 (76.8)	271.6 (77.7)
19	409.8 (616)	321.4 (0.2)	380.0 (91.1)	375.2 (90.2)	307.0 (94.8)	296.6 (94.2)
20	453.8 (384)	373.8 (0.2)	387.2 (84.2)	361.6 (82.6)	339.4 (86.6)	305.8 (86.8)
21	392.8 (320)	302.6 (0.2)	329.4 (86.2)	345.2 (87.3)	294.4 (88.2)	278.6 (89.4)
22	496.5 (410)	428.0 (0.2)	465.0 (124.6)	443.4 (120.8)	394.2 (125.5)	362.0 (125.6)
23	497.8 (905)	417.2 (0.2)	466.0 (121.1)	428.0 (116.7)	391.8 (122.3)	348.4 (123.3)
24	592.5 (465)	465.6 (0.2)	545.0 (162.1)	537.4 (158.5)	438.2 (165.1)	402.6 (166.3)

Note: [a]Cited from Leung *et al.* (2010).
(Heu.): initial solutions are composed from the solution of the heuristic. ( ): CPU time (s).

Table 10. Improvement rates for each proposed method (%).

Problem	Number of jobs	Heuristic	Genetic algorithm	Tabu search	Genetic algorithm (Heu.)	Tabu search (Heu.)
1∼9	6	10.53 (56.05)	24.64 (73.72)	23.68 (76.23)	28.22 (77.40)	29.72 (79.78)
10∼15	9	18.93 (44.00)	21.68 (52.49)	24.43 (58.11)	34.55 (62.30)	40.39 (72.16)
16∼21	12	20.99 (35.86)	12.75 (30.49)	16.07 (35.38)	30.69 (46.78)	40.58 (57.66)
22, 23	15	17.66 (23.12)	6.80 (19.41)	14.14 (25.97)	26.50 (36.92)	40.02 (48.23)
24	18	27.26 (30.16)	8.72 (12.45)	10.25 (18.83)	35.21 (36.51)	47.17 (47.49)

Note: ( ): in case of the best makespan. (Heu.): initial solutions are composed from the solution of the heuristic.

7. Conclusions

In this paper, we presented several methods for the flexible job-shop scheduling problem with 'AND' and 'OR' precedence constraints. The problem we examined is an extension of the classical job-shop scheduling problem; the operations are allowed to be processed on any of the alternative resources, and there are 'OR' precedence constraints among the job operations. An MILP formulation was developed. Various objective functions can be used without considerably changing the formulation. This formulation can be used not only to compute the optimal solutions for small problems, but also to test the performance of the existing heuristic algorithms. However, solving the MILP for large problems is a very time-consuming task. To overcome this shortcoming of the MILP, a heuristic for the problem was developed in this paper. This heuristic has been proven to yield a good solution for problems, regardless of the problem's size. The developed genetic and tabu search algorithms have been proven to be effective for the problems as well. The developed meta-heuristics may be used to improve the solutions yielded by the heuristic. The improved solutions may approach a near-optimal solution to the problem. A schedule builder for an individual was introduced for the meta-heuristics in this paper, which always produce a feasible solution for the problem regardless of the reproduction operator of the meta-heuristics. The representation and schedule builders can be applied to any meta-heuristics for problems such as the present research problem. The schedule builder makes use of a priority-sequence chromosome and heuristic for the resource assignment of operations. The results of the experiments indicate that the methods in this paper are better than the previous algorithms proposed by related research. Our future research will attempt to develop other kinds of meta-heuristics and hybrid algorithms for this kind of problem.

Acknowledgements

The authors are grateful to the careful and constructive reviews from three anonymous referees. This work was supported by the Grant of the Korean Ministry of Education, Science and Technology (The Regional Core Research Program/Institute of Logistics Information Technology).

References

Brandimarte, P., 1993. Routing and scheduling in a flexible job-shop by tabu search. *Annals Operations*, 41 (3), 157–183.

Cheng, R., Gen, M., and Tsujimura, Y., 1999. A tutorial survey of job-shop scheduling problems using genetic algorithms, part II: hybrid genetic search strategies. *Computers and Industrial Engineering*, 36 (2), 343–364.

Gao, J., *et al.*, 2007. A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job-shop scheduling problems. *Computers and Industrial Engineering*, 53 (1), 149–162.

Glover, F., 1989. Tabu search-Part I. *ORSA Journal on Computing*, 1 (3), 190–206.

Hutchison, J., *et al.*, 1991. Scheduling approaches for random job-shop flexible manufacturing systems. *International Journal of Production Research*, 29 (5), 1053–1067.

Jia, H., *et al.*, 2003. A modified genetic algorithm for distributed scheduling problems. *International Journal of Intelligent Manufacturing*, 14 (3–4), 351–162.

Kacem, I., Hammadi, S., and Borne, P., 2002. Approach by localization and multi-objective evolutionary optimization for flexible job-shop scheduling problems. *IEEE Transcations on Systems. Man, and Cybernetics, Part C*, 32 (1), 1–13.

Kim, Y., Kim, J., and Lee, W., 2003a. A multi-level symbiotic evolutionary algorithm for FMS loading problems with various flexibilities. *Journal of the Korean Institute of Industrial Engineers*, 29 (1), 65–77.

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

Kim, Y., Kim, J., and Shin, K., 2004. The integration of FMS process planning and scheduling using an asymmetric multileveled symbiotic evolutionary algorithm. *Journal of the Korean Institute of Industrial Engineers*, 30 (2), 130–145.

Kim, Y., 2003. A set of data for the integration of process planning and job shop scheduling. Available from: http://syslab.chonnam.ac.kr/links/data-pp&s.doc [Accessed 1 February 2010].

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

Leung, C., *et al.*, 2010. Integrated process planning and scheduling by an agent-based ant colony optimization. *Computers and Industrial Engineering*, 59 (1), 166–180.

Li, X., Shao, X. and Gao, L., 2010. An effective hybrid algorithm for integrated process planning and scheduling. *International Journal of Production Economics*, doi:10.1016/j.ijpe. 2010.04.001.

Liang, M. and Dutta, S., 1990. A mixed-integer-programming approach to the machine loading and process planning problem in a process layout environment. *International Journal of Production Research*, 28 (8), 1471–1484.

Mastrolilli, M. and Gambardella, L., 2000. Effective neighborhood functions for the flexible job-shop problem. *Journal of Scheduling*, 3 (1), 3–20.

Moon, C., Kim, J., and Hur, S., 2002. Integrated process planning and scheduling with minimizing total tardiness in multi-plants supply chain. *Computers and Industrial Engineering*, 43 (1–2), 331–349.

Moon, I., Lee, S., and Bae, H., 2008. Genetic algorithms for job-shop scheduling problems with alternative routings. *International Journal of Production Research*, 46 (10), 2695–2705.

Nasr, N. and Elsayed, E., 1990. Job-shop scheduling with alternative machines. *International Journal of Production Research*, 28 (9), 1595–1609.

Park, J., Choi, H., and Kim, Y., 1998. A genetic algorithm approach to job-shop scheduling considering alternative process plans. *Journal of the Korean Institute of Industrial Engineers*, 24 (4), 551–558.

Wilhelm, W. and Shin, H., 1985. Effectiveness of alternative operations in a flexible manufacturing system. *International Journal of Production Research*, 23 (1), 65–79.

Wong, T., *et al.*, 2006. An agent-based negotiation approach to integrate process planning and scheduling. *International Journal of Production Research*, 44 (7), 1331–1351.

Wu, Z. and Weng, M., 2005. Multi-agent scheduling method with earliness and tardiness objective in flexible job-shops. *IEEE Transactions on System, Man, and Cybernetics-part B*, 35 (2), 293–301.

Zhang, H. and Gen, M., 2005. Multistage-based genetic algorithm for flexible job-shop scheduling problem. *Journal of Complexity International*, 11, 223–232.

Zhou, H., Feng, Y., and Han, L., 2001. The hybrid heuristic genetic algorithm for job-shop scheduling. *Computers and Industrial Engineering*, 40 (3), 191–200.