Journal of Universal Computer Science, vol. 31, no. 12 (2025), 1349-1385
submitted: 18/1/2025, accepted: 25/4/2025, appeared: 28/10/2025 CC BY 4.0

A SIPyOC-RC-based Computing Framework for Complex IPPS Problem Modelling and Solving

Zhichao Chen
(School of Software, Shanghai Jiao Tong University, Shanghai, China,
COMNOVA Co., Ltd., Shanghai, China,
czchaohao@126.com)

Zixi Han
(COMNOVA Co., Ltd., Shanghai, China,
https://orcid.org/0000-0001-6097-8471, zixi.han@outlook.com)

Bingqing Shen
(Department of Data Science and Big Data Technology, Shanghai International Studies University, Shanghai, China,
bqshen@shisu.edu.cn)

Yuxin Zeng
(School of Software, Shanghai Jiao Tong University, Shanghai, China,
zengyuxin@sjtu.edu.cn)

Min Wang
(COMNOVA Co., Ltd., Shanghai, China,
wangmin.luck@foxmail.com)

Hongming Cai*
(School of Software, Shanghai Jiao Tong University, Shanghai, China,
hmcai@sjtu.edu.cn)

Minqi Wang
(COMNOVA Co., Ltd., Shanghai, China,
wangminqi@comac.cc)

Abstract: The Integrated Process Planning and Scheduling (IPPS) problem has long been a practical concern in industry, with numerous problem modelling and solving methods to optimize multiple objectives, such as makespan and cost. As yet, existing modelling methods have difficulties in satisfying new modelling requirements for adequately representing real complex manufacturing jobs. This paper proposes a novel computing framework for modelling and solving complex IPPS problems based on a Supplier-Input-Process-yield-Output-Customer-Resource-Control (called SIPyOC-RC) model. This framework includes a SIPyOC-RC modelling method to address the modelling adequacy problem. With the proposed method, complex IPPS problems in real manufacturing scenarios, such as commercial airplane component production and assembly, can be modeled with the key demanding features. Next, the SIPyOC-RC model can be transformed into a Constraint Programming (CP) model, which can then be solved with any CP-solver. Based

*corresponding author

on an implemented testbed, the modelling adequacy and computation performance of the proposed approach has been evaluated. The experiment results show that it can successfully solve complex IPPS problems without significant computation cost. Moreover, its compatibility with conventional IPPS problems is also discussed. Overall, the outcome of this study contributes both theoretical methodology and practical experience to industrial computing.

Keywords: Complex product manufacturing system, IPPS, modelling method, SIPOC
Categories: J.6, I.1.1
DOI: 10.3897/jucs.146407

1 Introduction

Process planning and scheduling plays an important role in intelligent manufacturing systems [Sugimura et al., 2001]. Usually, the scheduling procedure is separated from process planning, which leads to poor resource coordination and conflicts with the change in resource availability and production demand [Li et al., 2010]. Consequently, issues that can change resource availability such as forced downtime can significantly impact production efficiency and raise safety risks, which have become increasingly severe with production complexity. Integrated Process Planning and Scheduling (IPPS) aims at integrating process planning and scheduling to optimize the utilization of production resources, reduce production cycles and costs, and ensure product quality and delivery reliability. It considers the high complexity of job workflows, task dependencies, resource constraints, and uncertainty factors [Phanden et al., 2011].

To represent the IPPS problem, various modelling approaches have been proposed, including AND-OR digraphs [Lin and Solberg, 1991], feature tables [Li et al., 2012], network graphs [Liu et al., 2021], etc. Nevertheless, complex product (e.g., commercial aircraft) manufacturing job has some additional characteristics. First, tasks have complex relations. Two tasks can be in a sequential, parallel, alternative, or nested relation. Also, two sequential tasks may have a waiting time in between them. Second, executing a task involves various constraints, such as resource available time and a special resource control condition. Moreover, manufacturing resources selection is more flexible. For example, a task can be done by a group of workers with different qualification types. Due to the above characteristics, existing IPPS modelling methods cannot adequately describe the entire process of complex product manufacturing with limited modelling capability. Based on a low accurate model, the problem solution may be impractical to a real manufacturing scenario or actually non-optimal. This problem is called the "modelling adequacy problem".

Modern process improvement theories have been widely applied in the manufacturing of complex products. Among them, the SIPOC (suppliers, inputs, process, outputs, customers) model [Jin and Zhang, 2019] has been widely used for managing and improving complex processes as well as identifying core processes. The SIPOC network diagram consists of five elements (Supplier, Input, Process, Output, and Customer), which represent many factors involved in the process. It helps identify the optimal path to achieve goals from a panoramic perspective and can easily expand the controlling variables. However, it still lacks some elements in the IPPS problem, such as different kinds of resources like machines or workers, controlling conditions of processes, and precedence relation between processes.

To address the modelling adequacy problem in complex product manufacturing, this paper proposes a new IPPS problem modelling and solving approach. The key idea of

the approach is based on a new modelling method, called SIPyOC-RC (Supplier-Input-Process-yield-Output-Customer-Resource-Control). The SIPyOC-RC model extends the SIPOC model by adding the precedence (y) edge, resource (R) node, and control (C) node. These additional elements enable the description of complex task relations, various task constraints, and flexible resource selection through an integrated view of manufacturing factors along job life cycle. This approach enhances the understanding of business process modelling and provides practical insights for managing and optimizing processes in a manufacturing context. Additionally, the approach is implemented and evaluated through experiments and analysis with the real case data of commercial airplane component production and assembly (called the real manufacturing case in the sequel).

The main contributions of this paper are threefold:

1. After studying the key modelling features in complex product manufacturing, a new modelling method, called SIPyOC-RC, has been proposed to address the modelling adequacy problem. With the SIPyOC-RC method, job modelling has been demonstrated with two typical real manufacturing examples.

2. The transformation scheme from the SIPyOC-RC-based job model to the constraint programming (CP) model has been devised with a new model transformation algorithm design, which can be further computed with a CP-solver to obtain optimal solutions.

3. With an implemented computing system, the applicability, expressiveness, and computational performance of the approach has been evaluated, showing its advantage over existing modelling methods in IPPS problem solving. The results indicate that the proposed approach has the chance to find more optimal solutions than existing ones with limited performance cost.

The rest of this paper is organized as follows. In Section 2, the related work will be reviewed first. After that, a solution overview is provided in Section 3 after studying and summarizing modelling requirement for complex product manufacturing. Section 4 elaborates the proposed SIPyOC-RC-based IPPS problem modelling and solution approach, which is exemplified with two real manufacturing jobs. The proposed approach is evaluated and compared with existing modelling methods in Section 5. Finally, Section 6 will provide the conclusion and outlines future work.

2 Related Work

Before the presentation and discussion of the proposed model, relevant studies including other modelling methods of IPPS problems will be reviewed in this section. Since the modelling method can also influence the solving of IPPS problems (especially in the preprocessing procedure), mainstream solutions to the IPPS problems will be reviewed first. Also, since the SIPOC model is the foundation of the proposed model, it will be reviewed in the last part of this section.

2.1 Mainstream Solutions to IPPS Problems

In the aspect of the number of objectives, IPPS problems can be classified into single-objective IPPS problems and multi-objective IPPS problems. From the feature of the problem statement, IPPS problems can also be classified into normal IPPS problems, IPPS

problems under uncertain disturbances, and distributed IPPS problems. The methods for single-objective IPPS or related problems are well-established, and mainstream methods include traditional integer programming methods [Jin et al., 2016] and heuristic algorithms [Łapczyńska et al., 2022, Li et al., 2014, Li et al., 2009, Bucki et al., 2015], etc.

For multi-objective IPPS problems, [Shokouhi, 2018] using weighted methods and genetic algorithms, [Wen et al., 2021] proposed an NSGA-II-based method for solving multi-objective IPPS problems. To address the challenges posed by the dynamic production environment in actual manufacturing processes, some researchers have focused on researching IPPS problems under uncertain disturbances. For example, [Liu et al., 2016] considered uncertain events of new job arrivals and designed a dynamic IPPS solution method based on the ant colony optimization algorithm. [Li et al., 2019] used interval numbers to describe uncertain processing times and designed a hybrid genetic swarm optimization algorithm. [Elgendy et al., 2023] developed a parallel distributed Genetic Algorithm (GA) to solve to optimize a multi-objective Flexible Job shop Scheduling Problem (FJSP) with dynamic events (which means that the tasks and the service providers can be arriving or withdrawing at any time). [Zhou et al., 2023] proposed a multi-objective scheduling model based on cloud-edge computing with dynamic availability of machines, and solve it with an improved NSGA-II algorithm. In some cases, the scheduling problems needs to be solved in distributed manufacturing systems, leading to the emergence of distributed IPPS problems. For example, [Lin et al., 2020] presented a method based on the GA framework to search the solution space of distributed IPPS problems by generating load-balancing solutions.

Although significant progress has been made, existing IPPS technologies still cannot be effectively applied in large-scale, highly constrained, and frequently changing production processes, such as civil aircraft manufacturing, due to their limitation in problem modelling.

2.2 Manufacturing Process Modelling

Process modelling is important to the smart manufacturing system, which describes and analyzes each link and operation of the production process. These models can help understand and optimize production processes to improve productivity, quality, and operability. Some early methods use tables to record machines, workpieces, processes, and processing time to achieve the goal of describing the machine flexibility [Li et al., 2012]. More sophisticated, a network flow diagram is used to describe the process path flexibility of the manufacturing system. Manufacturing equipment, workpieces, and processes are divided into several classes to generate multiple alternative processing paths [De Giovanni and Pezzella, 2010]. For example, [Benjaafar and Ramakrishnan, 1996] proposed a set theory model to describe machine flexibility, sequence flexibility, and process path flexibility, which can determine the labor order, features, and usable machines. [Kiritsis and Porchet, 1996] proposed a flexible description method based on Petri nets to provide an efficient simulation tool for process planning.

The studies on IPPS problems mainly employ feature tables and network graphs to describe manufacturing processes. Typically, a feature table records the product features, operations, alternatives of machines, and processing time in the manufacturing process, which are then converted to design variables, constraints, parameters, and objectives [Li et al., 2012], as shown in Table 1. On the other hand, a network graph uses a directed graph to represent the relationship between tasks, as in Fig. 1. Optimal solutions can be obtained by removing unnecessary precedence constraints [Liu et al., 2021]. With such

a representation, the IPPS problem can be solved using techniques such as topological sorting, critical path analysis, and scheduling algorithms. Alternatively, a mixed integer linear programming model can be used to transform the IPPS problem into mathematical formulas that defines objective functions, constraints, and design variables [Jin and Zhang, 2019].

Feature	Operations	Alter. machines	Process time	Precedence
F_1	O_1	M_3,M_8	8,13	Before F_2,F_3
F_2	O_2-O_3	M_5,M_6,M_8/M_2	16,12,13/21	Before F_3
F_3	O_4-O_5	M_1,M_5,M_{10}/M_9	13,16,18/17	

Table 1: An example of feature table [Li et al., 2012].

The above modelling methods can be selected or combined according to the specific situations and requirements to describe and analyze the production process. Nevertheless, there is a lack of a general modelling method for different kinds of manufacturing problems. For example, the constraints within the following scenarios are difficult to express using existing mainstream modelling methods:

- One task needs to be executed only after or only within a certain period of time after another task is finished. For example, after applying the adhesive, two components must be assembled together before the adhesive dries.
- Whether one task can be executed depends on the availability of its input material, and whether one task has to be executed depends on the availability of its output material. For example, if certain semi-finished products are already in inventory, then some tasks of a job do not need to be executed. This depends on the inventory status and the input and output of each task.
- Some tasks can be influenced by external factors, such as supplier or customer. For example, the start time of a task can be influenced by the availability of the input material, which is provided by the supplier. The finish time of a final task can be constrained by the delivery time of the product, which is determined by the customer.
- Some tasks have special controlling conditions, which can influence the executability of the task. For example, if the quality of a part does not meet the requirement, the remaining process of the part can be skipped. As another example, a particular process of two parts can be simultaneously carried out, thereby utilizing only one set of resources for two parts during simultaneous processing.

Existing modelling methods are more solution-coupled, which limits their capability to describe highly complex scenarios. Thus, a universal modelling approach is needed to model the entire manufacturing process and support more efficient production optimization and decision making.

2.3 SIPOC Model

The theory of process improvement and quality management is originally developed to enhance organizational efficiency, quality, and performance. Among a vast amount

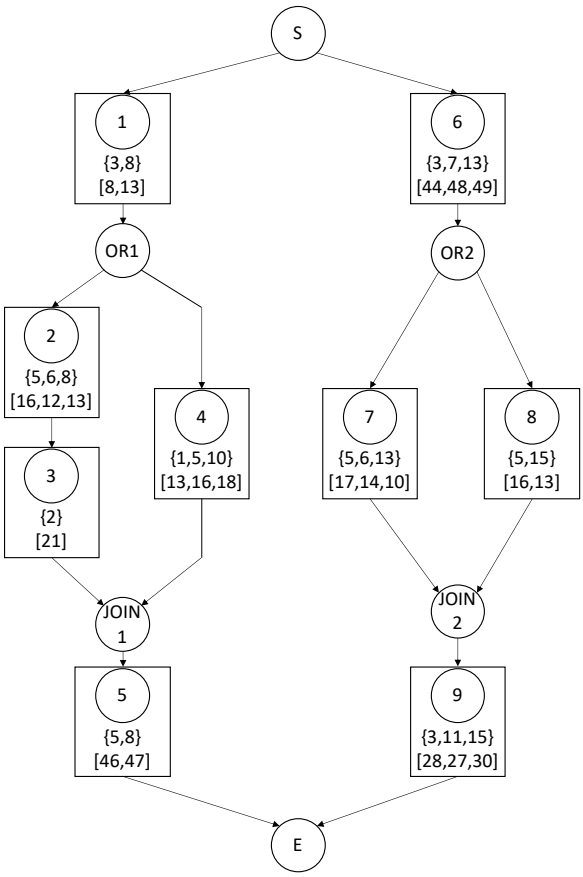


Figure 1: An example of a network graph [Liu et al., 2021]. The number in circle represents the operation number. The number in square brackets represents the machine number, and the number in curly brackets represents the processing time.

of tools, the SIPOC model [Mishra and Kumar Sharma, 2014] is used to describe and improve business processes, which also has the potential to describe and analyze manufacturing process. It visually displays the key elements of a process, including Supplier, Input, Process, Output, and Customer, helping people understand and manage complex processes. The framework of the SIPOC model is shown in Fig. 2. The SIPOC model clarifies the boundaries of the process, helping to understand the scope and elements within the process. It also aids in identifying and understanding key steps, enabling teams to focus on key points for improvement. Additionally, the SIPOC model facilitates information sharing and coordination among team members, promoting overall organizational effectiveness. The SIPOC model finds wide application in various areas. In project management, for example, the SIPOC diagram can serve as a tool during project initiation and planning phases to help teams clarify project objectives and scope [Marques and Requeijo, 2009]. In supply chain management, the SIPOC diagram can be used to analyze critical links and information flow within the supply chain, promoting

coordination and optimization [Mishra and Kumar Sharma, 2014]. Note that the SIPOC model already possesses many elements necessary for a manufacturing system.

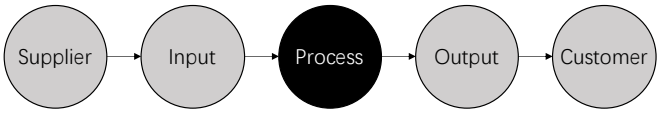


Figure 2: The framework of SIPOC model.

However, the existing SIPOC model are not capable of describing various types of factors and constraints along complex manufacturing processes. First, it only provides a simple description of the process, lacking in-depth analysis. Second, the model does not consider the time dimension, failing to reflect time-related sequencing and issues. Moreover, due to its primary focus on identifying and describing process elements, the SIPOC model lacks support for quantitative data analysis.

3 Solution Overview

To identify modelling requirements, the IPPS problem of complex product manufacturing is studied first from the real manufacturing case, which is described in [Anonymous, 2024]. The problem study focuses on the demanded modelling features from two typical scenarios, a production line example and an assembly line example. Then, a new framework is proposed in this section.

3.1 Problem Study

3.1.1 Complex Job Modelling

Complex job modelling needs to describe task relationship, identify redundant tasks, and represent execution conditions. Salient challenges are found in the real production line example (which is called Example 1 in the sequel). It contains 1409 tasks with 41 resources used by them. The execution of a task includes one or multiple resources.

First, the precedence of two tasks determines the input-output relationship of them, which forms a job execution route. Existing modelling methods introduces OR/AND node and its corresponding JOIN node to describe branches of routes, called the "branch-based" methods. Nevertheless, they cannot express complicated cases. For example, in Kim's dataset [Kim et al., 2003], the AND and OR branches can be intricately intertwined when trying to describe precedence relation of Process 17 in Job 5.

Second, it is important to consider semi-finished/finished parts in inventory to reduce the length of a production job. In the real production example, Task #9705a324 has an output 9a64ef6d_1, which is the input of Task #364dc792. But in the inventory, there is already a part 9a64ef6d_1, which means that Task #9705a324, together with its preceding tasks, can be skipped to minimize the total makespan.

In task execution, special control conditions also need to be described by a model. For example, a hot working machine can simultaneously process 3 A-shaped parts and 2 B-shaped parts collected from different tasks, but not 4 A-shaped parts as shown in Fig. 3. It means the tasks sharing the machine have to begin and end at the same time. In

the real production example, Resource #693d7c4a is such type of machine, and Task #9cb31d58", for instance, needs slot "A" of this resource. Also, in this problem, the number of every slot is 1.

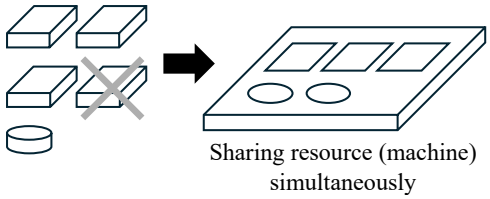


Figure 3: A resource (machine) with multiple slots of different shapes. Tasks using this machine can be executed simultaneously as long as the shapes of the parts match the shapes of the slots.

3.1.2 Flexible Resource Selection

In an assembly line (which is called Example 2 in the sequel), workers collaborate to build a complex mechanical system. A real assembly example contains 910 tasks with 174 workers and 17 zones as resources. In this example, Task #16910961 should be executed by three workers, but nine workers of different types are capable of performing this task, showing high flexibility of human resource selection. Generally, a task may need to be executed by workers with different qualification types, which requires a selection of resources from multiple types. For example, in Fig. 4, Task 1 requires workers of qualification type 1, which is possessed by Worker A, B, C, and D. Conversely, Task 2 requires workers with qualification type 2, which is possessed by A, B, E, and F. It can be learned that Task 1 has 6 alternative human resource combinations, which significantly increases the complexity of the IPPS problem. Note that, generally, the resource selection is actually a type of processing method selection in scheduling problems, which means that different processing method has different resource usage.

3.1.3 Various Time Constraints

In the assembly example, some tasks have specified delivery time (e.g., due date), which means that they should be finished as early as required. For example, Task #d04d7866 should be finished at time 96. In business, these tasks are either critical or delivery tasks, and therefore require specific milestones or delivery time. Actually, there exists a specific category of problems that focus on the study of due dates, known as "due-date assignment" problems [Demir et al., 2024].

In a production line, similarly, some raw material may not be available at the beginning, but will be available later, which will influence the start time of certain tasks. In the real production example, Task #0bee2d8f cannot be executed before time 65.0.

Also in production, two consecutive tasks are executed with a period of time in between them, called waiting time. Particularly there are two cases. A task may be executed within a certain period of time after its preceding task is finished. which is the maximum waiting time constraint. In the real production example, Task #3c8a9df0

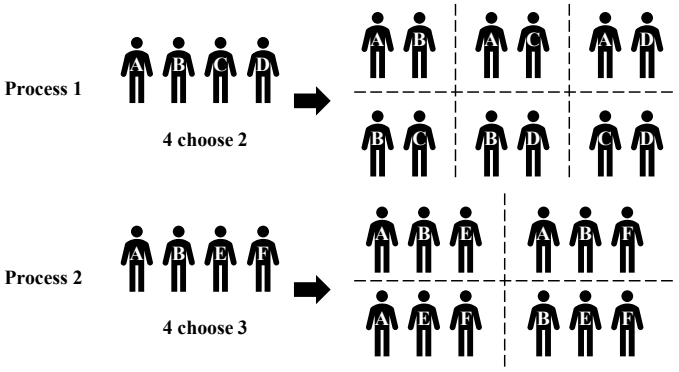


Figure 4: An example of "flexible selection of resources", which allows the arbitrary selection of a given number of resources from a set of specific resources. Note that in this example, Worker A and B can be used in both processes, while other workers can only be used in one of the processes.

should be executed within 4.0 time units after Task #68472bf9. In contrast, one task can only be executed after a certain duration has elapsed following the completion of its preceding task, which is the minimum waiting time constraint. It describes the necessary gap between two tasks or the moving time of workpiece.

3.2 Modelling Requirements

From the aforementioned two examples, one can see that they impose partially overlapping yet distinct demands on the modelling method. The demanded modelling features, together with manufacturing factors, are summarized in Table 2. Specifically,

- Example 1 has five features. The OR type precedence relation and the time requirement between tasks require an approach to describe the complex precedence relation. The semi-finished parts can be clearly described with the assistance of the input and the output of certain tasks. The available time is related with the supplier. The special controlling conditions require a controlling approach to influence the behavior of the task.
- Example 2 has two features, which are the flexible selection of resources and the delivery time. The flexible selection of resources requires a controlling approach to influence the behavior of the resource, while the delivery time is related with the customer.

Therefore, one can see that the real scheduling problems in smart manufacturing can be very complicated. On the other hand, existing modelling methods fail to satisfy the above requirements. Thus, a new modelling method is needed.

3.3 Overall Framework

To address the modelling problems in real manufacturing scenarios, a new computing framework for IPPS problem modelling and solving SIPyOC-RC is proposed based on a

	Example 1	Example 2
Supplier		✓
Input		✓
Process	✓	✓
Output		✓
Customer	✓	
Resource	✓	✓
Controlling	✓(influencing resource)	✓(influencing process)
Precedence Relationship	✓	✓(complex)

Table 2: The demand on necessary features in the modelling method for two examples.

SIPyOC-RC modelling method, as illustrated in Fig. 5. The SIPyOC-RC model extends the conventional SIPOC model by integrating key manufacturing factors, including the ones summarized in Table 2, and their relations. In the SIPyOC-RC modelling method, a task is described as a SIPyOC-RC unit, containing multiple related attributes. Next, with obtained manufacturing information, including order, workflow, process specification, resource state, inventory level, etc., a specific manufacturing job is modeled by combining multiple SIPyOC-RC units. To solve the problem, the SIPyOC-RC-based job model is then converted to a CP model by identifying variables, parameters, and objective function(s), and constraints from the SIPyOC-RC model. The conversion result is a set of mathematical equations that can be solved by a CP-solver, such as COPT and IBM CPLEX, to search for an optimal solution. Finally, the CP-solver outputs a job execution schedule, including task sequence, resource allocation, and execution time, which can be visualized through a Gantt Chart.

Notably, the optimization algorithm design is out of the scope of this study and only an existing CP-solver is applied in mathematic problem solving. Thus, this paper mainly focuses on SIPyOC-RC modelling definition, job modelling, and model transformation.

4 Proposed Approach

To overcome the limitations of existing modelling methods, a new modelling method, SIPyOC-RC, is proposed, followed by applying it to model the typical jobs in the real manufacturing case. Then, a model transformation scheme is provided to convert a SIPyOC-RC job to a CP model. Finally, the CP problem is solved within a CP-solver to obtain optimal makespan.

4.1 SIPyOC-RC Model Definition

In the SIPyOC-RC model, a task of a manufacturing job is represented by seven elements:

$$task := (S, I, P, y, O, C, R, C) \tag{1}$$

where P (Process) identifies a task and specifies the process to accomplish the task; I (Input) is the input of the task, especially the expendable resources, such as raw materials and assembly parts; O (Output) is the output of the task, such as processed workpiece

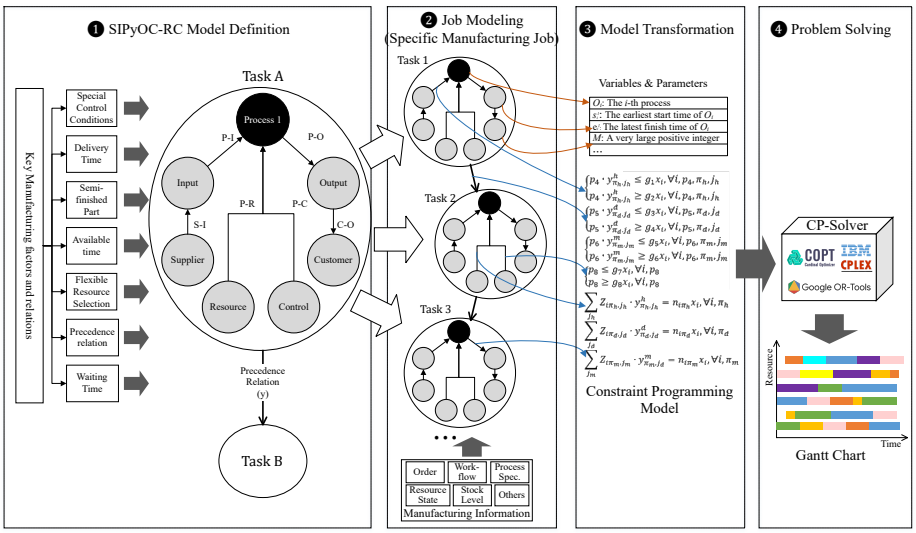


Figure 5: A SIPyOC-RC-based computing framework IPPS problem modelling and solving. (A enlarged version can be seen in Appendix A.)

and assembled parts; S (Supplier) is the supplier of the input; the first C (Customer) is the customer of the output; R (Resource) contains the resources necessary to perform the task, especially the non-expendable ones, including machine, human, fixture, etc; the second C (Control) is the control condition of the task, which influences the executability of the task or the behavior of other nodes; and y (yield) describes the precedence relation between two processes. Notably, within the context of SIPyOC-RC modeling, a task is a basic unit of the model, which includes a process and other components. The process within SIPyOC-RC specifies the processing method and procedure of the task. Without the context of SIPyOC-RC modeling, task and process are two interchangeable terms.

It is important to identify the relations of these elements, which are defined by the following predicates:

- $Exec(\cdot)$: $Exec(P_i)$ is true if and only if Task P_i is executed.
- $Occu(\cdot)$: $Occu(R_i)$ is true if and only if R_i (which is the collection of resources that can be used by Task P_i) is occupied. $Occu(R_{i,k})$ is true if and only if $R_{i,k}$ (which is the set of resources used by the k-th processing method of Task P_i) is occupied.
- $Avail(\cdot)$: $Avail(I_i)$ is true if and only if Input I_i is available.
- $Valid(\cdot)$: $Valid(O_i)$ is true if and only if Output O_i is valid.
- $Exist(\cdot)$: $Exist(y(P_i, P_j))$ is true if and only if there is a precedence relation from Task P_i to Task P_j.
- $Qualify(\cdot)$: $Qualify(I_i)$ is true if and only if Input I_i satisfies the quality requirement.
- $ruleQualify(\cdot,\cdot)$: $ruleQualify(I_i, C_i)$ is true if and only if Input I_i satisfies the quality requirement regulated by Control C_i.

First, the resources in Element R are occupied when the task is being executed, and released when the task is finished. Yet, a task can also have multiple processing methods, using certain resources in one method and others in another method. $Exec(P_i)$ is true if P_i is executed, which is equivalent to $Occu(R_i)$, as when a process is executed, all resources linked to it are utilized. $Occu(R_i)$ can be further converted into the exclusive disjunction of multiple $Occu(R_{i,k})$, representing the selection of a processing method. The above logic is described by Rule (2) and (3).

$$Exec(P_i) = Occu(R_i) = \bigvee_k Occu(R_{i,k}) \tag{2}$$

$$R_i = \bigcup_k R_{i,k} \tag{3}$$

Second, the Element O of P_i is identical to the Element I of P_j if P_j is the succeeding task of P_i, namely,

$$(I_j \equiv O_i) \leftarrow Exist(y(P_i, P_j)). \tag{4}$$

Rule (4) can be used to indicate the materials or semi-finished products left on site or in warehouses. Such historical information can be obtained from bill of materials (BOM) or enterprise resource planning (ERP) system, and can be used to organize the manufacturing job, such as deciding the set of tasks to be executed.

Relevantly, Element $S/Customer$ contains the owner information of the input/output of a task, which can be used in task management. They can influence the earliest start time and latest finish time of the task. For example, Element S contains the available time of the Input, and Element $Customer$ contains the deadline of the $Output$, which is defined by Rule (5) and (6) with predicate $Avail(I)$ and $Valid(O)$, indicating the availability of input and the validity of the output respectively. Therefore, the time constraints defined in Section 3.1.3 can be expressed by

$$Avail(I_i) \leftarrow (t_{current} \geq t_{S_i}), \tag{5}$$

$$Valid(O_i) \leftarrow (t_{current} \leq t_{C_i}), \tag{6}$$

where $t_{current}$ is the current time, t_{S_i} is the available time of the Input S_i, and t_{C_i} is the deadline of the Output C_i. These relations can be later converted to the time constraints in the mathematical model.

Moreover, the control element can influence the executability of a task or the behavior of other elements. For example, it is particularly useful in quality control. When a part as an input does not reach some quality requirement, this and the subsequent tasks needs to be blocked, either by directly removing the tasks or by Rule (7), which extends Rule (2) and (3) to constrain a task with the control element and a quality-checking predicate $Qualify(I_i)$, such that the task is executed only if a processing method is selected and its input satisfies the quality requirement regulated by the control element.

$$\begin{aligned} &Exec(P_i) \leftarrow (Occu(R_i) \wedge Qualify(I_i) \wedge Qualify \in C_i) \\ \Leftrightarrow &Exec(P_i) \leftarrow \bigvee_k (Occu(R_{i,k}) \wedge ruleQualify(I_i, C_i)) \end{aligned} \tag{7}$$

To elaborate task relations in practice, the precedence relation contains the following information.

- One task and its list of predecessors.
- Type of dependency. There are four types of dependency, which is finish-to-start (FS), finish-to-finish (FF), start-to-start (SS), and start-to-finish (SF).
- Time difference threshold between tasks.
- Time difference operator. There are two types of time difference operators, which is greater-or-equal (GE), and less-or-equal (LE).
- Combination type of the precedence relation. There are two types of combination types, which is AND and OR.

The information of precedence edge can be represented by a tuple:

$$(predecessor, successor, dependency, time, operator, combination)$$

For example, the tuple $([P1], P2, [FS], [3.0], [GE], AND)$ represents the following precedence constraint:

$$\text{P2 start time} - \text{P1 finish time} \geq 3.0 \tag{8}$$

The tuple $([P1, P2], P3, [FS, FF], [3.0, 1.0], [GE, LE], OR)$ means that task P3 can be executed if one of the following constraints is satisfied:

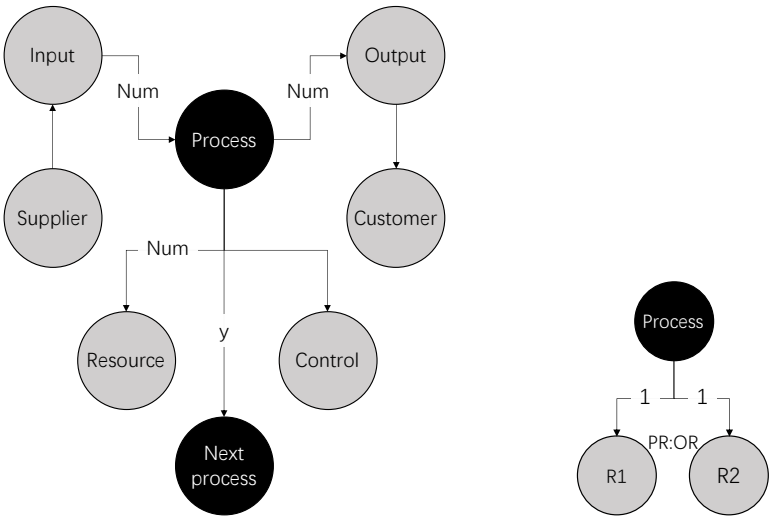
$$\text{P3 start time} - \text{P1 finish time} \geq 3.0 \tag{9}$$

$$\text{P3 finish time} - \text{P2 finish time} \leq 1.0 \tag{10}$$

One can see that the SIPyOC-RC model not only extends the SIPOC model by introducing the Resource element, the Control element, and the precedence relation, but also introduces new features into the elements. These features provide extensibility for the model, enabling it to describe more complex problems. To align SIPyOC-RC with existing IPPS models, the graphical representation of the SIPyOC-RC model is shown in Fig. 6(a), in which the precedence relation between two processes is also called a "y edge" between them. Fig. 6(b) illustrates processing method selection with resource selection labeled by a "PR:OR" edge. In this example, the task has two processing methods, using R1 or R2 resource, respectively.

In the graphical representation, the character "y" on the y edge between two tasks is omitted for simplicity when there is only one predecessor of a task, or the combination type is AND. All the precedence relations between tasks are assumed to be AND type by default, unless otherwise specified. Also note that in the SIPyOC-RC model, only the predecessors of tasks are defined. The successors of tasks are not explicitly given, but can be inferred from precedence relations. Therefore, in the SIPyOC-RC model, there will only be y edges, rather than pairs of "AND/OR" and "JOIN".

In some special cases, 'pseudo-task' can be used to simplify problem description. For example, assume that task P1 can be executed when P2a or P2b is finished, and P3 is executed. Such nested combination of precedence relationship can be described as follows: P1 can be executed when FP2 (a pseudo-task with zero processing time) and P3



(a) (b)

Figure 6: (a) An illustration for the SIPyOC-RC model, and (b) An example of a task with multiple processing methods, using R1 or R2 resource, respectively. The numbers of the edges represent the number of Input/Resources required by the Process, or the number of the Output produced by the Process. Multiple Input/Resource/Output nodes can be used in application if necessary.

is finished; FP2 can be executed when P2a or P2b is finished. In this case, the nested combination of precedence relation is avoided by using pseudo-task.

The Unified Modeling Language (UML) class diagram of the SIPyOC-RC model is shown in Fig. 7. The "duration" attribute of the Process class indicates the time needed to execute this process. The "resources" attribute contains all the resources needed for all the processing methods of this process. The "edge_y_list" attribute contains all predecessors of the process. Since the combination type of a precedence relation is shared by all the predecessors of the process, it is reasonable to store this variable ("combination") in Process. The Input, Resource, and Output class have "quantity" attribute, indicating the amount of available corresponding items in the problem. The "predecessor", "dependency", "time", and "operator" of Edge_y class refer to the predecessor, type of dependency, time difference threshold, and time difference operator of the precedence relation between the predecessor and the successor, respectively. The Control class has two functions: "validate()" and "execute()". The former is used to check whether the process can be executed, and the latter is used to influence the behavior of other nodes. The Supplier class has "min_start_time" attribute to indicate the earliest start time of the process. The Customer class has "max_end_time" attribute to indicate the latest finish time of the process.

4.2 Job Modelling

With the SIPOC-RC modelling method, complex product manufacturing jobs are then sufficiently modeled with demanded features, including production and assembly jobs. Without the loss of generality, job modelling is demonstrated by the two typical real

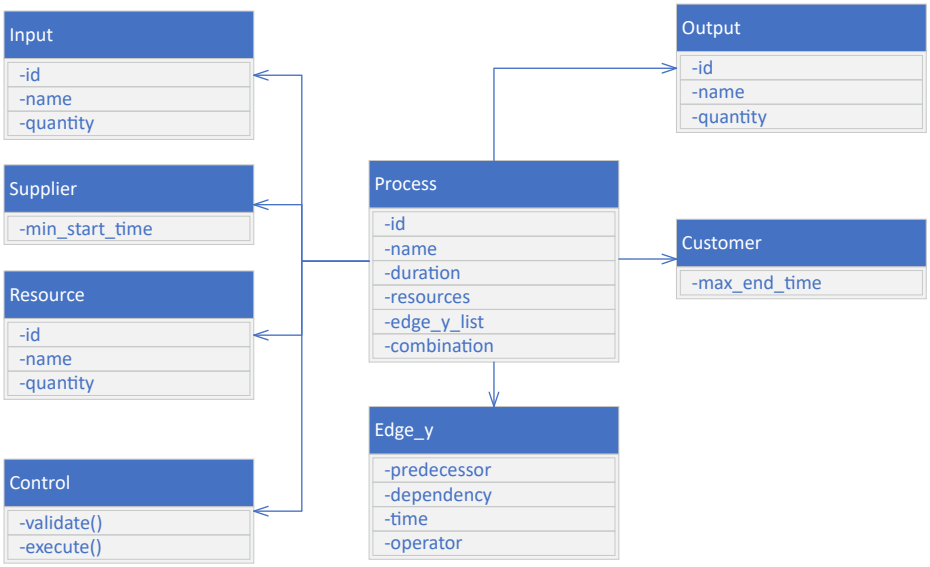


Figure 7: The UML class diagram of the SIPyOC-RC model in the software.

examples in Section 3, including a production job and an assembly job from the real manufacturing case.

All the characteristics of Example 1 are represented by the SIPyOC-RC model, as shown in Fig. 9. The OR type precedence relation is shown in Fig. 9 a). There are two ways to finish the job. One is to execute Task #cc655932 occupying two resources for 6 time units, while the other is to execute Task #47cff2df, #9f00300c, and #0f2fad34 occupying one resource for only 0.2 time units. Fig. 9 b) shows the time requirement between tasks. One task must start execution within 4.0 time units after its predecessor is finished. Fig. 9 c) describes the input/output of certain tasks, facilitating the analysis of inventory and semi-finished parts regulated by Rule (4). Fig. 9 d) shows the constraints of available time from supplier regulated by Rule (5), indicating that the task can only be started after 65.0. This constraint is described by the Supplier node, and influences the start time of its Process node. Fig. 9 e) describes the special controlling condition mentioned in Section 3.1.1 regulated by Rule (7).

The flexible selection of resources and delivery time in Example 2 are modeled by the SIPyOC-RC model, as shown in Fig. 9. As mentioned in Section 3.1.2, Task #16910961 can use 9 resources, but only three of them are required for execution, which is indicated by the Control node “C: Flex Res=3” and also regulated by Rule (2) – (3). Another Task #f86b0a0d can use 5 resources and requires 5 resources for execution, which is equivalent to the situation that the task requires all the resources. This task also has a delivery time constraint, indicated by the Customer node “Customer: End: 96.0” regulated by Rule (6). The zone (resource) required by the tasks are not explicitly shown in the figure for the clarity, but can be added in the same way as the human resource.

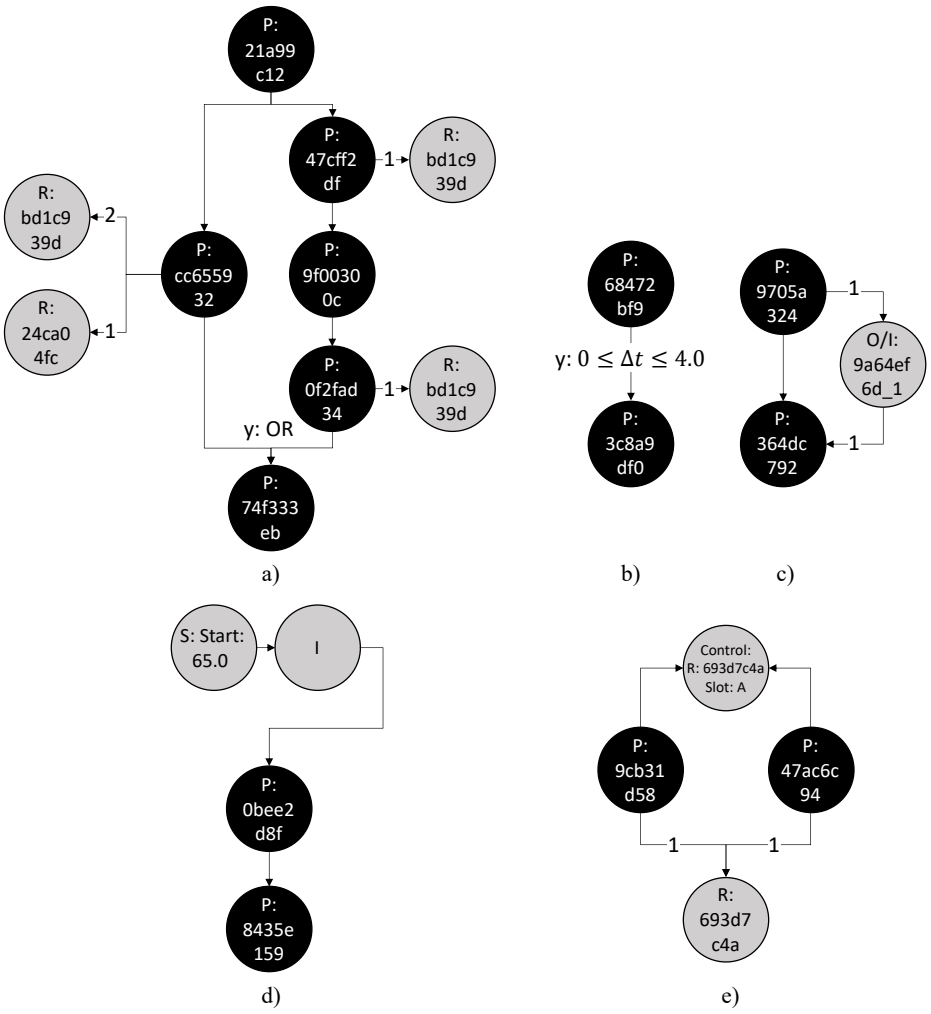


Figure 8: Part of SIPyOC-RC model of Example 1: production line. The black solid circles represent process node of tasks. The arrows between processes indicate the precedence relation between tasks. The gray circles represent the Resource, Control, Output/Input, and Supplier nodes of tasks. The number on the arrow indicates the number of corresponding resources required or given by the task. "O/I" node means that this node is an output of a task, but is also an input of another. "Control: R: 693d7c4a, Slot: A" means that this control condition is used to control the task to be executed in Slot "A" using Resource #693d7c4a. "Supplier: Start: 65.0" means that the task can only be started after 65.0.

4.3 Model Transformation

Similar to other modelling methods, the SIPyOC-RC model itself cannot be directly utilized for scheduling problem-solving, unless it is transformed into a mathematic

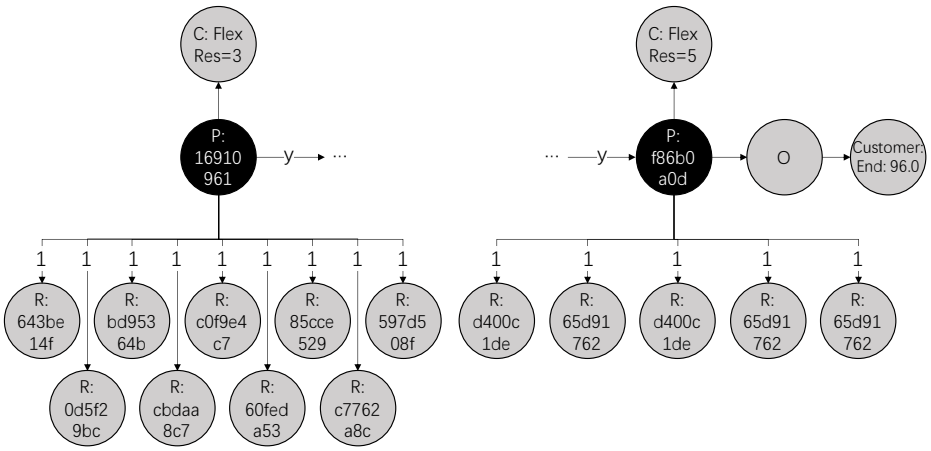


Figure 9: Part of SIPyOC-RC model of Example 2: assembly line. The black solid circles represent the process of tasks. The arrows between processes indicate the precedence relation between tasks. The gray circles represent the Resource nodes, Control, and Customer nodes of tasks. The number on the arrow indicates the number of corresponding resources required by the task. "C: Flex Res=3" is a Control node and means that the task requires only 3 human resources, regardless of the number of resources that can be used for this task. "Customer: End: 96.0" means that the task should be finished before 96.0.

model, such as the CP model or the Mixed Integer Linear Programming (MILP) model. Generally, both models can be solved with modern optimization software. In the practice of scheduling problems, CP models show the advantage of simplicity in problem formulation and solution performance [Meng et al., 2020]. Therefore, the SIPyOC-RC model is transformed into CP model before solving the IPPS problem. Given an IPPS job described by the SIPyOC-RC model which is regulated by Rule (2) – (7), a CP model is then derived from it to produce an optimal job execution schedule. Particularly, an approach similar to the "reduced combination" [Jin et al., 2016] is applied to model the OR precedence relation. To facilitate the description of the model, moreover, some notations are given in Table 3.

First, a common objective function is taken as an example: minimizing the makespan. The objective function is as follows:

$$\min \max_i s_i + \sum_k d_{ik} x_{ik}. \tag{11}$$

Next, task combination is formulated to represent flexible task selection. For the clarity of description, some definitions are given first. An **OR end node** represents a task with multiple predecessors, one of which needs to be executed. An **OR start node** represents a possible task with multiple successors corresponding to an OR end node, where the predecessors of the OR end node are the direct and indirect successors of the OR start node. OR start node and OR end node usually appear in pair, between which the sequences of tasks form **branches**. For example, in Fig. 10, there are two OR end nodes,

Design variables	s_i	Starting time of the i-th task
	x_{ik}	$= 1$, if the i-th task select k-th processing method; $= 0$, otherwise
	y_{de}	$= 1$, if the d-th max branch domain select e-th combination; $= 0$, otherwise
Variables	I_i	$:= [s_i, s_i + \sum_k d_{ik}x_{ik})$, the operating interval of the i-th task
Params	o_i	The i-th task
	r_{ikj}	The required amount of j-th resource of the k-th processing method of the i-th task
	d_{ik}	The duration of the k-th processing method of the ith task
	p_{ij}	The index of the j-th preceding task of the i-th task
	q_{ij}^k	$= 1/2/3$, if the type of dependency between i-th task and its j-th predecessor is FS/SS/FF
	q_{ij}^t	The time difference threshold between the i-th task and its j-th predecessor
	q_{ij}^i	$= 1$, If the time interval between the ith task and its jth preceding task needs to be greater than or equal to the threshold; $= 2$, if less than or equal to
	P_i	$:= \{o_k : k \in \{p_{ij}\}_j\}$, the set of predecessors of the i-th task
	$v_j(t)$	At time t, the total amount of j-th resource
	s_i^l	The earliest start time of the i-th task
	e_i^l	The latest finish time of the i-th task
	D_d	The set of tasks in the d-th max branch domain
	$\mathcal{D}$	The set of all tasks in the max branch domain
	B_{de}	The e-th task combination in the d-th max branch domain
	M	A very large positive integer
	$\mathcal{T}$	Set of all considered time
	$\mathbb{I}_I(t)$	$= 1$, if $t \in I$; $= 0$, otherwise

Table 3: Notations

which are 8 and 7. For y1, there are two branches, which is (3) and (4,5,6,7). For y2, the two branches are (5) and (6). The tasks between OR start node and its corresponding OR end node forms a **branch domain**. One branch domain can be nested within another one. The branch domain that is not nested within any other ones is called a **max branch domain**. For example, in Fig. 10 there are two branch domains, which are (3,4,5,6,7) and (5,6). (3,4,5,6,7) is a max branch domain, while (5,6) is not. As max branch domains are independent, branches are constructed through task combination within max branch domains. Such a branch is called a **combination**. For example, in the max branch domain of y1, there are three combinations, which are (3), (4,5,7), and (4,6,7).

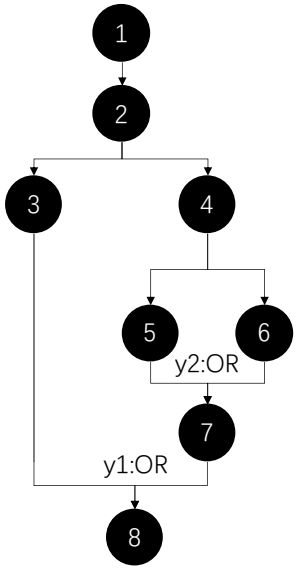


Figure 10: A sample of processes with OR-type precedence relationship.

The tasks that are not in any max branch domains should be executed in any case:

$$\sum_k x_{ik} = 1, \quad \forall i \notin \mathcal{D}. \tag{12}$$

The tasks in the max branch domains should be executed if one of the combinations it belongs to is selected:

$$\sum_k x_{ik} = \max_{(d,e)\in\{(d,e):i\in B_{de}\}} y_{de}, \quad \forall i \in \mathcal{D}. \tag{13}$$

Every max branch domain should have at least one combination selected:

$$\sum_e y_{de} = 1, \quad \forall d. \tag{14}$$

The constraints provided by resource (R) node should be satisfied. In other words, the sum of required resources of all executing tasks should not exceed the total amount of resources, at any considered time.

$$\sum_i \sum_k \mathbb{I}_{I_i}(t) \cdot r_{ikj} \cdot x_{ik} \leq v_j(t), \quad \forall t \in \mathcal{T}, j. \tag{15}$$

For the precedence constraints (y edge), one of the following equations can be used.

$$\begin{cases} s_{i'} + (\sum_{k'} d_{i'k'} x_{i'k'}) + q^t_{ii'} \le s_i + M\,(1 - \sum_{k'} x_{i'k'})\,, \\ \quad \text{if } q^k_{ii'} = 0 \text{ and } q^i_{ii'} = 0 \\ s_{i'} + (\sum_{k'} d_{i'k'} x_{i'k'}) + q^t_{ii'} \ge s_i - M\,(1 - \sum_{k'} x_{i'k'})\,, \\ \quad \text{if } q^k_{ii'} = 0 \text{ and } q^i_{ii'} = 1 \\ s_{i'} + q^t_{ii'} \le s_i + M\,(1 - \sum_{k'} x_{i'k'})\,, \\ \quad \text{if } q^k_{ii'} = 1 \text{ and } q^i_{ii'} = 0 \\ s_{i'} + q^t_{ii'} \ge s_i - M\,(1 - \sum_{k'} x_{i'k'})\,, \\ \quad \text{if } q^k_{ii'} = 1 \text{ and } q^i_{ii'} = 1 \\ s_{i'} + (\sum_{k'} d_{i'k'} x_{i'k'}) + q^t_{ii'} \le s_i + (\sum_k d_{ik} x_{ik}) \\ \quad + M\,(1 - \sum_{k'} x_{i'k'})\,, \\ \quad \text{if } q^k_{ii'} = 2 \text{ and } q^i_{ii'} = 0 \\ s_{i'} + (\sum_{k'} d_{i'k'} x_{i'k'}) + q^t_{ii'} \ge s_i + (\sum_k d_{ik} x_{ik}) \\ \quad - M\,(1 - \sum_{k'} x_{i'k'})\,, \\ \quad \text{if } q^k_{ii'} = 2 \text{ and } q^i_{ii'} = 1 \end{cases} \qquad \forall i, i' \in \{i' : o_{i'} \in P_i\} \tag{16}$$

The S and C (Customer) node provide the constraints of earliest start time and latest finish time of the task. The constraints are as follows:

$$s_i \ge s^l_i - (1 - \sum_k x_{ik})M, \quad \forall i. \tag{17}$$

$$s_i + \sum_k d_{ik} x_{ik} \le e^l_i + (1 - \sum_k x_{ik})M, \quad \forall i. \tag{18}$$

The constraints provided by Input and Output node should be considered in preprocessing. The tasks which do not have enough expendable resources or whose output product has already been produced should not be given to the solver. The constraints provided by Control node should be interpreted as decorations of other nodes. For example, a task that has a Control node allowing split of the task should divide the task into two tasks.

The procedure of model transformation can be more clearly illustrated by the pseudocode in Algorithm 1. Note that the Control node is used to influence the behavior of other nodes, which means that from the solver point of view, the Control node has changed the behavior of other nodes before all the inputs are given to the solver and the effects of the Control node has already been considered. Therefore, the Control node is not in the input of the pseudocode. Also, the constraints introduced by the Input and the Output nodes have already been addressed in the preprocessing procedure, and therefore, they are not included in the input of the pseudocode. Since the CP-SAT solver of Google OR-Tools is selected as the solver in this program, the model is constructed in the way of the CP-SAT solver. Therefore, variables used in the model are defined first (Line 2 – 18), followed by constraints (Line 19 – 43) and the objective function (Line 44 – 45). The variable definition has considered the time constraints regulated by Rule (5) – (6). In the part of constraints, specifically, Line 19 – 23 represent precedence constraints regulated by Rule (4), Line 24 – 34 represent resource constraints regulated by Rule (2) – (3), and Line 35 – 43 represent model constraints to guarantee the validity of the

model. Algorithm 1 employs some utility functions to simplify the pseudocode, which are described in Table 9 (Appendix B).

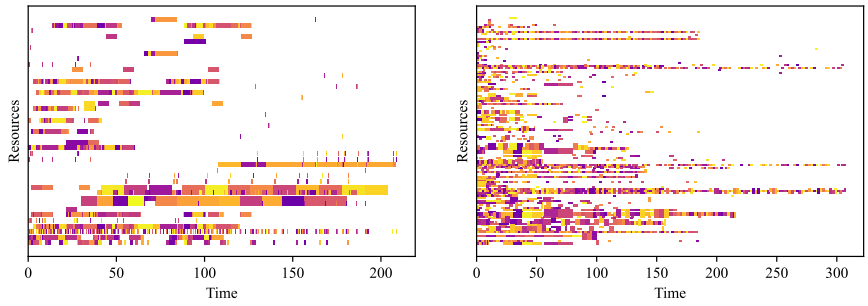
4.4 Problem Solving

After transforming job models to CP models, the IPPS problem in the two typical examples are solved with the CP-SAT solver of Google OR-Tools [Perron and Didier, 2024], a constraint optimization solver.

When modelling the problem in the CP-SAT solver, tasks are modeled using the interval variables ("NewIntervalVar"). Particularly, for Example 1, the time requirement between tasks is modeled by building an inequality containing the start time of the task and the end time of its predecessor. The constraint of semi-finished parts is addressed by a preliminary analysis in which the unnecessary tasks are identified. The tasks with output already in the inventory are identified as *unnecessary*. These unnecessary tasks are removed from the list of tasks to be scheduled. Also, since this paper focuses on the modelling of the problem rather than the solving of particular constraints, the special controlling condition in this example is simplified to the case that parts can be taken out of or inserted into the machine when the machine is working.

Resource usage is modeled using the "AddCumulative" function in the CP-SAT solver. The flexible selection of resources can be modeled as multiple processing methods, as shown in Fig. 4. Therefore, if one task requires two of Worker A, B, and C, and also requires zone α, then the resources required by the task are "1 A, 1 B, 2 α", "1 A, 1 C, 2 α", or "1 B, 1 C, 2 α". The delivery time constraint is modeled using the end time of the interval variable of the corresponding task.

The optimal makespan of the production job and the assembly job are 209.0 and 293.0 respectively. Fig. 11(a) and (b) shows the scheduling result of the two examples in Gantt chart. A full solution, as well as the SIPyOC-RC model (which is exported from neo4j graph database and can be imported into it), can be found in [Anonymous, 2024].



(a) *(b)*

Figure 11: Gantt chart of solution of (a) Example 1 and (b) Example 2. The Y axis represents the resources used by the tasks. Since the number of resources is too large, and displaying specific resource names is insignificant, the resource names are concealed for the sake of brevity.

Algorithm 1: Transform SIPyOC-RC model to a constraint programming model

Input: Sets of SIPyOC-RC Instances $\{S, P, Y, C, R\}$
Output: OR-Tools Model := $\{variables, constraints, objective\}$

Function `constructModel(`S, P, Y, C, R`):`
  $variables = list()$;
  $constraints = list()$;
  $objective = new\ objective()$;
  foreach $p \in P$ **do**
    if $y.combination ==$ "OR" **then**
      $variables.add(new\ boolVar(\text{"}branchSelect\text{"}, p.id))$;
    end
    $constraints.add(sum(Y.getPreprocessVars(p)) == 1)$;
  end
  foreach $p \in P$ **do**
    foreach $i \in getProcessingMethods(p)$ **do**
      $variables.add(new\ boolVar(\text{"}processingMethod\text{"}, p.id, i))$;
    end
    $constraints.add(sum(getProcessingMethodVars(p)) <= 1)$;
    $variables.add(new\ intervalVar(\text{"}interval\text{"}, p.id))$;
    $constraints.add(getInterval(p).duration ==$
     $getSelectedProcessingMethodDuration(p))$;
     $constraints.add(getInterval(p).start >= S.MinStartTime(p))$;
     $constraints.add(getInterval(p).end <= C.MaxEndTime(p))$;
  end
  foreach $p \in P$ **do**
    foreach $y \in Y.getPreprocesses(p)$ **do**
      $constraints.addPreprocess(y.predecessor, y.dependency,$
      $y.time, y.operator)$;
    end
  end
  foreach $p \in P$ **do**
    foreach $(k, r) \in p.resources$ **do**
      foreach $(r_name, r_quantity) \in r$ **do**
        $r_required =$
         $r_quantity * getProcessingMethodVar(p, k)$;
        $constraints.addResourceRequirement(r_name, p,$
        $r_required)$;
      end
    end
  end
  foreach $r \in R$ **do**
    $resources.addResource(r.name, r.quantity)$;
  end

(Continued)
foreach $p \in P$ **do**
  foreach $pre_p \in Y.getPreprocesses(p)$ **do**
    $constraint = (sum(getProcessingMethodVars(pre_p)) \geq sum(getProcessingMethodVars(p)))$;
    if $y.type ==$ "OR" **then**
      $constraint.OnlyEnforceIf(getPreprocessVar(p, pre_p))$;
    end
    $constrains.add(constraint)$;
  end
end
$total_time = max(getIntervalEnds)$;
$objective.Minimize(total_time)$;
return $variables, constraints, objective$

5 Evaluation and Discussion

In this section, the SIPyOC-RC-based computing framework is implemented as a software system as a testbed. On the basis of the software, several experiments are conducted with real manufacturing case data to evaluate the modelling adequacy and computational performance of the proposed modelling method. It is also compared with existing approaches, including table-based (e.g., feature table) and network graph-based (e.g., network graph and AND-OR digraph) modelling methods. Finally, the generality of the approach is discussed by applying the method to modelling existing IPPS problems.

5.1 System Implementation

To show the practicality of the proposed approach, the SIPyOC-RC model has been implemented in our software system, including three key components:

- Preprocessor: parses the input data (i.e., SIPyOC-RC description) and transforms it into a standard description in this program, and then transforms it into a CP model.
- Solver: solves the CP model.
- Postprocessor: organizes the results obtained by the solver, outputs formatted results, and visualizes them with Gantt chart.

The core of the software is the CP-SAT solver of Google OR-Tools. The problems are first parsed and transformed by the preprocessor, and then solved by the solver. The results are then post-processed and visualized by the postprocessor. The software provides a testbed for the experiments in the following subsections. The software can effectively solve the problem if the problem to be solved has limited number of constraints or the constraints are not very complex. If there are too many constraints, then it will become inefficient, which is out of the scope of this paper. The software has also been used in

our real production line for production scheduling, and in most cases there are not many complex constraints such as flexible resource selection.

Moreover, the user interface of the software is shown through Fig. 12-15. A user can import and edit the resources and input material in the "Data Modelling" tab. In "Process Modelling", information about the tasks and other nodes can be created, edited, and revised, as shown in Fig. 12 and 13. Fig. 14 shows an abstract diagram of the SIPyOC-RC model containing necessary types of nodes after some basic configuration has been set. Some confidential information has been replaced with text "name". Note that there is an edge whose endpoints are both Process nodes (P). Since this window shows the types of nodes (rather than actual node instances), this edge represents the precedence relation between two different tasks. In Fig. 15, the user can view the result of the problem.

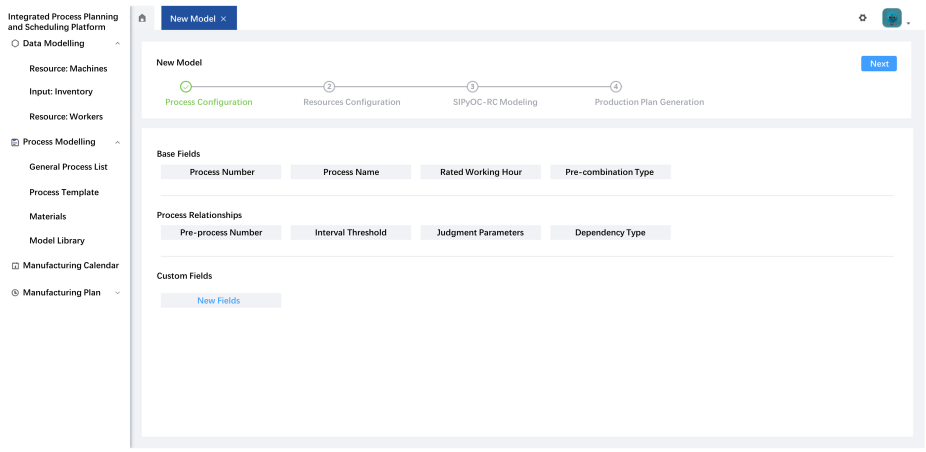


Figure 12: The screenshot of the software. This page helps the user to define attributes of the Process node.

5.2 Adequacy of Job Modelling

The adequacy of complex manufacturing job modelling is evaluated by the description loss of a model on specific IPPS problems, including the two examples of the real manufacturing case, and the corresponding influence on solution optimality. The examples are modeled by different modelling methods for comparison and then solved by our software.

Given an IPPS problem, a modelling method may have to modify some of the original constraints to partly express a new constraint, or omit an inexpressible new constraint of the problem, which will cause description loss, since some complex constraints of the problem cannot be fully described by the modelling method. Description loss may lead to solution deviation, such as a deviated makespan, compared to the optimal solution of the problem when it is fully described. Note that the value of such deviation can

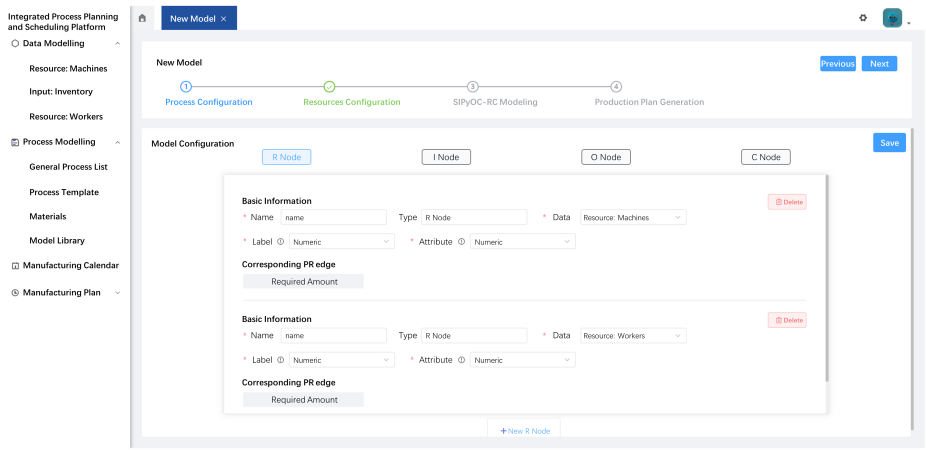


Figure 13: The screenshot of the software. This page allows the user to edit the R/I/O/Control nodes.

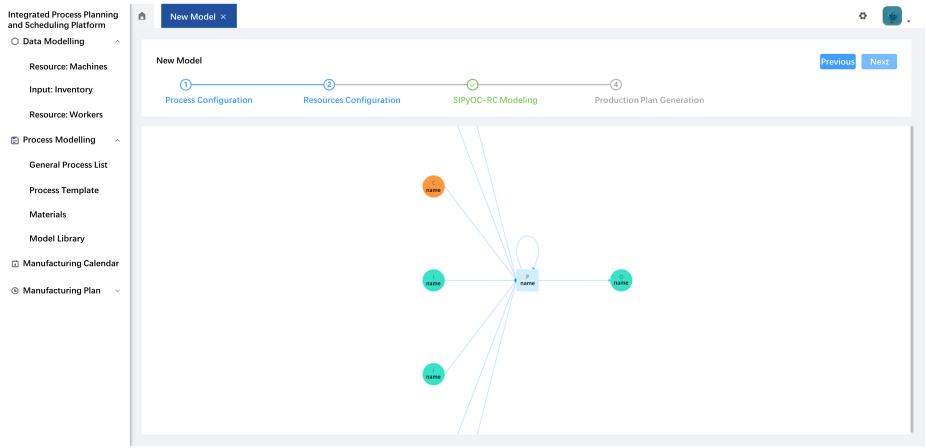


Figure 14: The screenshot of the software. This page helps the user to revise the types of nodes in SIPyOC-RC model.

be positive or negative. A positive / negative deviation means that the result of the problem becomes larger / smaller after the constraints are modified or omitted. When the constraint is omitted or modified to be a weaker one, more solutions become feasible, and the deviation tends to be negative. When the constraint is modified to be a stronger one (e.g., changing an OR type precedence relation to an AND type), the flexibility of

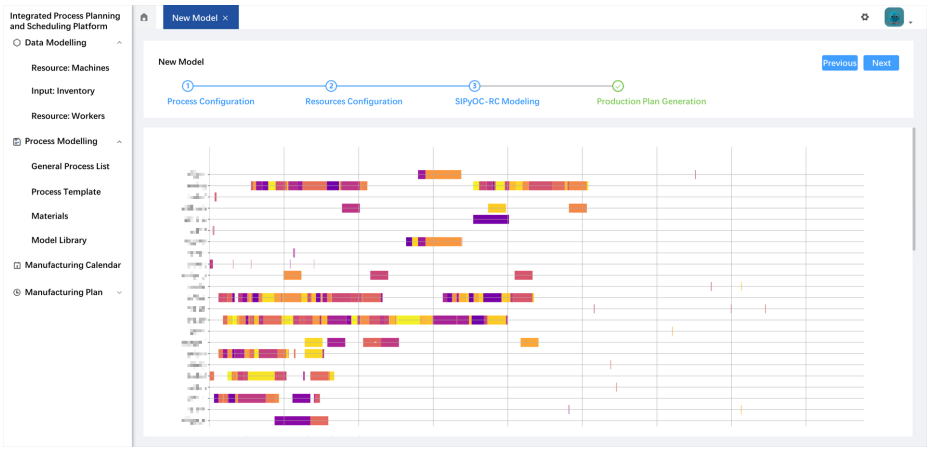


Figure 15: The screenshot of the software. This page can show the result of the problem.

the problem is reduced, and the deviation tends to be positive.

To satisfy the new modelling requirements in practice, the following modifications of constraints have to be made for feature table and network graph. For Example 1, the OR type precedence relation is omitted for the feature table, and only one branch of the OR type precedence relation is kept. Since the network graph supports the OR type precedence relation, no modification is made for it. The time requirement between tasks, semi-finished parts, available time, and special controlling conditions are omitted for both modelling methods. For Example 2, the flexible resource constraint is modified to require all the capable workers for the feature table and the network graph. If the number of capable workers is larger than required, then only the workers of required quantity will be used. The delivery time constraint is omitted for both modelling methods.

After solving CP problems, description loss and solution deviation of different modelling methods on the two problems are analyzed. The description on a feature (constraint) is lost if a process with that feature (constraint) cannot be fully described by a modelling method in that feature (constraint). Thus to analyze description loss, every main feature in Section 3 is separately inspected. Notably, although feature table and network graph does not support the simultaneous use of multiple resources in one task, this paper still assumes their support in order to illustrate the expressiveness of different modelling methods for the main demanded features.

The comparison result in description loss is shown in Table 4. It can be learned that although the network graph can represent more constraints than the feature table, it still lacks the capability to describe many constraints in both problems. For Example 1 the network graph has a better result than the feature table, while for Example 2, two methods have the same result. The SIPyOC-RC model can represent all the constraints in both problems, and has the same result as the fully described problems. This result shows that the SIPyOC-RC model is capable of describing highly complex scheduling problems with no loss.

The description loss in feature table and network graph will lead to solution deviation, compared with a SIPyOC-RC-based model. For Example 1, the influence of description loss on the solution is shown in Table 5. The results of the problem described by different modelling methods are shown in the first several rows in this table. It can be seen that

		Feature Table	Network Graphs (AND/OR Graphs)	SIPyOC-RC
Desc. Loss	E1	OR Precedence relation (29) Waiting Time (13) Semi-finished product (10) Available Time (4) Special Control Conditions (26)	Waiting Time (13) Semi-finished product (10) Available Time (4) Special Control Conditions (26)	None
	E2	Flexible Resource (41) Delivery Time (5)	Flexible Resource (41) Delivery Time (5)	None

Table 4: Comparison of different modelling methods on the description loss for the two example problems. The number in the brackets is the number of constraints not fully described when trying to be described by the corresponding modelling method.

the optimal makespan of the problem described by feature graph and network graph are larger than that of the problem fully described (i.e., in the SIPyOC-RC model). To show the influence of the description loss of certain constraints, different optimal makespan of the problem are shown in the table by controlling description loss. It can be found that, first, the loss of OR type precedence relation leads to a large increase of optimal makespan, as the task can only be executed in one way. Second, the semi-finished product constraint has a significant influence on the result, which directly influences the number of tasks that need to be executed. Moreover, the constraint of waiting time and available time have no impact on the problem solution. By examining the data of the problem, it can be found that the number of precedence relations in this problem is small, which means that the order of tasks is very flexible. So, these two constraints do not have a significant influence on the optimal makespan.

Particularly from Table 5, it seems that the loss of the controlling requirement constraint does not change the optimal makespan. To investigate the reason, Fig. 16 shows the usage of Machine #693d7c4a related with the "special controlling condition". When the constraint is lost, the machine cannot be shared among tasks, which means that the total makespan is likely to be extended. From the figure, nevertheless, it can be learned that the machine is not fully occupied in a certain period even when the constraint is lost, which means that this machine is not a bottleneck resource in scheduling. Therefore, in this problem, the loss of the controlling requirement constraint does not have a significant influence on the optimal makespan. Nevertheless, one can expect that if more tasks share this machine, the influence of the description loss of this constraint will be more significant.

A similar experiment is also conducted for Example 2, and the result is shown in Table 6. As feature table and network graph lose both the flexible selection of resources constraints and the delivery time constraints, the optimal makespan of them is 4.8% larger than the optimal makespan when the constraints fully described. Theoretically, the description loss of flexible selection of resources will increase the optimal makespan, as the task can only be executed by fixed workers, leading to flexibility loss. The description loss of delivery time will decrease the optimal makespan, as tasks can be executed at any time, leading to feasible options. Consequently, the description loss of the two constraints influence the problem solution from different directions, and their influences are not simply additive. It is interesting to observe from the result that delivery time seemingly does not have an influence on the optimal makespan. Due to the same reason as the

	OR Precedence Relation	Waiting Time	Semi-finished Product	Available Time	Special Control Conditions	Problem Solving Result
Feature Table	✗	✗	✗	✗	✗	301.2
Network Graphs		✗	✗	✗	✗	263.6
SIPyOC-RC						209.0
Loss of the below features						
OR Precedence Relation	✗					250.7
Waiting Time		✗				209.0
Semi-Finished Product			✗			299.6
Available Time				✗		209.0
Special Control Conditions					✗	209.0

Table 5: Optimal makespan of Example 2 in different situations of description loss. "SIPyOC-RC", "Feature Table", and "Network Graph" refer to the results of the problem described by corresponding modelling methods. "... Loss" refers to the result of the problem when certain constraint is not fully described while the others are not. Across "✗" in the table indicates that the corresponding constraint is lost.

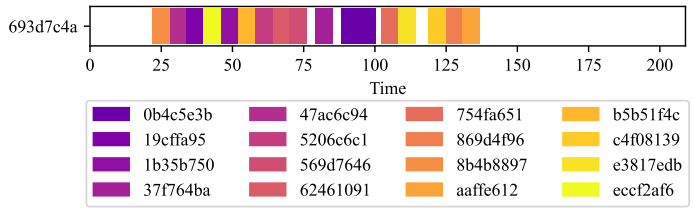


Figure 16: Usage of the resource Machine #693d7c4a related with the "special controlling condition". Each box represents the usage of the machine by one task. The ID of the tasks are listed in the legend. The X axis covers the whole makespan (0–209.0).

waiting time constraint and the available time constraint in Example 1, the delivery time constraints in Example 2 also do not have a significant influence on the results.

5.3 Computation Performance

The average total computation time and the pure solving time are also recorded. Each problem is solved three times using the same settings, and the average solving time is taken as the solving time of the problem. To simulate the entire process from receiving the data to presenting the results, the time required for parsing the raw data (json), transforming it into a standard description in the computer program and a solvable model, and organizing the final results is also recorded. These times along with the pure solving

	Flexible Resource	Flexible Resource Lost
Delivery Time	293.0 (SIPyOC-RC)	307.0
Delivery Time Lost	293.0	307.0 (FT, NG)

Table 6: Optimal makespans of Example 2 in different situations of description loss. "SIPyOC-RC" refers to the result of the problem described by the SIPyOC-RC model. "FT" and "NG" refer to the results of the problem described by the feature table and the network graph.

time are called "total computation time". All experiments are run on a computer with Intel Core i9-12900H CPU @ 2.50 GHz.

The experiment result is shown in Table 7 for two examples. For Example 1, first note that the semi-finished product constraint can only be described by the Input node of the SIPyOC-RC model. The absence of the semi-finished product constraint makes the solving time of the problem described by the feature table and the network graph longer than that by SIPyOC-RC, due to more tasks to be arranged. The computation time of the problem described by the feature table is smaller than that of the network graph, because the OR precedence constraint, which can be described by the network graph, can make the problem more complex and time-consuming. The average time of parsing the SIPyOC-RC model and transforming it into the standard description in the computer program is 0.20 seconds. Together with Example 1, one can see that the time of parsing the SIPyOC-RC model and transforming it into the standard description is very small and can be ignored. Notably, all problems have to be transformed into the standard description regardless of modelling method. One can conclude that the SIPyOC-RC modelling method can achieve modelling adequacy without significant computation cost.

For Example 2, one may think that such implementation of flexible selection of resources (introduced in Section 3.1.2) leads to a significant increase in the complexity of the problem. Indeed, one can see from Table 7 that when the flexible selection of resources is ignored, the computation time is greatly reduced. Yet, the optimization of the solving approach is out of the scope of this paper, and the solving time is also acceptable for this problem. The average time of parsing the SIPyOC-RC model and transforming it into the standard description in the computer program is 0.06 seconds.

		Total Computational Time (s)	Pure Solving Time (s)
E1	Feature Table	178.80	9.01
	Network Graphs	237.16	24.82
	SIPyOC-RC	171.63 (Parse SIPyOC-RC using 0.20)	3.55
E2	Feature Table	86.67	0.35
	Network Graphs		
	SIPyOC-RC	215.97 (Parse SIPyOC-RC using 0.06)	57.80

Table 7: Computation time of different modelling methods on two examples.

5.4 Applicability with Low-flexibility Problems

A good modelling model should not only be able to represent new and higher complex problems, but also represent the conventional ones with consistent paradigm. In this subsection, the applicability of the SIPyOC-RC model to problems of different complexity is illustrated by the implementation of multiple categories of problems in the literature.

FT10 problem [Muth and Thompson, 1963] is a job shop scheduling problem (JSP), containing 10 jobs, each with 10 tasks. Every task utilizes only one machine. The SIPyOC-RC model of the first job of this problem is shown in Fig. 17. The optimal solution of this problem is found using the converted CP model. Note that this problem has multiple optimal solutions, which is not influenced by the modelling method of the problem.

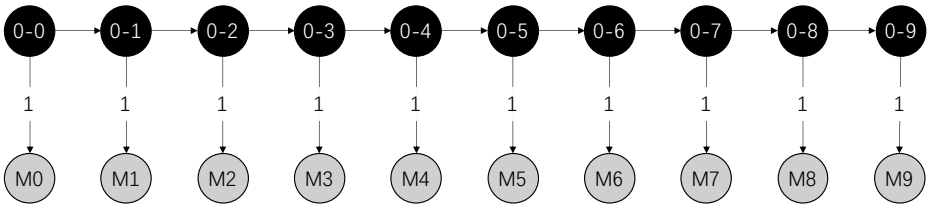


Figure 17: SIPyOC-RC model of the first job of FT10 problem [Muth and Thompson, 1963]. The black solid circles represent the tasks with name 'job_index - task_index'. The arrows between tasks indicate the precedence relation between tasks. The gray circles represent the resources required by the task linked by an arrow. The number on the arrow indicates the number of corresponding resources required by the task.

Moon's problem [Moon et al., 2008] is an FJSP problem according to the classification of this paper. This problem contains 5 jobs, each with several tasks using one of five resources. The SIPyOC-RC model of this problem is shown in Fig. 18.

Shao's problem [Shao et al., 2009] is an IPPS problem, containing 7 jobs and 8 machines. Tasks have different processing methods, and each processing method requires different resources and different processing time. The modelling result also ininclude OR-type precedence relations. The SIPyOC-RC model of Job 1 in this problem is shown in Fig. 19.

One can see that the SIPyOC-RC model differs in many aspects from existing modelling methods. A summary of the comparison between the SIPyOC-RC model and other modelling methods on different types of problems is shown in Table 8. Some table-based methods may have the ability to represent IPPS problems. However, due to their two-dimensional structure, table-based methods have limited representation capability, making the representation of complex problems cumbersome and often requiring back-and-forth referencing. For example, in Table 1, more effort is needed to figure out the precedence relation among F1, F2, and F3. On the other hand, a graph model can easily and clearly provide the precedence information of tasks. Therefore, the attempts to describe IPPS problems using table-based methods should not be entirely negated, though it is not recommended doing this.

The graph-based method can represent a large number of IPPS problems, but still has some limits in representing complex IPPS problems. From the aspect of resource requirement, in Fig. 1, only the machine resource of tasks can be represented, while other

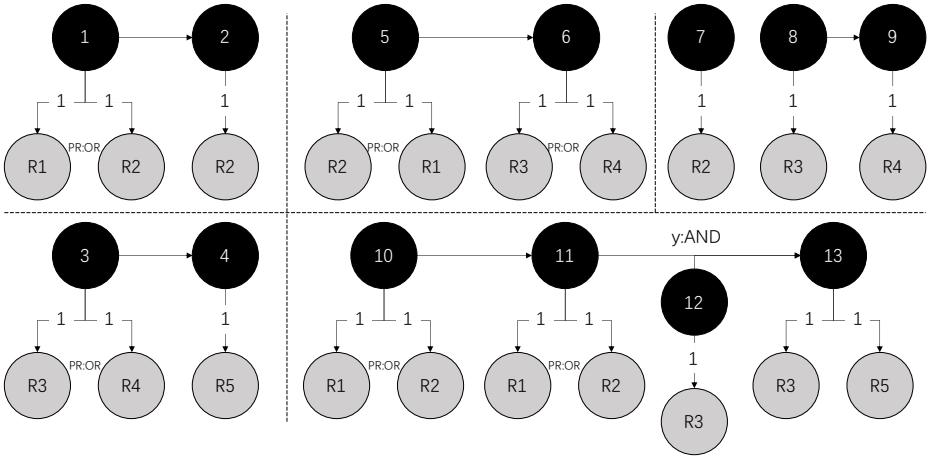


Figure 18: SIPyOC-RC model of the first job of Moon's problem [Moon et al., 2008]. Five jobs are separated using dashed lines. The 'OR' in this figure represents resources required by different processing methods. For example, Task 1 has two processing methods. One requires R1, and the other requires R2. The 'AND' in this figure represents that Task 13 can be executed only after Task 11 and Task 12 are executed.

resource types cannot. Also, only one of the resources can be used in each processing method of a task, instead of a combination of them. From the aspect of precedence relation, the network graphs can only represent typical JOIN-AND/OR "branch-based" precedence relation, as mentioned in Section 3.1.1. The "branch-based" modelling method limits the freedom of problem description, since there must be a "JOIN" node for one AND/OR node, which is actually not necessary as discussed before. Also, graph-based methods do not support type of dependency, time difference threshold between tasks, and time difference operator in a precedence relation. Moreover, graph-based methods are not capable of describing other constraints, such as the delivery/available time, influence of semi-finished parts, and special controlling conditions.

In contrast, the SIPyOC-RC-based modelling method can represent all the problems that can be represented by table-based and graph-based methods. Meanwhile, it can also represent many complex IPPS problems, including all the constraints summarized in Section 3, which is enabled by different types of nodes and the extensibility of the SIPyOC-RC model.

6 Conclusion and Future Work

This paper has proposed a new computing framework for modelling and solving IPPS problems of complex product manufacturing. Based on the SIPOC model, the proposed SIPyOC-RC modelling method can comprehensively represent all the modelling features supporting complex job modelling, flexible resource selection, and various time constraints, which are demanded in describing complex product manufacturing jobs, such as commercial airplane component production and assembly. To solve complex IPPS problems, a scheme to transform the SIPyOC-RC model to the CP model has also been

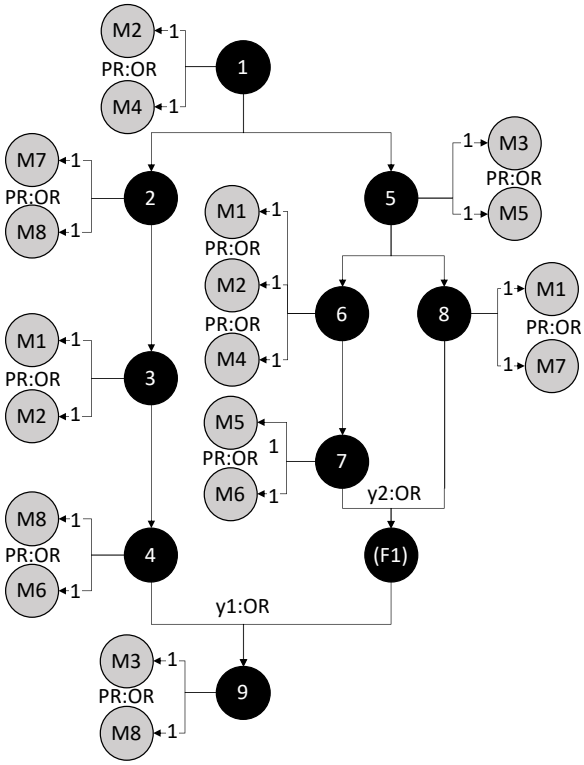


Figure 19: SIPyOC-RC model of the first job of Shao's problem [Shao et al., 2009].

Modelling Method	JSP	FJSP	IPPS	Complex IPPS
Table-based (e.g., Feature Table)	✓	✓	-	✗
Graph-based (e.g., AND-OR Digraph and Network Graph)	✓	✓	✓	✗
SIPyOC-RC-based	✓	✓	✓	✓

Table 8: Comparison of different modelling methods on different types of problems.

given, which is then solved with a CP-solver. Multiple real and conventional scheduling problems in manufacturing have demonstrated the effectiveness and efficiency of the proposed framework.

The proposed model solves the problem of complex IPPS problem description. However, the solving of complex IPPS problems is still a challenge. For example, when the number of tasks or the resources in the IPPS problem is large, the solving method may require a significant amount of time to complete the solving, assuming it can find a solution at all. Some special constraints are hard to be modeled, or require a huge amount

of computational resources. When unexpected situations arise, the IPPS problem may also change accordingly, requiring the solving algorithm to complete the solution within a short period of time. Therefore, the authors plan to develop an effective solving method for the IPPS problems described by the SIPyOC-RC model in the future.

Acknowledgements

This research is kindly supported by the Young Scientist Program of National Key R&D Program of China (No. 2022YFB3302800).

References

[Anonymous, 2024] Anonymous (2024). Data of real examples. https://github.com/FrankHanZ/SIPyOC-RCpaper. [Online; accessed 26-Feb-2024].

[Benjaafar and Ramakrishnan, 1996] Benjaafar, S. and Ramakrishnan, R. (1996). Modelling, measurement and evaluation of sequencing flexibility in manufacturing systems. *International journal of production research*, 34(5):1195–1220.

[Bucki et al., 2015] Bucki, R., Chramcov, B., and Suchánek, P. (2015). Heuristic algorithms for manufacturing and replacement strategies of the production system. *JUCS - Journal of Universal Computer Science*, 21(4).

[De Giovanni and Pezzella, 2010] De Giovanni, L. and Pezzella, F. (2010). An improved genetic algorithm for the distributed and flexible job-shop scheduling problem. *European journal of operational research*, 200(2):395–408.

[Demir et al., 2024] Demir, H. I., Kökçam, A. H., and Erden, C. (2024). *Integrated Process Planning, Scheduling, and Due-date Assignment*. CRC Press.

[Elgendy et al., 2023] Elgendy, A., Yan, J., and Zhang, M. (2023). A parallel distributed genetic algorithm using apache spark for flexible scheduling of multitasks in a cloud manufacturing environment. *International Journal of Computer Integrated Manufacturing*, pages 1–16.

[Jin et al., 2016] Jin, L., Tang, Q., Zhang, C., Shao, X., and Tian, G. (2016). More MILP models for integrated process planning and scheduling. *International Journal of Production Research*, 54(14):4387–4402.

[Jin and Zhang, 2019] Jin, L. and Zhang, C. (2019). Process planning optimization with energy consumption reduction from a novel perspective: Mathematical modeling and a dynamic programming-like heuristic algorithm. *IEEE Access*, 7:7381–7396.

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

[Kiritsis and Porchet, 1996] Kiritsis, D. and Porchet, M. (1996). A generic Petri net model for dynamic process planning and sequence optimization. *Advances in Engineering Software*, 25(1):61–71.

[Li et al., 2009] Li, W., Gao, L., and Li, X. (2009). Application of intelligent strategies for cooperative manufacturing planning. *JUCS - Journal of Universal Computer Science*, 15(9).

[Li et al., 2019] Li, X., Gao, L., Wang, W., Wang, C., and Wen, L. (2019). Particle swarm optimization hybridized with genetic algorithm for uncertain integrated process planning and scheduling with interval processing time. *Computers & Industrial Engineering*, 135:1036–1046.

[Li et al., 2012] Li, X., Gao, L., and Wen, X. (2012). Application of an efficient modified particle swarm optimization algorithm for process planning. *The International Journal of Advanced Manufacturing Technology*, 67.

[Li et al., 2010] Li, X., Gao, L., Zhang, C., and Shao, X. (2010). A review on integrated process planning and scheduling. *International Journal of Manufacturing Research*, 5(2):161–180.

[Li et al., 2014] Li, X., Wen, X., and Gao, L. (2014). An effective genetic algorithm for multi-objective integrated process planning and scheduling with various flexibilities in process planning. *JUCS - Journal of Universal Computer Science*, 20(14).

[Lin et al., 2020] Lin, C.-S., Li, P.-Y., Wei, J.-M., and Wu, M.-C. (2020). Integration of process planning and scheduling for distributed flexible job shops. *Computers & Operations Research*, 124:105053.

[Lin and Solberg, 1991] Lin, G. Y.-J. and Solberg, J. J. (1991). Effectiveness of flexible routing control. *International Journal of Flexible Manufacturing Systems*, 3(3):189–211.

[Liu et al., 2021] Liu, Q., Li, X., and Gao, L. (2021). A novel MILP model based on the topology of a network graph for process planning in an intelligent manufacturing system. *Engineering*, 7(6):807–817.

[Liu et al., 2016] Liu, X., Ni, Z., and Qiu, X. (2016). Application of ant colony optimization algorithm in integrated process planning and scheduling. *The International Journal of Advanced Manufacturing Technology*, 84:393–404.

[Marques and Requeijo, 2009] Marques, P. A. and Requeijo, J. G. (2009). Sipoc: A six sigma tool helping on iso 9000 quality management systems. In *XIII Congreso de Ingeniería de Organización*, pages 1229–1238.

[Meng et al., 2020] Meng, L., Zhang, C., Ren, Y., Zhang, B., and Lv, C. (2020). Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem. *Computers & Industrial Engineering*, 142:106347.

[Mishra and Kumar Sharma, 2014] Mishra, P. and Kumar Sharma, R. (2014). A hybrid framework based on sipoc and six sigma dmaic for improving process dimensions in supply chain network. *International Journal of Quality & Reliability Management*, 31(5):522–546.

[Moon et al., 2008] Moon, C., Lee, Y. H., Jeong, C. S., and Yun, Y. (2008). Integrated process planning and scheduling in a supply chain. *Computers & Industrial Engineering*, 54(4):1048–1061.

[Muth and Thompson, 1963] Muth, J. F. and Thompson, G. L. (1963). *Industrial scheduling*. Prentice-Hall.

[Perron and Didier, 2024] Perron, L. and Didier, F. (2024). Google CP-SAT solver. url: https://developers.google.com/optimization/cp/cp_solver/.

[Phanden et al., 2011] Phanden, R. K., Jain, A., and Verma, R. (2011). Integration of process planning and scheduling: a state-of-the-art review. *International Journal of Computer Integrated Manufacturing*, 24(6):517–534.

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

[Shokouhi, 2018] Shokouhi, E. (2018). Integrated multi-objective process planning and flexible job shop scheduling considering precedence constraints. *Production & Manufacturing Research*, 6(1):61–89.

[Sugimura et al., 2001] Sugimura, N., Hino, R., and Moriwaki, T. (2001). Integrated process planning and scheduling in holonic manufacturing systems. In *Proceedings of the 2001 IEEE International Symposium on Assembly and Task Planning (ISATP2001). Assembly and Disassembly in the Twenty-first Century. (Cat. No.01TH8560)*, pages 250–255.

[Wen et al., 2021] Wen, X., Wang, K., Li, H., Sun, H., Wang, H., and Jin, L. (2021). A two-stage solution method based on NSGA-II for green multi-objective integrated process planning and scheduling in a battery packaging machinery workshop. *Swarm and Evolutionary Computation*, 61:100820.

[Zhou et al., 2023] Zhou, Z., Xu, L., Ling, X., and Zhang, B. (2023). Digital-twin-based job shop multi-objective scheduling model and strategy. *International Journal of Computer Integrated Manufacturing*, pages 1–21.

[Łapczyńska et al., 2022] Łapczyńska, D., Łapczyński, K., Burduk, A., and Machado, J. (2022). Solving the problem of scheduling the production process based on heuristic algorithms. *JUCS - Journal of Universal Computer Science*, 28(3).

A Figure of the Overall Framework

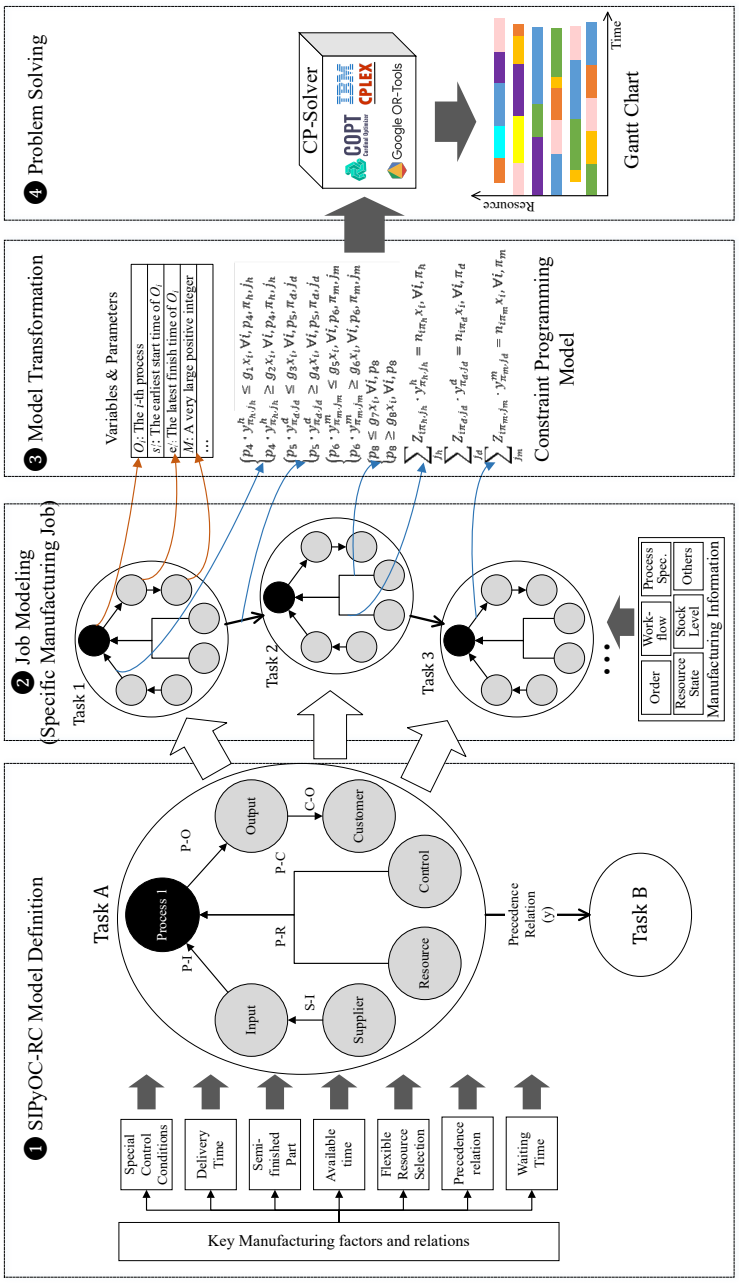


Figure 20: (Enlarged Fig. 5) A SIPyOC-RC-based computing framework IPPS problem modelling and solving.

B Utility Functions of Algorithm 1

Function Name	Description
boolVar, intervalVar	Create a variable of bool, integer, and interval type.
add	Add variables or constraints when used with “variables” or “constraints”.
addResourceRequirement	Add resource constraints with resource name, process, and the amount of resources as parameters.
addPreprocess	Add precedence constraints with predecessor, type of dependency, time different threshold, and time difference operator as parameters.
addResource	Add one resource with resource name and its quantity as parameters.
getSelectedProcessing-MethodDuration	Return the duration of the selected processing method of a task P.
getProcessingMethods	Return all processing methods of a task P.
getProcessingMethodVars	Return all processing method selection variables of a task P.
getProcessingMethodVar	Return the processing method selection variable of a task P and its processing method k.
getPreprocessVar	Return the predecessor selection variable of a task P and its predecessor.
getPreprocesses	Return all predecessors of a task P.
getPreprocessVars	Return all predecessor selection variables of a task P.
getMinStartTime	Return the minimum start time of a task P.
getMaxEndTime	Return the maximum end time of a task P.
getInterval	Return the end times of all the intervals.

Table 9: Function Description of Algorithm 1