Computers & Operations Research
Manuscript Draft

Manuscript Number: COR-D-19-00455

Title: A graph-based constraint programming approach for the integrated process planning and scheduling problem

Article Type: Research Article

Keywords: Integrated process planning and scheduling; IPPS; constraint programming; graph-based modeling; AND/OR graphs.

Abstract: In this paper, the integrated process planning and scheduling (IPPS) problem is solved, with consideration of three types of flexibility. A graph-based constraint programming is proposed as a solution approach. At first, the AND/OR graph is coupled to represent IPPS instances, where the OR-relationship is used to model the exclusive alternative process sequences for a job, and the AND-relationship is used to represent the sequential relationship between an operation and its multiple immediate successors. To couple with constraint programming, the AND/OR graph is augmented by facilitating three types of dummy nodes: start and end nodes to indicate start and endpoints of jobs, and or-nodes to indicate start or endpoints of exclusive process sequences. After that, the AND/OR graph for IPPS is projected to a constraint programming model. Specific constraints are formulated to reflect sequential-, AND- and OR-structures of the graph. Constraints from problem assumptions like no-overlapping operations of the same job or operations processed by the same machine are well taken care of by existing global constraints. The graph-based constraint programming model is tested on benchmark problems. Experiment results show that the graph-based constraint programming approach works efficiently in solving the IPPS problem.

A graph-based constraint programming approach for the integrated process planning and scheduling problem

Luping Zhang[1], Chunxia Yu[2], T. N. Wong[3],
[1] Southwestern University of Finance and Economics, Chengdu, China.
nguzlp@gmail.com
[2] School of Economics and Management, China University of Petroleum, Beijing, China.
yuchunxiasd@163.com
[3] The University of Hong Kong, Pokfulam Road, Hong Kong.
tnwong@hku.hk

Abstract
In this paper, the integrated process planning and scheduling (IPPS) problem is solved, with consideration of three types of flexibility. A graph-based constraint programming is proposed as a solution approach. At first, the AND/OR graph is coupled to represent IPPS instances, where the OR-relationship is used to model the exclusive alternative process sequences for a job, and the AND-relationship is used to represent the sequential relationship between an operation and its multiple immediate successors. To couple with constraint programming, the AND/OR graph is augmented by facilitating three types of dummy nodes: start and end nodes to indicate start and endpoints of jobs, and or-nodes to indicate start or endpoints of exclusive process sequences. After that, the AND/OR graph for IPPS is projected to a constraint programming model. Specific constraints are formulated to reflect sequential-, AND- and OR-structures of the graph. Constraints from problem assumptions like no-overlapping operations of the same job or operations processed by the same machine are well taken care of by existing global constraints. The graph-based constraint programming model is tested on benchmark problems. Experiment results show that the graph-based constraint programming approach works efficiently in solving the IPPS problem.

Keywords
Integrated process planning and scheduling; IPPS; constraint programming; graph-based modeling; AND/OR graphs.

1. Introduction

Process planning and scheduling are two important functions in manufacturing industries. Process planning is to design a series of operations to finish all features of a product with consideration of sequential requirements and limited manufacturing resources. A process plan comprises the entire bundle of operations, operations' sequential constraints, and resource requirements. In conventional process planning practices, a set of multiple alternative process plans are generated. The best process plan in terms of the planning goal is then selected as the output. Subsequently, scheduling is carried out to deal with the selected process plan and allocate manufacturing resources to all necessary operations.

As described above, in conventional practices, process planning is usually carried out prior to the scheduling, making separate decisions regarding different goals. For instance, process planning module may try to minimize the total processing cost, but the scheduling attempts to minimize the makespan. Conflicts occur therefrom, which may erode the global optimality of the planning

system. Worse still, separate operations of process planning and scheduling cannot flexibly respond to occurrences of uncertainties and frequent changes in the dynamic manufacturing environment. When changes occur, the generated process plan and schedule may turn to be infeasible immediately (Leung, Wong, Mak, & Fung, 2010). Some researchers then proposed an integrated process planning and scheduling (IPPS) framework that combines the process planning and scheduling together as a single decision-making problem. As regards, IPPS allows finishing the process planning and scheduling simultaneously at a single decision-making stage.

It was well proven that traditional jobshop scheduling problems belong are NP-hard (Sotskov and Shakhlevich (1995). The integrated process planning and jobshop type of scheduling problem that combines partial process planning functions into jobshop type of scheduling is no doubt NP-hard. The computational complexity of IPPS problem expands exponentially as more jobs, operations, or machines being involved in the problem. Due NP-hardness of the IPPS problem, it is not realistic to analytically solve the problem and get exact optimal solutions with reasonable computing resources. Numerous efforts have been made on approximate approaches to get satisfying solutions with reasonable time and computing resource consumption. Local search methods, negotiation-based approaches, and meta-heuristics of this kind were proposed for the IPPS problem in last decades.

Constraint programming (CP) is a powerful emerging approach for combinatorial optimization problems. CP uses declarative expressions to state constraints and adopts separate technologies like search algorithms and constraint propagation to interpret CP models and search for feasible solutions (Rossi, Van Beek, & Walsh, 2006). The problems to be solved by CP are usually defined as constraint satisfaction problems (CSP). A CSP consists of a set of variables, a set of domains which define possible values of variables, and a set of constraints which restrict relations among variables (Barták, Salido, & Rossi, 2010). Global constraints, as defined by van Hoeve and Katriel (2006), *is a constraint that captures a relation between a non-fixed number*. Some global constraints like $nooverlap(\cdot)$, $alternative(\cdot)$ and $endbeforstart(\cdot)$ provide substantial convenience to model scheduling problems, and efficient algorithms are available in literature to solve them (Barták, 2003). However, existing literature only reported constraint programming approaches for simple scheduling problems, such as resource-constrained project scheduling (Berthold, Heinz, Lübbecke, Möhring, & Schulz, 2010), typical jobshop scheduling, or hybrid flow-shop scheduling (Baptiste, Le Pape, & Nuijten, 2012). This paper reported an approach of using constraint programming to solve the more complex IPPS problems. However, constraint programming cannot work solely as a solution approach for IPPS. With this consideration, the graph representation is incorporated.

This paper reports a hybrid graph-based constraint programming approach to solve the IPPS problem. Since the performance of CP approach heavily depends on the complexity of the model, the CP is coupled with a graph representation of the problem, which provides a way to model the IPPS problem concisely. This paper is organized as follows. In section 2, we review the most famous approximate algorithms implemented to solve the jobshop type of scheduling problems. The AND/OR graph and CP implementations in this field are also discussed. The graph-based CP model for IPPS is elaborated in Section 3. Later in section 4, a simple implementation of the proposed approach to solve a hypothetical IPPS instance is provided. Section 5 gives experiment results for a benchmark test. Finally, Section 7 provides conclusions and discussions of this paper.

2. Related work of approximate and CP approaches in IPPS research

Numerous search-based approaches to solve the jobshop type of scheduling problems can be found in literature. Typically, a simulated annealing-based algorithm was proposed for the typical job-shop scheduling problems with the objective to minimize the makespan (Pezzella, Morganti, & Ciaschetti, 2008; Steinhofel, Albrecht, & Wong, 1999). Later, a hybrid TS/SA algorithm and a genetic algorithm are proposed for the typical and flexible jobshop scheduling problems respectively (C. Y. Zhang, Li, Rao, & Guan, 2008). Most recently, García-León, Dauzère-Pérès, and Mati (2019) reported an efficient Pareto approach for the multi-objective flexible job-shop scheduling problem. The approaches for solving jobshop type of problems provide a solid foundation for constructing algorithms for the more complicated IPPS problem. MORAD and ZALZALA (1999) proposed a GA-based approach for a simple IPPS model with the consideration of alternative machines, processing costs, and a number of rejects. However, the most important factor – the alternative process routes that feature a flexible process planning and scheduling system – is not taken into account. A simulated annealing approach was proposed to settle a combined version of process planning and scheduling problem (Li & McMahon, 2007). The alternative process plans that Li and McMahon (2007) declared in their model only referred to the selectable machines and tools for processing the same operation. A particle swarm optimization approach was adopted to solve an IPPS problem, but the IPPS they depicted is much similar to Li & McMahon's model. The main contribution of their work is on the introduction of an improved evolutionary algorithm for flexible scheduling problems. Besides, the authors' research group proposed a two-staged ant colony optimization approach (Sicheng Zhang, Wong, Zhang, & Wan, 2011) and a highly competitive object-coding genetic algorithm (L. Zhang & Wong, 2015) for the IPPS problem. Most recently, Menezes, Mateus, and Ravetti (2017) proposed a branch and price algorithm for a similar integrated version of process planning and scheduling problem in bulk ports, and S. Zhang and Wong (2018) proposed an enhanced ant colony optimization approach for the IPPS problem.

In addition to search-based approaches, agent-based approaches were proposed for the IPPS. Wong, et al. (2006) proposed a Multi-Agent Negotiation approach (MAN) and a Hybrid-based Agent Negotiation approach (HAN) for the integrated process planning and scheduling/rescheduling (T. Wong, C. Leung, K. Mak, & R. Fung, 2006a; T. N. Wong, C. W. Leung, K. L. Mak, & R. Y. K. Fung, 2006). Manufacturing flexibilities including the processing flexibility were fully taken into consideration by the MAN and HAN systems.

CP approaches were also used to solve typical scheduling problems. Baptiste et al. (2012) systematically reported the CP implementations in the scheduling problem domain. In their book, a CP model for typical jobshop scheduling problems and corresponding search-based solution approaches were reported. At the early stage, CP was used solely as a solution approach for scheduling problems. Harjunkoski and Grossmann (2002) coupled an MIP model and a CP model to solve scheduling problems in two separate stages. Sadykov and Wolsey (2006) proposed a CP-based approach for a multi-machine assignment scheduling problem. Some latest literature that reported coupling CP and local search methods to increase the efficiency in solving complex scheduling problems can also be found. For instance, Beck, Feng, and Watson (2011) reported an approach which coupled CP with the taboo search for the job-shop scheduling problems (JSPs). Furthermore, the theory of implementing constraint satisfaction technologies in planning and

scheduling problems was both given by Bartak, Salido, and Rossi (2010), and the integration of planning and scheduling was even considered. However, it is just a theoretical framework and cannot be used directly in practice. Timpe (2002) established a more detailed model for general planning and scheduling problems with combined IP and CP approaches. To the best of our knowledge, until now there is no literature reporting successful applications of CP for IPPS problems.

3. The graph-based modeling for IPPS

The integration of process planning and scheduling is to select proper process plans for all jobs and allocate machining resources simultaneously at a single decision-making stage, supposing that alternative process plans have been pre-generated by other systems like CAPP (computer aided process planning system). Typically, the IPPS problem features n jobs to be processed on m machines in a jobshop type of manufacturing system. A job consists of a sequence of operations to be performed sequentially. It is possible that more than one machine can perform the same operation, and there are possibly multiple sequences of operations capable of finishing the same job. The sequential relationship among operations may exist due to mechanic engineering reasons (T. N. Wong et al., 2006).

3.1 A hypothetical IPPS instance and corresponding graph representation

Suppose that manufacturing two parts as shown in Fig. 1 – a holder on the left and a bolt on the right – is to be planned and scheduled. The holder has 5 features including 6 surfaces, a groove, a hole and chamfer to be machined. The top surface of the holder has to be specially handled due to a high level of required accuracy. The bolt has 7 features in total, including two surfaces, an outer diameter, a slot, a relief groove, a thread, and a chamfer. Suppose that a series of operations are designed to machine all features, as shown in Fig. 2 where nodes represent operations and directed arcs represent sequential constraints.

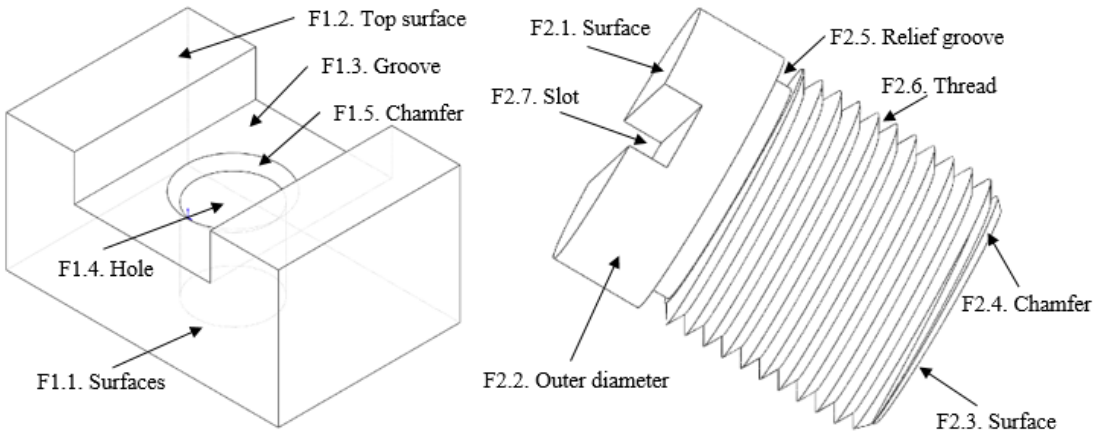

Fig. 1. A hypothetical IPPS instance with two parts.

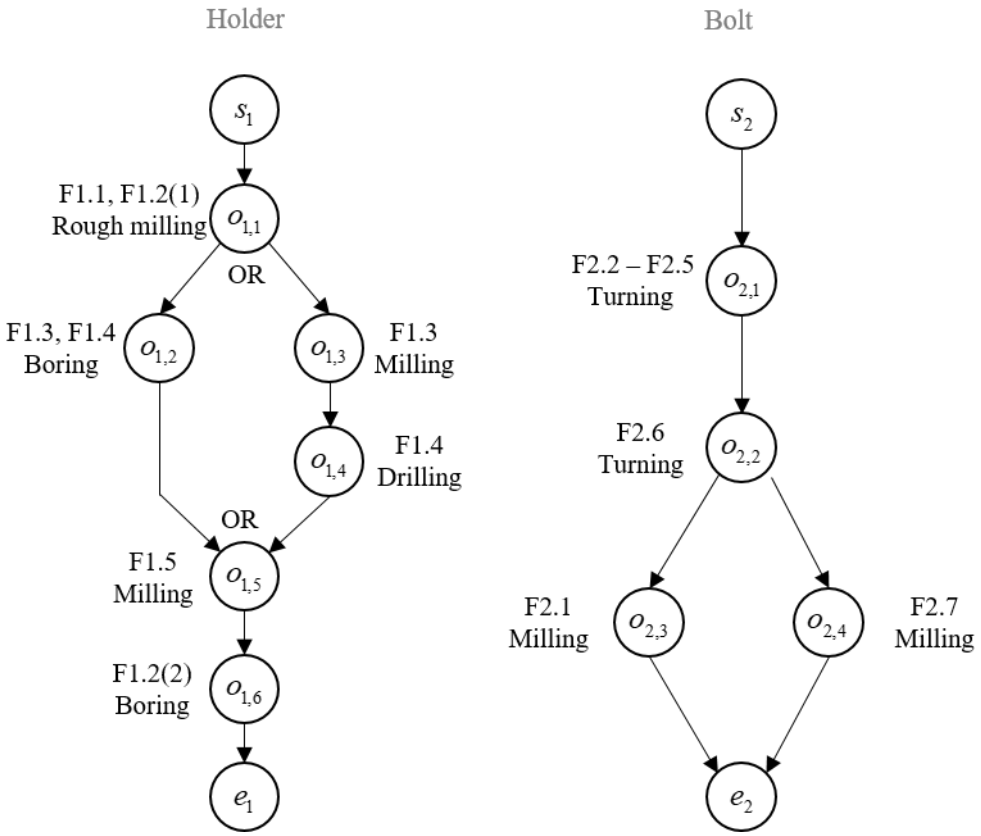

Fig. 2. Operations and sequential constraints for the hypothetical IPPS instance.

The operation $o_{1,1}$ is designed to finish Feature F1.1 and rough machining the top surface F1.2 with one setup. Since the top surface requires a higher level of accuracy, another operation $o_{1,6}$ is designed for the finish machining. Since the groove F1.3 and hole F1.4 can be done by either performing milling $O_{1,3}$ and drilling $o_{1,5}$ sequentially with a milling machine and driller respectively, or conducting operation $o_{1,2}$ on a boring machine with a single setup, one operation sequence $\left[o_{1,2}\right]$ or $\left[o_{1,3},o_{1,4}\right]$ is enough for processing the holder. Hence the two exclusive operation sequences are connected by an OR-link. That is, after performing operation $o_{1,1}$, there are two alternatives. And the alternatives end before performing $o_{1,5}$ for machining the chamfer $F1.5$ which can only be done after milling the groove and drilling the hole. Obviously, direct arcs here explicitly indicate sequential constraints.

Furthermore, in the process plan for the bolt, either milling the surface $o_{2,3}$ or milling the slot $o_{2,4}$ can be down after turning the outer diameter $o_{2,2}$, and there is no sequential constraint

between them. Hence both $o_{2,3}$ and $o_{2,4}$ are connected to $o_{2,2}$, indicating that both of them have to be performed and either can immediately be started after finishing $o_{2,2}$. The graph structure among $o_{2,2}$, $o_{2,3}$ and $o_{2,4}$ implicitly indicates an AND-relationship. Since there are both OR-relationship and AND-relationship, the graph is defined as AND/OR graph (de Mello & Sanderson, 1986; Kim, Park, & Ko, 2003).

In Fig. 2, dummy nodes s_j and e_j are introduced to indicate the start and endpoints of the job. The dummy start and end nodes clearly indicate the boundary of a job, which may be helpful in the process planning and scheduling.

Kim et al. (2003) used the AND/OR graph to represent their testbed for the integration of process planning scheduling, incorporating three types of flexibility: the processing flexibility, operation flexibility, and sequencing flexibility. Coupling the three types of flexibility and the IPPS problem illustrated with the hypothetical IPPS instance is given as follows.

- Processing flexibility to reflect alternative process sequences for the same job. As shown in Fig. 2, two alternative process sequences $\left[o_{1,1}, o_{1,2}, o_{1,5}, o_{1,6}\right]$ and $\left[o_{1,1}, o_{1,3}, o_{1,4}, o_{1,5}, o_{1,6}\right]$ are applicable for finishing the handler. What should be highlighted is that different process sequences contain different sets of operations for the same job.
- Operation flexibility to show that there may be more than one machine capable of performing the same operation. Taking operation $o_{1,4}$ as an example, drilling the hole can be done by a lathe, CNC lathe, a milling machine, or even a boring machine.
- Sequencing flexibility to show that the same set of operations can be arranged in different orders for the same job. For instance, the set of operations for the bolt can be arranged in two different sequence by putting one of $o_{2,3}$ and $o_{2,4}$ before the other, due to the fact that there are no sequence constraints between them and both of them need to be performed.

Different types of jobshop scheduling problems take different combinations of flexibility listed above. The IPPS problem is featured by taking all three types of flexibility, which renders the problem NP-hard.

The AND/OR graph representation enables establishing a concise and practical constraint programming (CP) model for the IPPS problem. compared with existing models for the IPPS type of problems like the one proposed by Qiao and Lv (2012), the graph-based CP model is much simpler and easier to be implemented in real-world problems. Later in this section, the way to project the graph representation for the IPPS to a CP model is elaborated.

3.2 Projecting the graph-based IPPS representation to a CP model

The AND/OR graph in Fig. 2 has to be augmented for the projection. An alternative sub-sequence is called a branch in this paper, and all branches for doing the same things are called a group of peer branches. To clearly identify the starting and ending points of a group of peer branch, a pair

of or-start and or-end dummy nodes are installed as shown in Fig. 3. The processing time of all dummy nodes are set to be zero, and let manufacturing the handler and bolt is Job 1 and 2 respectively. Integer numbers nodes indicate the types of nodes, which will be elaborated in Section 4.

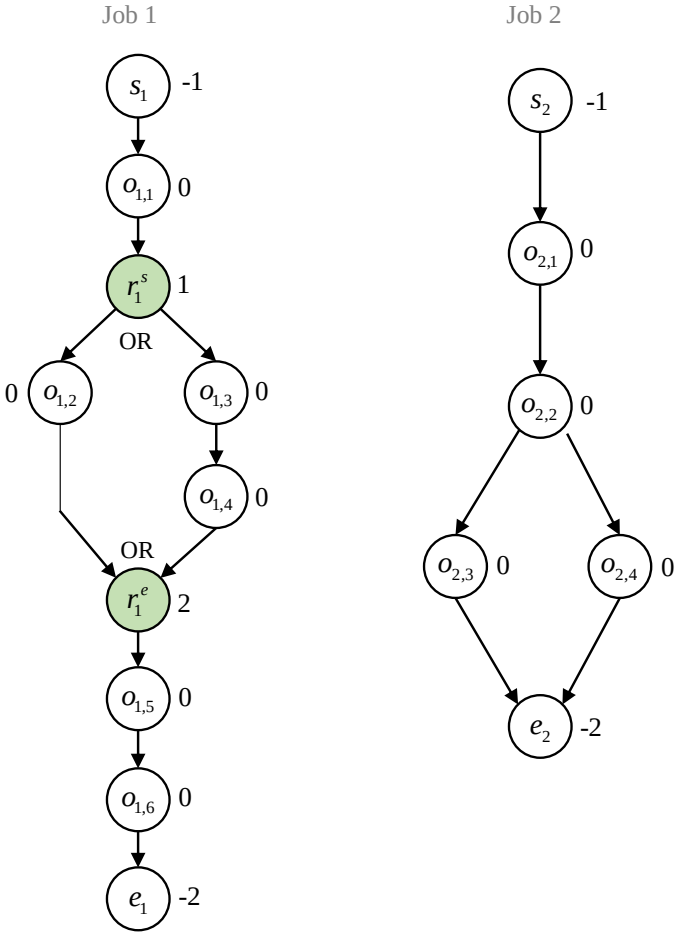

Fig. 3. Augmentation of the AND/OR graph

It is shown above the way to use the AND/OR graph to represent the IPPS problem. Formally speaking, the AND/OR graph $G=(O,A)$ mentioned in this paper is set of nodes O coupled with a set of arcs A. Nodes in the IPPS problem domain represent operations, and arcs represent sequential relationships. If there is a sequential relationship defined between two operations o_{j,k_1} and o_{j,k_2}, or say o_{j,k_2} can only be started after finishing o_{j,k_1} (denoted $<o_{j,k_1},o_{j,k_2}>$), a directed arc connecting from o_{j,k_1} to o_{j,k_2} is defined correspondingly. In this circumstance, o_{j,k_1} is an immediate predecessor of o_{j,k_2} and o_{j,k_2} is an immediate successor of o_{j,k_1}. In almost all cases, the sequential relationship is only defined between operations of the same job. Table 1 summarizes problem-related elements and their notations.

Table 1. Elements and notations for IPPS problem.

Notation	Description
J	The set of all jobs for the IPPS instance $J=\{j \mid j=1,2,3,...\}$.
M	The set of all available machines $M=\{v \mid v=1,2,3,...\}$
$o_{j,k}$	An operation of job j indexed by k.
$M\left(o_{j,k}\right)$	The set of machines capable of processing $o_{j,k}$.
O	The set of all operations designed for the IPPS instance.
O_j	The set of all operations designed for job $O_j=\{o_{i,k} \in O \mid i=j\}$.
$\left(o_{j,k},v\right)$	A tuple to indicate processing operation $o_{j,k}$ by machine v.
P	The set of all defined tuples $\left(o_{j,k},v\right)$, that is $P=\{\left(o_{j,k},v\right) \mid o_{j,k} \in O, v \in M\left(o_{j,k}\right)\}$.
$\tau\left(o_{j,k},v\right)$	The processing time of processing operation $o_{j,k}$ on machine v.
$<o_{j,k_1},o_{j,k_2}>$	An arc defining a sequential constraint between o_{j,k_1} and o_{j,k_2}, indicating that o_{j,k_1} is an immediate predecessor of o_{j,k_2}.
A	The set of all arcs.
s_j and e_j	A pair of dummy nodes indicating the start and endpoint of job j.
r_i^s and r_i^e	A pair of dummy nodes indexed by i indicating the start and end of a group of peer branches.
R^s and R^e	The set of or-start nodes and or-end nodes respectively.

3.2.1 Assumptions

The IPPS discussed in this paper follows the general assumptions made for this category of problems, that is:

(1) All operations are non-preemptive. It means that a started operation cannot be interrupted until it is finished.

(2) Operations of the same job cannot be overlapped. Since in the jobshop type of problem, operations of the same job are usually performed on a single work piece, it is impossible to conduct two operations of the same job simultaneously.

(3) A machine can perform at most one operation at a time, constrained by its capability.

(4) All jobs are released at time *zero*, and each operation can be started immediately after all its predecessors are finished.

(5) Internal logistics and setting-up consumption are neglected or absorbed by processing times of operations.

3.2.2 Decision variables

The data structure time interval which puts the start point, endpoint and duration of a time slot together as a single object is borrowed to define decision variables (Laborie & Rogerie, 2008). A time interval is usually denoted by I. Since a defined time interval may or may not be present in the final schedule, the presence of an interval I is defined as follows.

$$preof(I) = \begin{cases} 1 \ \textit{if } I \textit{ is present in the schedule} \\ 0 \ \textit{otherwise} \end{cases} \tag{1}$$

As there may be different process routes for a single job, process planning in this scenario refers to the selection of process routes for each job; while scheduling is to allocate time resources to – or say determine the start points and endpoints of time slots for – all operations on selected process routes. Two types of time intervals have to be considered in the IPPS problem instance.

(1) $I(o_{j,k}, v)$, an optional time interval (present or not in the schedule), indicating the time slot assigned for processing operation $o_{j,k}$ on machine v. Firstly, due to the aforementioned operation flexibility, the system needs to select a capable machine to process $o_{j,k}$, or say determine the presence of $I(o_{j,k}, v): v \in M(o_{j,k})$. No more than one time interval in $\{I(o_{j,k}, v) \mid v \in M(o_{j,k})\}$ can be present in the final schedule, that is

$$\sum_{v \in M(o_{j,k})} preof\left(I(o_{j,k}, v)\right) \le 1 \tag{2}$$

Secondly, due to the fact that the processing time $\tau(o_{j,k}, v)$ can be predetermined, the CP system needs to determine the start point and endpoint of the present $I(o_{j,k}, v)$. Hence the presence of $I(o_{j,k}, v)$, the start point and endpoint of it, are variables.

(2) $I(o_{j,k})$, an optional time interval, indicating the time slot assigned for processing operation $o_{j,k}$. Because of processing flexibility, $I(o_{j,k})$ may not be present in the final schedule if $o_{j,k}$ is on an OR-branch. All the presence of $I(o_{j,k})$, its start point, size, and endpoint are variables. The presence of $I(o_{j,k})$ dominates the presence of $I(o_{j,k}, v)$ in $\{I(o_{j,k}, v) \mid v \in M(o_{j,k})\}$. Specifically, once the branch including $o_{j,k}$ is selected for job j, the CP model has to assign a machine for it. By contrast, if $I(o_{j,k})$ is not present, none of elements in $\{I(o_{j,k}, v) \mid v \in M(o_{j,k})\}$ can be present. That is

$$\sum_{v \in M(o_{j,k})} preof\left(I\left(o_{j,k}, v\right)\right) = preof\left(I\left(o_{j,k}\right)\right) \tag{3}$$

3.2.3 Projecting the AND/OR graph-based IPPS to CP model

To cater to the assumption that restricts no-overlapping operations of the same job, a collection of time intervals is defined for each job, that is

$$\forall j : Q_j^J = \left\{ I\left(o_{j_0,k}, v\right) \mid o_{j_0,k} \in O_j, v \in M\left(o_{j_0,k}\right) \right\} \tag{4}$$

Q_j^J contains time intervals for all operations designed for job j. According to the assumption, all time intervals in Q_j^J must be pairwise disjoint. Likewise, to cater to the assumption that a machine can only perform one operation at a time, a collection of time intervals is defined for each machine, that is

$$\forall v : Q_v^M = \left\{ I\left(o_{j,k}, v_0\right) \mid \left(o_{j,k}, v\right) \in P, v_0 = v \right\} \tag{5}$$

Q_v^M is a set of time intervals for operations that can be performed by machine v. Q_j^J and Q_v^M later will be used for the modeling. Additional notations used for modeling and illustration are summarized in Table 2.

Table 2. Additional notations for the CP model and illustration

Notation	Description
I	A time interval.
$start(I)$	A function to get the start point of time interval I.
$end(I)$	A function to get the endpoint of time interval I.
$size(I)$	A function to get the duration of time interval I.
$preof(I)$	A variable to indicate the presence or absence of time interval I in the final schedule.
$I\left(o_{j,k}\right)$	A time interval variable for operation $o_{j,k}$. The start point, endpoint and size are to be determined.
$I\left(o_{j,k}, v\right)$	A time interval variable for the processing $\left(o_{j,k}, v\right)$. The size of $I\left(o_{j,k}, v\right)$ is equal to the processing time $\tau\left(o_{j,k}, v\right)$, and the start point and endpoint are to be determined.
Q_j^J	A set of time intervals for job j.
Q_v^M	A set of time intervals for machine v.

$c_{\max}$	The makespan.

Well-defined global constraints are used to establish a concise model which can be dealt with by existing algorithms. Fig. 3 shows an illustration of projection from AND/OR graph-based IPPS to CP constraints, where the alternative machine constraint and no-overlapping constraint for machines are not shown yet.

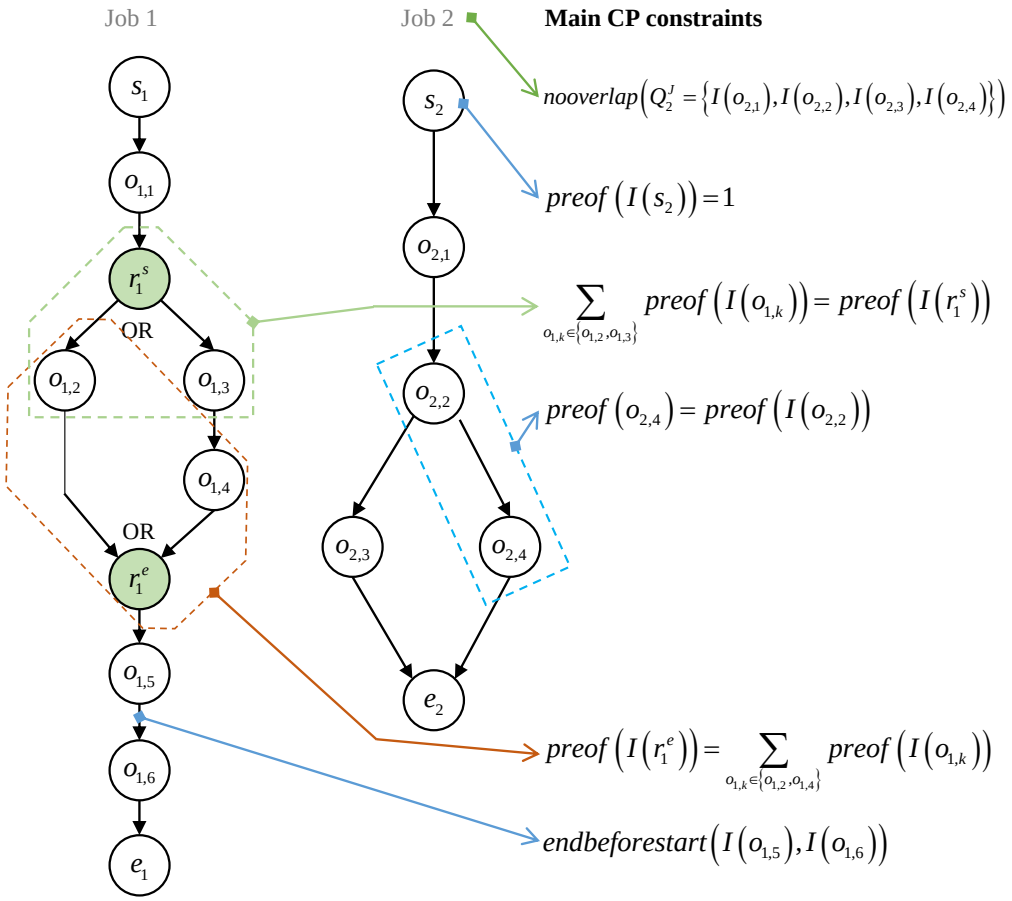

Fig. 3. An illustration of projecting AND/OR graph-based IPPS to CP constraints.

Following part provides the explanation of projection.

(1) $nooverlap\left(Q_2^J=\left\{I\left(o_{2,1}\right),I\left(o_{2,2}\right),I\left(o_{2,3}\right),I\left(o_{2,4}\right)\right\}\right)$, a constraint for job 2 corresponding to the no-overlapping assumption made in section 3.21. Since job 2 for producing the bolt is to be performed on the single work piece, time intervals assigned for all operations of job 2 cannot overlap each other, implying that it is unnecessary to perform two operations on a single work piece simultaneously.

(2) $preof\left(I\left(s_2\right)\right)=1$, a constraint to regulate that job 2 needs to be scheduled. Presences of other operations of job 2 are regulated by other constraints.

(3) $\sum_{o_{1,k}\in\{o_{1,2},o_{1,3}\}} preof\left(I\left(o_{1,k}\right)\right)=preof\left(I\left(r_i^s\right)\right)$, a constraint to regulate that once an or-start

node is present, exactly one of its successor has to be present. By contrast, if an or-start node is not present, neither its successors can be present. The presence of the or-start node is controlled by its predecessors.

(4) $preof\left(o_{2,4}\right) = preof\left(I\left(o_{2,2}\right)\right)$, a constraint to regulate the presence of a successor if the arc is not involved in any or-relationship. It means that if both predecessor and successor defined by an arc are not or or-nodes, then this pair of nodes have to be present or absent together.

(5) $preof\left(I\left(r_i^e\right)\right) = \sum_{o_{1,k} \in \left\{o_{1,2}, o_{1,4}\right\}} preof\left(I\left(o_{1,k}\right)\right)$, a constraint to regulate the presence of an or-end node according to the presences of its predecessors. It means that if one of its predecessors is present, the or-end node has to be present. By contrast, if none of its predecessors are present, the or-end node cannot be present.

(6) $endbeforestart\left(I\left(o_{1,5}\right), I\left(o_{1,6}\right)\right)$, a constraint for all sequential constraint defined by arcs. It means that the start point of $I\left(o_{1,6}\right)$ must be greater than or equal to the endpoint of $I\left(o_{1,5}\right)$, or say, $end\left(I\left(o_{1,5}\right)\right) \le start\left(I\left(o_{1,6}\right)\right)$.

A general form of the CP model can be written as follows.

The Graph-Based CP Model for IPPS

$$\forall \langle o_1, o_2 \rangle \in A : endbeforestart\left(I\left(o_1\right), I\left(o_2\right)\right) \tag{6}$$

$$\forall j \in J : nooverlap\left(Q_j^J\right) \tag{7}$$

$$\forall v \in M : nooverlap\left(Q_v^M\right) \tag{8}$$

$$\forall j \in J, o_{j,k} \in O_j : alternative\left(I\left(o_{j,k}\right), \left\{I\left(o_{j,k}, v\right) \mid v \in M\left(o_{j,k}\right)\right\}\right) \tag{9}$$

$$\forall r_i^s \in R^s : \sum_{o_{j,k} \in O, s.t. \langle r_i^s, o_{j,k} \rangle \in A} preof\left(I\left(o_{j,k}\right)\right) = preof\left(I\left(r_i^s\right)\right) \tag{10}$$

$$\forall r_i^e \in R^e : preof\left(I\left(r_i^e\right)\right) = \sum_{o_{j,k} \in O, s.t. \langle o_{j,k}, r_i^e \rangle \in A} preof\left(I\left(o_{j,k}\right)\right) \tag{11}$$

$$\forall \langle o_1, o_2 \rangle \in A, o_1 \notin R^s, o_2 \notin R^e : preof\left(I\left(o_1\right)\right) = preof\left(I\left(o_2\right)\right) \tag{12}$$

$$\forall j \in J : preof\left(I\left(s_j\right)\right) = 1 \tag{13}$$

Constraint (6) forces the sequential relationship between any pair of nodes connected by directed arcs. Constraint (7) and (8) reflect the no-overlapping assumptions stated in Section 3.2.1 respectively. Constraint (9) means that only one machine can be selected to process an operation. The $alternative\left(I, \left\{I_1, I_2, ...\right\}\right)$ is a well-defined global constraint which models an exclusive alternative among the set of time intervals $\left\{I_1, I_2, ...\right\}$. If the time interval I is present, exactly

one time interval in $\{I_1, I_2, ...\}$ will be present, and I will be equal to the present I_i in the set.

Constraint (10) deals with the presence relationship between an or-start node and its immediate successors. Similarly, constraint (11) takes care of the presence relationship between an or-end node and its immediate predecessors. Constraint (12) deals with the presence relationship between any pair of nodes connected by arcs and neither of them is an or-node. Constraint (13) points out that all jobs have to be scheduled.

4. Implementation

In this section, the graph-based CP model is implemented to solve the hypothetical IPPS instance in Section 3. Hypothetically, suppose that there are 5 machines available, including a lathe (M1), a CNC lathe (M2), a milling machine (M3), a boring machine (M4), and drilling machine (M5). The capability of machines and corresponding processing times are listed in Table 3. And basic data of the hypothetical IPPS instance are listed in Table 4. A dummy machine M0 is allocated to process all dummy nodes $\{s_j \mid j \in J\}$, $\{e_j \mid j \in J\}$, R^s, and R^e with zero processing time.

Table 3. Processing data for the hypothetical IPPS instance.

Operation $o_{j,k}$	Capable machines $M(o_{j,k})$	Processing time $\tau(o_{j,k}, v)$	Operation $o_{j,k}$	Capable machines $M(o_{j,k})$	Processing time $\tau(o_{j,k}, v)$
$o_{1,1}$	M3, M4	30, 24	$o_{1,6}$	M4	16
$o_{1,2}$	M4	16	$o_{2,1}$	M1, M2	34, 14
$o_{1,3}$	M3	8	$o_{2,2}$	M1, M2	18, 14
$o_{1,4}$	M1, M2, M5	18, 10, 14	$o_{2,3}$	M3	16
$o_{1,5}$	M3, M4	14, 10	$o_{2,4}$	M3	12

Table 4. Basic data for the hypothetical IPPS instance.

Notation	Data for the hypothetical IPPS instance
J	$J = \{1,2\}$.
M	$M = \{1,2,3,4,5\}$.
O	$\{o_{1,1}, o_{1,2}, o_{1,3}, o_{1,4}, o_{1,5}, o_{1,6}, o_{2,1}, o_{2,2}, o_{2,3}, o_{2,4}\}$.
$M(o_{j,k})$	$M(o_{1,1}) = \{3,4\}$, $M(o_{1,2}) = \{4\}$, $M(o_{1,3}) = \{3\}$, $M(o_{1,4}) = \{1,2,5\}$, etc.

O_j	$O_1=\{o_{1,1},o_{1,2},o_{1,3},o_{1,4},o_{1,5},o_{1,6}\}$, $O_2=\{o_{2,1},o_{2,2},o_{2,3},o_{2,4}\}$.
P	$P=\{(o_{1,1},3),(o_{1,1},4),(o_{1,2},4),(o_{1,3},3),(o_{1,4},1)...\}$.
A	$A=\left\{\begin{array}{l}<s_1,o_{1,1}>,<o_{1,1},o_{1,2}>,<o_{1,1},o_{1,3}>,<o_{1,2},o_{1,5}>,<o_{1,3},o_{1,4}>,<o_{1,4},o_{1,5}>,<o_{1,5},o_{1,6}>,\\<o_{1,6},e_1>,<s_2,o_{2,1}>,<o_{2,1},o_{2,2}>,<o_{2,2},o_{2,3}>,<o_{2,2},o_{2,4}>,<o_{2,3},e_2>,<o_{2,4},e_2>\end{array}\right\}$.
R^s & R^e	$R^s=\{r_1^s\}$, $R^e=\{r_1^e\}$.

To make it simpler for application, a type parameter $T(o_i)$ is designated to each node, in which case the system just needs to maintain one set of operations and identify the types of nodes by using the type parameter. One definition of type parameter is defined as follows, which can simplify programming constraint (9) to (13) in practices.

$$T(o_i)=\begin{cases}-1 \ \textit{if } o_i \textit{ is a start node, that is } o_i\in\{s_j \mid j\in J\}\\ -2 \ \textit{if } o_i \textit{ is an end node, that is } o_i\in\{e_j \mid j\in J\}\\ 1 \quad \textit{if } o_i \textit{ is an or-start node, that is } o_i\in R^s\\ 2 \quad \textit{if } o_i \textit{ is an or-end node, that is } o_i\in R^e\\ 0 \quad \textit{otherwise}\end{cases} \tag{14}$$

Based on the Type definition, the graph-based CP model can be rewritten as

The Applied Graph-Based CP model for IPPS

$$\forall\langle o_1,o_2\rangle\in A: endbeforestart(I(o_1),I(o_2)) \tag{15}$$

$$\forall j\in J: nooverlap(Q_j^J) \tag{16}$$

$$\forall v\in M: nooverlap(Q_v^M) \tag{17}$$

$$\forall j\in J, o_{j,k}\in O_j: alternative\left(I(o_{j,k}),\{I(o_{j,k},v)\mid v\in M(o_{j,k})\}\right) \tag{18}$$

$$\forall o_i\in O, T(o_i)=1: \sum_{o_k\in O,s.t.\langle o_i,o_k\rangle\in A} preof(I(o_k))=preof(I(o_i)) \tag{19}$$

$$\forall o_i\in O, T(o_i)=2: preof(I(o_i))=\sum_{o_k\in O,s.t.\langle o_k,o_i\rangle\in A} preof(I(o_k)) \tag{20}$$

$$\forall\langle o_1,o_2\rangle\in A, T(o_1)\neq 1, T(o_2)\neq 2: preof(I(o_1))=preof(I(o_2)) \tag{21}$$

$$\forall o_i\in O, T(o_i)=-1: preof(I(o_i))=1 \tag{22}$$

Taking minimizing the makespan as the single objective, that is

$$\min\max\{end(I(o_i))\mid o_i\in O, T(o_i)=-2\} \tag{23}$$

the model is programmed with IBM CPLEX optimization studio, and the FailLimit is set to be 20,000. An optimal schedule generated by the model is depicted in Fig. 4, where all dummy operations are neglected. It is easy to check that all sequential constraints are satisfied. The system just used the CNC lathe (M2), the milling machine (M3), and the boring machine (M4) for the processing. It is meaningful since the three machines are much more efficient than the lathe (M1) and the drilling machine (M5) according to the data in Table 3.

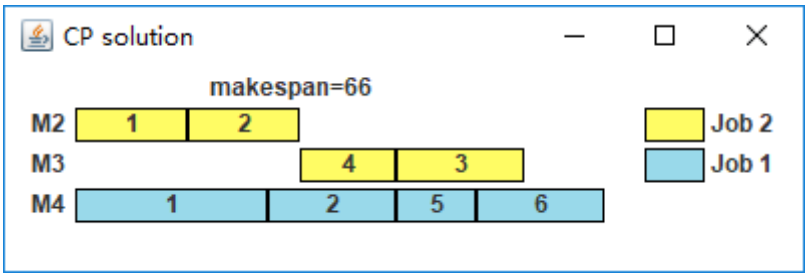

Fig. 4. Gantt Chart for the hypothetical IPPS instance.

5. Experiments and discussions

A testbed proposed by Kim, et al. is adopted for experiments (Kim et al., 2003). 18 jobs consisting of more than 300 operations are combined in different ways, thereby 24 problems of different level of flexibilities are generated.

A Desktop facilitated with I5-4460 CPU working at 3.2G (each core) and 8G RAM is used to conduct the experiments. IBM ILOG CPLEX optimization studio version 12.5 is used to program the model.

5.1 Experiment 1 – global performance test

In this experiment, a parameter *Failimit* to control the termination of the search procedure is assigned. It is to test the performance of the model with given computing resources. The FailLimit for all problems in this experiment is set to be 200,000. Since the CPLEX solver implements a deterministic search strategy, each problem of the testbed is executed only once and the objective values and running times are recorded. The obtained results are compared with those found by the symbiotic evolutionary algorithm (SEA) (Kim et al., 2003), the Improved Genetic Algorithm (IGA) (Lihong & Shengping, 2012), and the Object-Coding Genetic Algorithm (OCGA) (L. Zhang & Wong, 2015) .

Table 5. Makespans found by SEA, IGA, OCGA and graph-based CP (GCP).

Problem	Lower bound	SEA	IGA	OCGA	GCP	Best	Improved rate(%)	CPU time of GCP (s)
1	427	437.6	427	427	427	-	-	1.2
2	343	349.7	344.5	343.5	343	GCP	0.1%	8.4
3	344	355.2	351	346.4	344	GCP	0.7%	8.8
4	306	306.2	307.4	310.1	306	GCP	0.1%	8.3
5	304	323.7	309.8	323	319	IGA	-3.0%	8.5
6	427	443.8	427	427	427	-	-	1.6
7	372	372.4	372.7	373.3	372	GCP	0.1%	8.5

8	342	348.3	357	343.5	343	GCP	0.1%	9.4
9	427	434.9	427	427	427	-	-	1.7
10	427	456.5	431.6	427.1	427	-	-	2.6
11	344	378.9	379.7	350.6	345	GCP	1.6%	9.5
12	306	332.8	323.7	324.7	318	GCP	1.8%	9.6
13	427	469	442.8	427.2	427	-	-	2.7
14	372	402.4	415.3	377.4	372	GCP	1.4%	9.5
15	427	445.2	427.4	427	427	-	-	2.3
16	427	478.8	449.4	428.1	427	GCP	0.3%	3.5
17	344	448.9	426	383.1	356	GCP	7.1%	11.4
18	306	389.6	373.6	354.5	346	GCP	2.4%	10.9
19	427	508.1	471.3	433.7	427	GCP	1.5%	5
20	372	453.8	446.6	390.5	372	GCP	4.7%	9.8
21	427	483.2	447.8	427	427	-	-	4
22	427	548.3	508.1	452.9	427	GCP	5.7%	8.8
23	372	507.5	477.8	410	395	GCP	3.7%	11
24	427	602.2	548.5	471.2	457	GCP	3.0%	11

Table 5 reports makespans found by different approaches for the 24 problems. Lower bounds of each problem and CPU times of the graph-based CP approach are listed in the table for reference. The graph-based CP approach finds optimal solutions to 16 out of the 24 problems for which lower bounds are reached. For simpler problems number 1 to 16, the improvement is not significant since lower bounds are nearly reached by the OCGA. For more complex problems number 17 to 24, the quality of solutions is significantly improved. The CPU times of GCP show that the graph-based CP approach works efficiently on the testbed, and near-optimal solutions can be found quickly in no more than 12 seconds in the given experiment environment.

The search processes for solving four representative problems problem number 2, 15, 21 and 24 are shown in Fig. 5, where the horizontal axis represents the timeline and the vertical axis represents the makespans of solutions. It shows that the search procedure converges quickly to good solutions.

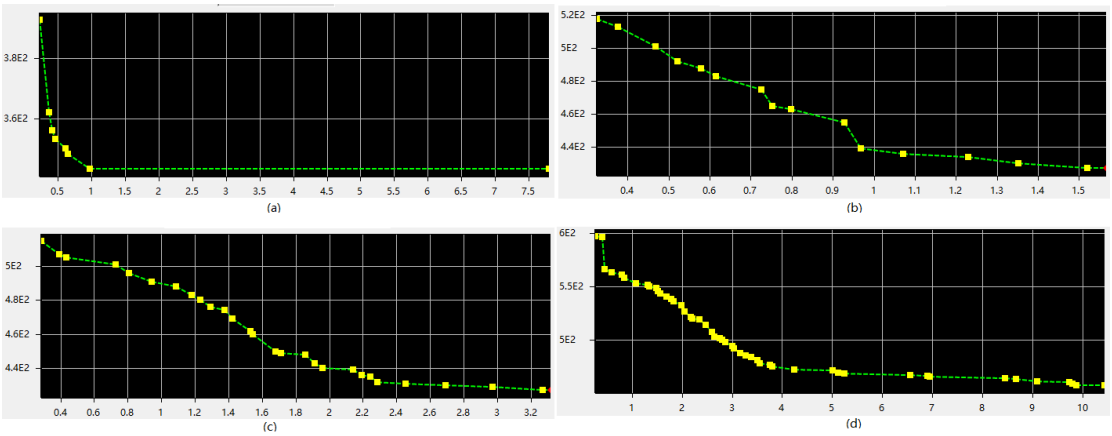

Fig. 5. The convergence procedure of solving problem 2 (a), 15 (b), 21 (c) and 24 (d)

5.2 Experiment 2 – extreme test

The FailLimit restriction is removed to test the extreme potential of the graph-based CP model. In this experiment, only those problems of which lower bounds are not reached are solved in the new setting-up. The results of the extreme test are listed in Table 6.

Table 6. Results found in extreme experiments.

Problem	Makespan	Total run time(s)	Best result reached time (s)
5	318	>1000	7.5
8	343	>1000	0.8
11	345	>1000	2.3
12	318	>1000	2.1
17	344	>1000	46
18	344	>1000	10
23	372	>1000	168
24	427	75	75

It shows that the solver runs for a much longer time if it is not terminated manually. What is interesting is that the search procedure of solving the most complex problem of the testbed problem number 24 terminates in 75 seconds, indicating that an optimal solution is found. One optimal schedule for problem number 24 is shown in Fig. 6. Besides, optimal solutions to problem number 17 and 23 are also found in extreme tests.

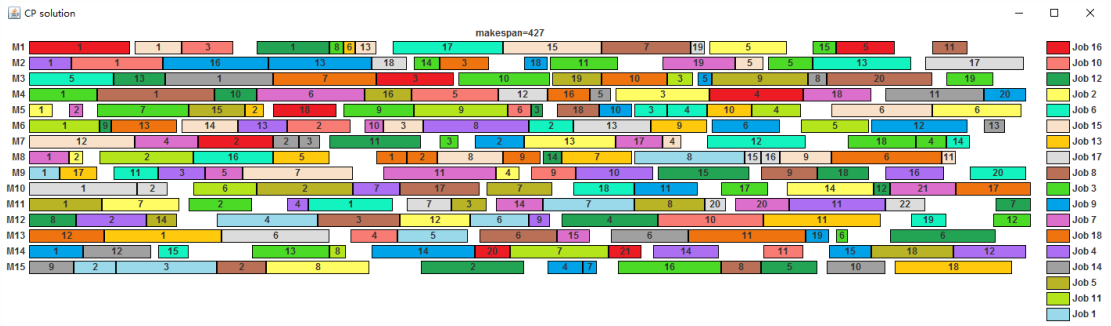

Fig. 6. An optimal schedule for problem 24 with makespan=427

6. Conclusions and discussions

In this paper, the integrated process planning and scheduling (IPPS) problems (Kim et al., 2003; T. Wong, C. Leung, K. Mak, & R. Fung, 2006b) with consideration of three types of flexibility are solved. At first, an AND/OR graph representation for IPPS is proposed, where the OR-relationship is used to represent exclusive alternative process sequences for the same job, and AND-relationship is used to represent that a single operation has multiple immediate successors all of which have to be performed.

To establish the constraint programming model, the original AND/OR graph is augmented by facilitating dummy or-nodes to explicitly indicate the start point and endpoint of a group of exclusive or-branches. Besides, another two types of dummy nodes – the start and end nodes indicating the start and endpoints of jobs are facilitated.

An approach to project elements of an AND/OR graph to a constraint programming (CP) constraints is provided. With regard to the structure of AND/OR graph itself, the presence of

or-nodes and their successors or predecessors are carefully handled. There is nothing special to deal with AND-relationship. The presences of two nodes connected by an arc are equal if there are no or-nodes involved, even if AND-relationship is involved. Other constraints are provided to deal with non-overlapping assumptions for the set of operations of the same job or the set of operations being processed by the same machine. The non-preemptive assumption is taken care of by the definition of time interval: the size of a time interval is equal to the difference between its endpoint and start point, in which case a started operation cannot be preempted.

A graph-based CP model is provided. In the implementation, the graph-based CP model can be further simplified by introducing a type parameter for each operation, in which case the model just needs to deal with a single operation set but identify different types of operations with their types.

Experiments are conducted on a benchmark test. The IBM ILOG CPLEX optimization studio is used to perform experiments. The graph-based CP outperforms the other three heuristic algorithms in almost all benchmark problems. The graph-based CP model even yields optimal a solution for the most complex problem of the testbed.

This paper mainly contributes to a hybrid approach that combines graph representation and constraint programming for the IPPS problem. The IPPS belongs to a category of difficult combinatorial optimization problems. The graph representation and simple augmentation enable formulating the IPPS with existing constraint programming technologies. The concise graph-based CP model has great potential to be used in practices.

Besides, the projection approach is another significant contribution to modeling the type of problems which can be represented with graphs, since the graph is *locally* projected to constraints. The locality means that each constraint just deals with relationships between operations connected by directed arcs. Taking the hypothetical IPPS instance, the or-relationship among r_1^s, $o_{1,2}$, and $o_{1,3}$ is local, since it only involves two directed arcs $<r_1^s, o_{1,2}>$ and $<r_1^s, o_{1,3}>$ that connect them. So corresponding constraints are only related to these two arcs. To project graphs locally has some benefits. First of all, the structure of a graph could be quite complicated, in which case to project the graph globally – or say to write a constraint with the involvement of too many graph's elements – is difficult. Secondly, it is easier to make any modification to the structure of a graph. Thirdly, the presence relationship in the graph-based CP model is transitive. That is, the presence of an interval is determined by its immediate predecessors whose presences are also determined by their immediate predecessors, and so on. That is why it just needs to force the presence of start nodes of each job explicitly in the model. The transitivity of presence relationship enables establishment of efficient algorithms to solve this model.

Incorporating the graph-based CP approach to solve extended IPPS problems with setup consideration, internal logistics, or dynamic disruptions are recommended in future work. Besides, as the expansion of problem scales, it possibly needs customized algorithms to solve CP models for IPPS type of problems.

Acknowledgment

The authors would like to acknowledge the financial support of the National Natural Science Foundation of China (No. 71501187).

Reference

Baptiste, P., Le Pape, C., & Nuijten, W. (2012). *Constraint-based scheduling: applying constraint programming to scheduling problems* (Vol. 39): Springer Science & Business Media.

Barták, R. (2003). *Constraint-based scheduling: An introduction for newcomers.* Paper presented at the Intelligent manufacturing systems.

Bartak, R., Salido, M. A., & Rossi, F. (2010). Constraint satisfaction techniques in planning and scheduling. *Journal of Intelligent Manufacturing, 21*(1), 5-15. doi:DOI 10.1007/s10845-008-0203-4

Barták, R., Salido, M. A., & Rossi, F. (2010). Constraint satisfaction techniques in planning and scheduling. *Journal of Intelligent Manufacturing, 21*(1), 5-15.

Beck, J. C., Feng, T. K., & Watson, J. P. (2011). Combining Constraint Programming and Local Search for Job-Shop Scheduling. *Informs Journal on Computing, 23*(1), 1-14. doi:DOI 10.1287/ijoc.1100.0388

Berthold, T., Heinz, S., Lübbecke, M. E., Möhring, R. H., & Schulz, J. (2010). *A constraint integer programming approach for resource-constrained project scheduling.* Paper presented at the International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming.

de Mello, L. S., & Sanderson, A. C. (1986). *AND/OR Graph Representation of Assembly Plans*. Retrieved from

García-León, A. A., Dauzère-Pérès, S., & Mati, Y. (2019). An efficient Pareto approach for solving the multi-objective flexible job-shop scheduling problem with regular criteria. *Computers & Operations Research, 108*, 187-200. doi:https://doi.org/10.1016/j.cor.2019.04.012

Harjunkoski, I., & Grossmann, I. E. (2002). Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. *Computers & Chemical Engineering, 26*(11), 1533-1552.

Kim, Y. K., Park, K., & Ko, J. (2003). A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling. *Computers & Operations Research, 30*(8), 1151-1171. doi:https://doi.org/10.1016/S0305-0548(02)00063-1

Laborie, P., & Rogerie, J. (2008). *Reasoning with Conditional Time-Intervals.* Paper presented at the FLAIRS conference.

Li, W. D., & McMahon, C. A. (2007). A simulated annealing-based optimization approach for integrated process planning and scheduling. *International Journal of Computer Integrated Manufacturing, 20*(1), 80-95. doi:Doi 10.1080/09511920600667366

Lihong, Q., & Shengping, L. (2012). An improved genetic algorithm for integrated process planning and scheduling. *The International Journal of Advanced Manufacturing Technology, 58*(5-8), 727-740.

Menezes, G. C., Mateus, G. R., & Ravetti, M. G. (2017). A branch and price algorithm to solve the integrated production planning and scheduling in bulk ports. *European Journal of Operational Research, 258*(3), 926-937.

MORAD, N., & ZALZALA, A. (1999). Genetic algorithms in integrated process planning and scheduling. *Journal of Intelligent Manufacturing, 10*(2), 169-179. doi:10.1023/a:1008976720878

Pezzella, F., Morganti, G., & Ciaschetti, G. (2008). A genetic algorithm for the flexible job-shop scheduling problem. *Computers & Operations Research, 35*(10), 3202-3212.

Qiao, L., & Lv, S. (2012). An improved genetic algorithm for integrated process planning and scheduling.

The International Journal of Advanced Manufacturing Technology, 58(5-8), 727-740. doi:10.1007/s00170-011-3409-0

Rossi, F., Van Beek, P., & Walsh, T. (2006). *Handbook of constraint programming*: Elsevier.

Sadykov, R., & Wolsey, L. A. (2006). Integer programming and constraint programming in solving a multimachine assignment scheduling problem with deadlines and release dates. *INFORMS Journal on Computing, 18*(2), 209-217.

Sotskov, Y. N., & Shakhlevich, N. V. (1995). NP-hardness of shop-scheduling problems with three jobs. *Discrete Applied Mathematics, 59*(3), 237-266. doi:http://dx.doi.org/10.1016/0166-218X(95)80004-N

Steinhofel, K., Albrecht, A., & Wong, C. K. (1999). Two simulated annealing-based heuristics for the job shop scheduling problem. *European Journal of Operational Research, 118*(3), 524-548.

Timpe, C. (2002). Solving planning and scheduling problems with combined integer and constraint programming. *Or Spectrum, 24*(4), 431-448.

van Hoeve, W.-J., & Katriel, I. (2006). Global constraints. In *Foundations of Artificial Intelligence* (Vol. 2, pp. 169-208): Elsevier.

Wong, T., Leung, C., Mak, K., & Fung, R. (2006a). Integrated process planning and scheduling/rescheduling-An agent-based approach. *International Journal of Production Research*.

Wong, T., Leung, C., Mak, K., & Fung, R. (2006b). Integrated process planning and scheduling/rescheduling—an agent-based approach. *International Journal of Production Research, 44*(18-19), 3627-3655.

Wong, T. N., Leung, C. W., Mak, K. L., & Fung, R. Y. K. (2006). Dynamic shopfloor scheduling in multi-agent manufacturing systems. *Expert Systems with Applications, 31*(3), 486-494. doi:DOI 10.1016/j.eswa.2005.09.073

Zhang, C. Y., Li, P., Rao, Y., & Guan, Z. (2008). A very fast TS/SA algorithm for the job shop scheduling problem. *Computers & Operations Research, 35*(1), 282-294.

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

Zhang, S., Wong, T., Zhang, L., & Wan, S. (2011). *A two-stage approach based on ant colony optimization algorithm for integrated process planning and scheduling.* Paper presented at the Proceedings of the 41st International Conference on Computers & Industrial Engineering.

Zhang, S., & Wong, T. N. (2018). Integrated process planning and scheduling: an enhanced ant colony optimization heuristic with parameter tuning. *Journal of Intelligent Manufacturing, 29*(3), 585-601. doi:10.1007/s10845-014-1023-3

Highlights (for review)

- The integrated process planning and scheduling (IPPS) is solved.
- The AND/OR graph is used to represent IPPS problem.
- The AND/OR graph is augmented for constraint programming formulation.
- A graph-based constraint programming model is proposed for IPPS.
- Benchmark problems are used to test the proposed approach.