Computers & Operations Research
Manuscript Draft

Manuscript Number: COR-D-19-00455R1

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; constraint programming; CP Optimizer; graph-based modelling; AND/OR graph.

Abstract: Integration of process planning and scheduling is to carry out the two functions simultaneously. This paper provides a graph-based constraint programming (CP) model based on the IBM ILOG CP Optimizer to solve the integrated process planning and scheduling (IPPS) problem in the jobshop environment. A tailored AND/OR graph structure is defined for the IPPS problem. The AND/OR graph defines not only the precedence relations but also the presence relations among nodes. Interval variables and scheduling-oriented constraints built in CP Optimizer are borrowed to project the AND/OR graph based IPPS problem to a concise CP model. An applied CP model based on type definition for nodes is provided for practical implementations. Incorporating minimization of makespan as the single objective, the graph-based CP model is tested on a set of benchmark problems. Experimental results show that the proposed approach outperforms compared algorithms in major IPPS instances. The performance of the proposed approach in solving IPPS instances of larger scales is even better.

***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 Kim et al. (2003) are used to test the proposed approach.

Response to Reviewers' Comments

We would like to sincerely thank Professor Francisco Saldanha da Gama, the Chief Editor, for offering us the opportunity to revise and resubmit our paper entitled *A graph-based constraint programming approach for the integrated process planning and scheduling problem* (COR-D-19-00455), as part of a major revision. We are also grateful to anonymous reviewers who have offered invaluable guidance for developing the paper further. The comments by the review team were instrumental for us to develop a significantly improved and enhanced version of our manuscript. In preparing the revised version of the paper, we did not consider the revision requirements as simply compliance with reviewer requests, but rather an opportunity to truly enhance this work and increase its potential to influence the field. We believe that revising the manuscript, based on the review team's comprehensive and constructive feedback, has elevated it to a new level of significance and contribution. We are therefore indebted to the guidance provided.

We agree with the well-founded criticisms. In the following, we describe in detail how we have addressed each comment by the reviewer. For ease of review, we address each comment below in the order that it was provided. We believe we have accommodated all review comments. Should there however be any additional comments, we will be happy to accommodate them.

Overall, we request the editor and anonymous reviewers to accept our sincerest gratitude. This study has benefitted significantly from the kind and constructive feedback provided.

Reviewer #1:

1) Abstract: It should be a summary of the whole paper, not only the establishment of the should be re-written (Such a detailed explanation of specific AND/OR model steps is not necessary, since this has already been presented in the scientific literature for other scheduling problems)

Answer.

Thanks very much for the reviewer's comments. We have rewritten the abstract and made it as concise as possible. The current abstract reports the main contributions and findings of this paper which we hope should satisfy the readers.

2) General: Many expression/careless/… errors in all sections (e.g. „It was well proven that traditional jobshop scheduling problems belong are NP-hard (Sotskov and Shakhlevich (1995)" or „…combines … into…" or „Global constraints, …, is .." or sentences starting with small letters or surprisingly appearing words with only capital letters); the article „the" is also missing in many cases in the whole paper. These mistakes all have to be fixed in order to guarantee a proper style of academic English language.

Answer.

Thanks very much for the sincere criticism. We feel really sorry for the carelessness in the language. We have revised the language carefully and tried our best to remove such errors. At the meantime, we have rewritten some parts to enhance the readability.

3) Introduction: It is described that exact approaches do not work well and therefore, approximate approaches have been developed. However, in the next paragraph, CP is described without referring to the former paragraph -> CP itself is an exact approach, which should be highlighted.

Answer.

We attempted to express that the general MIP approaches sometimes do not work well in the complex scheduling problems. Thanks to the reviewer's comment, we noticed that we have no sound evidences to prove that. The performance of algorithms varies in solving specific problems. Research communities in the scheduling area are continuously improving algorithms. As a result, the relative performance of algorithms keeps changing from time to time. In current version, we removed the misleading expressions about the judgement on exact and approximate approaches. Instead, we included two newly-published mathematical models and one exact algorithm for Type-1 IPPS problem.

4) Introduction: Citations are missing (1) "IPPS" is not cited; (2) "since the performance of CP approach heavily depends on the complexity of the model" is also not cited - moreover, it should be thought about this sentence again -> what exactly is meant with "complexity of the model" (please note that testing different problem instances of different scheduling problems with the CP Optimizer shows that depending on the objective of the problem, the problem size itself is not the major issue; other characteristics such as capacities etc. are much more challenging for the CP Optimizer. Therefore, this content has to be more specified)

Answer.

We have cited two articles

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

b) 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.

The citation a) is for the illustration of drawbacks of sequential process planning and scheduling, and the citation b) is for the concept of IPPS problem.

We have removed the sentence ""since the performance of CP approach heavily depends on the complexity of the model" which was the authors' perspective without careful proof. We are sorry for the careless expressions. At the meantime, we removed some other expressions without sound evidences. We are sorry for the unprofessional expressions in last version.

Thanks to the reviewers' comments, we have noticed the close connection of our work and the CP Optimizer. In the introduction, we briefly introduced the CP Optimizer and carefully placed the citation

c) Laborie, P., Rogerie, J., Shaw, P., & Vilím, P. (2018). IBM ILOG CP optimizer for scheduling. *Constraints*, *23*(2), 210-250.

4) Introduction: "Worse still, separate operations …": the hint that uncertainty could be incorporated by this paper is a bit dangerous, since no uncertainty is considered in the input data or in the model, as far as I can see.

Answer.

We are sorry for the misleading word "uncertainty'. We were supposed to mention the changes that occasionally happened in the manufacturing environment such as the machine breakdown, rush order, etc. we have removed the word "uncertainties" and just mentioned the "changes in dynamic manufacturing environment", following the illustration in Leung, et al.'s article (Leung, Wong, Mak, & Fung, 2010).

5) Introduction: last paragraph: "This paper reports …" would most probably better fit to the former paragraph. -> suggestion: the last paragraph should only contain "This paper is organized …"

Answer.

Thanks for the reviewer's suggestion. We have reorganized the two paragraphs. The objective statement is merged into the introduction of the constraint programming. The organization of this paper is put in a separate paragraph.

6) Related work paragraph 3: the abbreviation for JSP is introduced although it has already been used before

Answer.

In last version, we repeated the full names of important terms for several times. We thought it might enhance the readability especially for readers unfamiliar with this area. However, we noticed it might be confusing, thanks to the reviewer's comment. We have updated the use of abbreviations such as IPPS, CP and JPS. In current version, the full names for abbreviations are only given at the first time they appear in the main content.

7) Related work: in general, recent new works are missing. Nearly all citations concern

works which have been published more than five to 20 years ago. However, there are many recent works on approximation and CP methods for scheduling problems and also on scheduling with flexibility, which have to be included (e.g. Laborie et al. 2018, Kreter et al. 2018, 2017, Tao & Dong 2017, Burgelman & Vanhoucke 2018 and many more)

Answer.

We have almost rewritten the literature review. The most recent articles including those suggested by the reviewer have been reviewed, including

- ✧ Barzanji, R., Naderi, B., & Begen, M. A. (2020). Decomposition algorithms for the integrated process planning and scheduling problem. *Omega*, 93, 102025.
- ✧ Ham, A. M., & Cakici, E. (2016). Flexible job shop scheduling problem with parallel batch processing machines: MIP and CP approaches. *Computers & Industrial Engineering, 102*, 160-165.
- ✧ Jin, L., Zhang, C., Shao, X., & Tian, G. (2016). Mathematical modeling and a memetic algorithm for the integration of process planning and scheduling considering uncertain processing times. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 230*(7), 1272-1283.
- ✧ Kreter, S., Schutt, A., & Stuckey, P. J. (2017). Using constraint programming for solving RCPSP/max-cal. *Constraints, 22*(3), 432-462.
- ✧ Laborie, P. (2018). *An update on the comparison of MIP, CP and hybrid approaches for mixed resource allocation and scheduling.* Paper presented at the International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research.
- ✧ Laborie, P., Rogerie, J., Shaw, P., & Vilím, P. (2018). IBM ILOG CP optimizer for scheduling. *Constraints, 23*(2), 210-250.
- ✧ Li, X., Gao, L., Pan, Q., Wan, L., & Chao, K.-M. (2018). An effective hybrid genetic algorithm and variable neighborhood search for integrated process planning and scheduling in a packaging machine workshop. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 49*(10), 1933-1945.
- ✧ Luo, G., Wen, X., Li, H., Ming, W., & Xie, G. (2017). An effective multi-objective genetic algorithm based on immune principle and external archive for multi-objective integrated process planning and scheduling. *The International Journal of Advanced Manufacturing Technology, 91*(9-12), 3145-3158.
- ✧ Tao, S., & Dong, Z. S. (2018). Multi-mode resource-constrained project scheduling problem with alternative project structures. *Computers & Industrial Engineering, 125*, 333-347.
- ✧ Zhang, S., & Wang, S. (2018). Flexible assembly job-shop scheduling with sequence-dependent setup times and part sharing in a dynamic environment: Constraint programming model, mixed-integer programming model, and dispatching rules. *IEEE Transactions on Engineering Management, 65*(3), 487-504.
- ✧ 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

We hope these articles can provide a picture about the recent research in relevant areas.

8) Section 3: Please be carefully when describing parameters etc. in the running text: they should always be of the same format (e.g. at the beginning the "m" and "n" differs from later on used formats)

Answer.

We are sorry about the inconsistency in the parameters' formats. We have updated the formats by using the MathType format manager.

9) Section 3.1: The textual descriptions of Fig. 1-2 should be read through again and thought about explaining the figures more precise. Please do not start a sentence with "And …". The dummy nodes should probably be mentioned at the beginning of the whole description since the dummy start also is the first node in the graph. The name of Fig. 2 should be rethought - as it is an AND/OR graph.

Answer.

We have enhanced the textual descriptions of Fig.1-2, as given in the first two paragraphs of Section 3.1. The language is enhanced as well. As suggested by the reviewer, we moved the description of dummy start and end nodes to the beginning of description of Fig. 2. We have updated the caption of Fig. 2 as suggested. Many thanks.

10) Section 3.1: The mentioning of AND/OR graphs should be already done in the literature review in Section 2. Moreover, there are much earlier works on AND/OR relations graphs (e.g. Pritsker 1966)

Answer.

We have carefully reviewed the paper entitled "GERT-Graphical evaluation and review technique" authored by Pritsker (1966). We agree it is an important work, but the contribution of this work to

our manuscript is limited. As a result, we haven't included this paper in the literature review, but we have to say it is an important work. Instead, we carefully reviewed articles Crowston (1970), Mello and Sanderson (1990), Liu (1995), J Christopher Beck and Fox (1999), Barták and Cepek (2007), Moffitt, Peintner, and Pollack (2005), etc., focusing on the representation of alternatives for planning and scheduling problems.

11) Section 3.1: The sentence "Coupling the three types of flexibility …" should be reformulated.

Answer.

We are sorry for the ambiguous statement. We have removed this sentence and put the description of the three types of flexibility in a separate paragraph in Section 3.1.

12) Section 3.1: The three bullet points with regard to Kim et al. 2003 raise several questions: (1) if these contents already exist, why again examining it -> there should be a clear differentiation to this work; (2) bullet point 2 is not depicted in Fig. 2; (3) bullet point 3 does not comply with Fig. 2 -> if o2,3 and o2,4 can be arranged in a different sequence as described in the text, the depiction in Fig. 2 has to be changed in a way that represents this fact.

Answer.

Kim et.al. (2003) just simply describe the three types of flexibility. Since they are important features of the IPPS problem, so we repeated them in this paper and added some case-specific explanations. We have rewritten the description of processing flexibility to make it clearer.

Bullet point 2 – the operation flexibility is easy to be perceived. As a result, it is not presented in Fig. 2 which is mainly for the explanation of the AND/OR structure for the IPPS problem. Instead, the sample data As given in Table 3 can clearly reflect the operation flexibility.

Bullet point 3 – the sequencing flexibility is represented by the AND-relation between operations. For instance, the AND-structure $o_{2,2} \rightarrow o_{2,3}$ and $o_{2,2} \rightarrow o_{2,4}$ implies the sequencing flexibility since $o_{2,3}$ and can be arranged in different orders for processing.

13) Section 3.1: "The IPPS problem … NP-hard." -> a citation has to be added.

Answer.

Actually, until now we haven't found a strict proof for the NP-hardness of the IPPS problem. But it is obvious that IPPS problem is NP-hard since the it combines the process selection to the traditional JSP problem which is proved NP-hard. We have put the simple explanation of the NP-hardness of IPPS problem in the introduction.

14) Section 3.2: Please reformulate the sentence "The processing time of all dummy nodes …"

Answer.
We have reformulated it to be "Both or-initiator and or-terminator are dummy nodes. All dummy nodes do not consume manufacturing resources. Therefore, their processing times are equal to zero." Which we hope could be better.

15) Section 3.2: "It is shown above that …" - for every figure, a direct reference including the Fig. and number should be included.

Answer.
We are sorry for the unprofessional writing style. We have updated the references.

16) Section 3.2: Table 1 - please add missing values in the description of the sets J and M; moreover, "K" is completely missing.

Answer.
We have completed the description of sets *J* and *M*. The index systems have been updated as well. In this update, *j* is for job, *u* and *v* for node, and *k* is for machine. We have added these descriptions of indexes in the first two rows of Table 1.

17) Section 3.2.1 - Section 4: The content and also the order of the different contents should be changed in order to have a better overview of all developments. To name just a view suggestions: (1) Is it really necessary to only include the assumptions in 3.2.1? (2) Why are there three different tables for notations which all belong to one,

respectively two, CP models? (3) the function "presenceOf" out of the CP Optimizer is no notation, as it is described in Table2. The function "presenceOf" is also no decision variable, as it is described in 3.2.2. It is an expression/function and the decision variable here is called "interval". (4) Why are the functions sometimes abbreviated (e.g. start(I) or preOf(I)) and sometimes not (e.g. alternative(I(o …))? (5) Why is the description of the hypothetical instance placed before the models are presented? Although most probably the introduction of constraints 14 is the reason, this order is still questionable. (6) Fig. 3 and all including explanations are questionable: In general, a figure including only the most important and thus, most different to understand functions should be presented, since for example the function "presenceOf" is very common in CP (not only for the CP Optimizer) and thus, has not to be displayed that extensively. In contrary to that the alternative function has CPLEX-individual features in it and therefore would be appropriate for Fig. 3. Moreover, a detailed explanation of the CP model, including the original names of all functions and expressions, should make it also unnecessary having many "preOf" explanations in Fig. 3. (7) The section name "Implementation" is questionable, as an additional model is established. (8) In general: For 1 or 2 new CP models which nearly are the same, the standard procedure follows the one of the presentation of MIP Models: (a) Explanation of the problem, including assumptions; (b) Explanation of new developments, e.g. the alternative function etc. including a graphical representation; (c) notations; (d) the new model(s); (e) an illustrative (artificial) example, including optimization solutions.

Answer.

Thanks very much for the comments and questions.

a) We have merged the previous Section 3 and 4, and rearranged the contents. In current version, the main content in Section 3 includes:
   i. The problem description and AND/OR graph definition, together with a simple IPPS example.
   ii. The problem-oriented assumptions.
   iii. The AND/OR graph for IPPS problem, and the graph-oriented assumptions.
   iv. The notations of IPPS elements.
   v. The decision variables and constraints built in CP Optimizer.
   vi. The model formulation.
   vii. A simple case for the implementation and a instantiation of the CP model.

   We hope the structure of Section 3 could be clearer now.

b) The assumptions on IPPS problem are inherited from previous paper, which we think suffice the typical IPPS problem. Besides, we have added three assumptions on the AND/OR graphs to assure the validity of using AND/OR graphs to represent IPPS problems.

c) Table 1 is for the IPPS elements, or data to be used in the model, while Table 2 summarizes the variables and relevant functions/expressions.

d) The presenceOf is usually regarded as a function to express the presence of intervals. In our model, we did arithmetic calculation on a set of presenceOf expressions. Indeed, the solution algorithm has to decide values for the presenceOf of each optional interval. As a result, we regarded it as a decision variable.

e) We are sorry for the abbreviated functions. We attempted to save spaces, but we didn't notice it might cause confusion. In current version, the standard function names built in CP Optimizer are used.

f) We attempted to show the process of projecting the AND/OR graph to CP models before giving the whole model which includes more constraints such as no-overlapping operations of the same job. That's why we put the instance given in Fig. 3 before the whole model.

g) The AND/OR graph carries not only the precedence relations but also the presence relations between operations. It is the key to our approach. That's why we gave Fig. 3 to illustration the translation from AND/OR graph to CP constraints. We are sorry for the misleading expression preOf() in last version. We have updated all expressions in the manuscript to make them consistent to standard definitions.

h) We have removed the section named "implementation" and merged it to Section 3. Currently it is just a subsection to propose a way to instantiate the model with IBM ILOG CPLEX Studio. The applied CP model with type-definition can actually simplify the formulation since most constraints work on the same set of operations.

i) We appreciate the reviewer's suggestions. In this version, we have rewritten the main body of Section 3.

18) Section 5: A sentence should not be started with a number (18). Moreover, please do not start a new paragraph with only one sentence ("A Desktop …)

Answer.
We have rewritten the first paragraph of this section (Section 4 in current version). We are sorry for the unprofessional writing.

19) Section 5.1: As two CP models are presented in the former sections, for both models, results should be presented for comparison reasons. If this is not possible, it should be clearly explained why not.

Answer.
The applied model (second model) just proposes a way to instantiate the graph-based model base on the type definition. In the preparing the data file, the type-definition allows defining only one set of operations and distinguishes them by operation's type. It is only for easy implementation. As a result, the two models generate exactly the same solutions for all problems. That's why we just reported one group of solutions for the proposed approach. We have clearly made the statement in Section 3.2.3.

20) Section 5.1: "FailLimit" should be explained in more detail: Is this the fail limit for the RestartGrowthFactor or a different one? There is no unit given for the value of 200,000. Moreover, at the end of section 4, a different FailLImit of 20,000 is described.

Answer.
The FailLimit is a parameter built in IBM ILOG CP Optimizer that limits the number of failures allowed before terminating a search. Setting the FailLimit usually depends on the problem scale. In this experiment, the FailLimit is set to be 500,000 for all problems to test the performance of the model with restricted computing resources. The FailLimit setting for the simple IPPS case in Section 4 (Current Section 3) is now removed to avoid confusion. The CP Optimizer can find the optimal solutions and automatically terminate the search in seconds.

21) Section 5.1: The experiments have to be extended: Two comparisons are missing in general - the comparison with the work of Zhang & Wong 2016 (ACO published in "Information Sciences" and Zhang & Wong 2018 (Enhanced ACO). Moreover, only the mean makespan results are presented. In all other papers on the IPPS, also the best makespan in an extra results table is presented. Please conduct these missing

experiments and insert them as an extra table. Moreover, the best result per single instance should be highlighted and not only the optimal ones (e.g. a bold formatting of the best result per instance and an additional asterisk for optimal results).

Answer.
Thanks for the suggestions. We have included the results of Enhanced ACO (Zhang & Wong 2018) in the comparison. With regard to the ACO (Zhang & Wong 2016), it is mainly focused on the general framework for the constructive metaheuristic and the ACO is used as an instantiation. Besides, we think that the enhanced ACO (Zhang & Wong 2018) could be a representative of this kind of algorithm dedicated to solve the IPPS problem. As a result, we don't include the ACO (Zhang & Wong 2016) in the comparison.
We have added a table (Table 7 in current version) to report the comparison of best makespans. The best result per single instance is highlighted by bold font. Thanks again for the suggestions.

22) Section 5.1: "Since the CPLEX solver implements a deterministic search strategy …":

This is not fully true - it has to be added there are parts with non-deterministic search strategies (see CPLEX CP Optimizer descriptions and CP forum of IBM)

Answer.
We learnt a lot more about the CP Optimizer and the search strategies, thanks to the reviewers' comments. We have updated the IBM CPLEX Studio to Version 12.10 which performs aggressive diving strategy in the search tree without backtrack (Laborie et al., 2018). We have corrected the description about the CP Optimizer.

23) Section 5.1: "… for which lower bounds are reached" -> the mentioning of this fact is unnecessary as it is clear that an optimal solution complies with the lower bound.

Answer.
We attempted to simply explain the optimality of solutions in the experiment. The lower bounds cannot be proved tight. At the beginning of Section 4 we illustrate the way to calculate the lower bounds. Thanks for the suggestions. We have removed the unnecessary statement.

24) Section 5.1: Description of Fig. 5: the description of course is true; however, it should be added that one can see (due to different instance characteristics?) that the search procedure however behaves differently for the different presented 4 instances.

Answer.
Thanks for the suggestion. The search procedure is homogeneous as we can see from Fig. 5. The search dives quickly at the early stage to near optimal solutions for all four problems. For more complex problems 15, 21 and 24, the algorithm can continuously explore better solutions. That is the key information that the authors would like to deliver to readers.

25) Section 5.2: "setting-up" should be replaced by a better fitting academic language.

Answer.
We are sorry for the casual word. We have updated it to “setup”.

26) Section 5.2: The results table should be of the same style as Table 5, i.e. lower bounds etc. should be added to have a good overview of the results (since here really better results are reached, this should be pointed out by including the contents of Table 5). Moreover, the information of ">1000" for the total runtime(s) is not clearly described.

Answer.
Thanks for the suggestions. We have completed the Table 7 for the comparison of best makespans. The “>1000” means the runtime is greater than 1000 seconds, which have been clearly explained at the end of Table 7.

27) Section 5: Why is CPLEX 12.5 used? In the meantime, already 12.9. is available, providing improvements also for the CP Optimizer. Thus, results generation with the newest version is suggested.

Answer.
We have updated the CPLEX to 12.10. The experiments are repeated and the data have been updated. Many thanks for the suggestion.

28) Section 5.1+5.2: For all results tables, a description should be added which gives information about every column, including unit descriptions (e.g. "s" for seconds etc.)

Answer.
We have added detailed description of each table in the mentioned sections (Section 4.1 and 4.2 in current version). Thanks for the suggestion.

29) Conclusion: Although there are mistakes concerning the English language in all sections, the conclusion includes a lot more than the others - this has to be fixed. Compared to the results section, the conclusion is quite long - however, it should be brief and concise. Perhaps, it could be shortened more (especially the description of one hypothetical instances is perhaps not necessary for a conclusion) -together with extending the results section by more experiments and more results tables and descriptions. Besides, to call a CP model a hybrid approach is a bit questionable. CP is an exact approach and the graph-based modeling itself, such as it is also used e.g. for the RCPSP, cannot really be seen as hybrid, it is an exact model for the CP Optimizer.

Answer.
We would like to thank the reviewer's time and comments. We have rewritten important paragraphs and tried out best to make the language clearer and more precise. We have rewritten the conclusion section which is much shorter than last version. We have extended the results' section. We have removed the announcement of a "hybrid" approach since we noticed that the AND/OR graph is the input to the Type-2 IPPS problem as reported by the latest article (Barzanji, Naderi, & Begen, 2020). We have updated the description about constraint programming. All in all, we highly appreciate the time and effort that the reviewer spent on this manuscript.

Reviewer #2:

The article proposes an approach for solving (with an exact approach) an integrated planning and scheduling problem formulated as an AND/OR graph. This problem is very relevant to combinatorial optimization and operations research.

The manuscript is reasonably well written and easy to read though (1) more details could be given about the CP concepts used in the formulation and (2) the structure of the paper could be slightly improved to make it clearer and avoid some redundancy (more on this below).

In general, I think the contribution is interesting and could be accepted provided some major revisions that would really improve the paper. I review below these points that need improvement, by decreasing order of importance.

1. Clarify and formalize the representation of the problem as an AND/OR graph

1.1 AND/OR graph

a) The AND/OR graph formalism that is used to represent the problem is borrowed from [deMello&Sanderson-1986]. I think the exact structure and semantics of this graph should be presented in a more formal manner in the article, rather than just illustrating it on the example (but the example is really useful and of course should be kept). More precisely:

b) - what exactly is an "OR-link" ? is it a tuple (os, {ot_1,...,ot_n}) where 'os' is the source node/operation and 'ot_i' are a set of target nodes/operations?

c) - is it possible to have several "OR-links" with the same source node/operation? For instance after operation 'a' you need to perform 'b' or 'c' AND 'd' or 'e' ?

d) - what if the graph is more complex for instance if the OR-links have several targets in common? For instance a->AND{b,c}, b->OR{d,e}, c->OR{e,f}. What are the possible realizations of this graph? Can operation e be executed twice if it is selected for both OR links b->OR{d,e} and c->OR{e,f} ?

e) - does the formalism allows representing 'optional' operations, for instance if 'o' is an optional operation between 'a' and 'b', can this be represented as a->OR({o,b}), o->b

f) - when some "OR-links" converge on the same target, like the OR links arriving at operation o_1,5 on Fig. 2, it is not clear why these arcs are not just "simple" arcs only representing precedence constraints when the operations are executed

g) - can you have at the same time some "OR" and "AND" arcs departing from (resp. arriving at) the same operation?

While the example on Fig.2 is very useful, I think it is very particular and cannot replace a formal mathematical description of the structure.

Answer.

Thanks very much for the comments and suggestions.

a) We have carefully revised the AND/OR graph definition for the IPPS problem, which can be found at the beginning of Section 3.2. We have exerted three assumptions on the AND/OR graphs

for the validity. Which are

(1) The arcs that start from or go to a same node can have either and-relation or or-relation but not both.

(2) An or-branch has only one start node and one end node.

(3) The branches can be disconnected by removing the or-initiator and or-terminator.

b) with regard to the OR-link, we define it as special set $OR\{o_{j,u_1}, o_{j,u_2}, ..., o_{j,u_n}\}$ with mutually-exclusive start nodes of all or-branches. We put a prefix OR before the set notation to distinguish the OR-link from a normal set. It is reasonable if the assumptions (1) and (2) hold. The OR-relation are usually defined among a set of arcs which originate from the same or-initiator and lead different or-branches. Because all arcs involved in the same OR-relation have a common source node, we just put the target nodes together as the OR-link. It can substantially simplify the model formulation. Meanwhile, a hyperarc $o_{j,u_0} \to OR\{o_{j,u_1}, o_{j,u_2}, ..., o_{j,u_n}\}$ can be defined for which the o_{j,u_0} is the common or-initiator. Likewise, the OR-link $OR\{o_{j,v_1}, o_{j,v_2}, ..., o_{j,v_n}\}$ can be a set of mutually -exclusive end nodes of all or-branches. We define the hyperarc $OR\{o_{j,v_1}, o_{j,v_2}, ..., o_{j,v_n}\} \to o_{j,v_0}$ which goes from an OR-link $OR\{o_{j,v_1}, o_{j,v_2}, ..., o_{j,v_n}\}$ to an or-terminator, to represent the or-branches cluster to the same or-terminator o_{j,v_0}.

In the authors' opinion, defining the AND-link is not helpful for the model formulation. It can even make the structure overcomplicated. In the model, just OR-links should be taken care of. As a result, just OR-links are defined.

c) It is impossible to have several OR-links connected to the same or-initiator, restricted by the assumption (1) because different OR-links will have an AND-relation. If it is inevitable in formulating the IPPS problem, we can isolate each OR-link by adding a dummy or-initiator to lead them, which will not distort the relations among operations. For instance, if $a \to OR\{b, c\}$ and $a \to OR\{d, e\}$ is necessary, we can add two dummy nodes f_1 and f_2, and reconstruct the graph to be $a \to f_1, a \to f_2$, $f_1 \to OR\{b, c\}$, $f_2 \to OR\{d, e\}$.

d) We have assumed (3) to avoid such illegal structure. Take the instance suggested by the reviewer as an example. The structure a->AND{b,c}, b->OR{d,e}, c->OR{e,f} will violate assumption (3) since the by removing the or-initiator *b*, the or-branch starting from *e* is still connected to the graph via *c*. Such structure is illegal in representing IPPS problems.

e) It is similar to the problem d). With the assumption (3) such structure can be avoided.

f) In the formulation, all intervals are optional. Since there may be hierarchical OR-structures, or say, an or-branch is split to several sub-or-branches, an or-terminator might be optional. As a result, the presence of an or-terminator depends on the presence of its preceding OR-link. That is why we define the hyperarc $OR\{o_{j,v_1}, o_{j,v_2}, ..., o_{j,v_n}\} \to o_{j,v_0}$ instead of simple arcs.

g) According to the newly-added assumption (1), it is impossible to have "AND" and "OR" arcs

that depart from (resp. arrive at) the same operation. If it is inevitable, the method mentioned in c) can always properly reconstruct the graph.

1.2 Assumptions

Section 3.2.1 presents some assumptions on the problem being addressed (non-preemptivity, machine can perform only one operation at a time, operations of the same job cannot overlap).

First I think that these assumptions should be described earlier in the description of the problem (so with the current structure of the paper, in should be in section 3.1 but I really think that the problem should be formally and completely described, including the AND/OR graph, the resources/machines allocation and the objective function in a dedicated section followed by an illustration with the example) rather than in the section 3.2.1 that describe how the problem is translated into a CP formulation. Indeed, these restrictions are not related to CP and a CP solver like CP Optimizer can easily relax most of these assumptions (for instance using cumul functions for cumulative resources, or sequence-dependent setup times on machines, etc.).
Also, I'm not sure that the concept of "job" is really useful. In some real problems, operation of a job are not required to be fully ordered and they can overlap. And it seems that with the proposed representation, you could easily add an additional dummy machine that would be used by all the operations of a given job in case operations of this job are required not to overlap.

Answer.

We have rewritten the problem description (the beginning of Section 3), the AND/OR graph for the IPPS problem (the beginning of Section 3.1), and partial assumptions on the problem. We have put the assumptions on the problem in Section 3.1, after describing the AND/OR graph just because we need examples to illustrate each assumption. Thanks to the reviewer's comments and suggestions, the Section 3 has been substantially improved.

With regard to the no-overlap assumption on the job, we inherited the problem setting from existing literature in related areas such as JSP, Flexible JSP, and IPPS. Our model targets single-piece/small batch production for which a job is usually to produce one product or one part, working on the same stock. Usually a stock can only be processed by one machine like lathe, CNC, etc. at a time. We think that is the main reason for the no-overlap assumption on the job.

Regarding the non-preemption assumption, it is mainly due to the fact that setup is usually costly. Re-setup for some features is even impossible. The non-preemption assumption can avoid multi-setups for the same feature, which substantially simplifies the actual production. As a result, we think it is a reasonable assumption.

The term "job" is inherited from existing literature. A job usually corresponds to producing one part, a product, etc. It is a cluster of interrelated operations. Different jobs are usually independent from each other. We think the concept "job" can simplify the problem's description and model formulation. If we use dummy machines, we have to assign a dummy machine to each cluster of operations. Its function is much similar to the "job" but it is not that straightforward. As a result,

we would like to keep the “job” in this manuscript.
However, the reviewer proposed a pretty promising approach to generalize the IPPS problem. We would like to try to find a corresponding industrial case. We have mentioned the generalization of current model in the conclusion and future work. The reviewer does provide a guideline.

2. Clarify and complete the part about related work

2.1 Work related with AND-OR graph representation

There are several representations that are close to the AND-OR graph one that could be mentioned in the section about related work. In particular:
- XorNode/PEX in [Beck&Fox 1999]
- DTPFD in [Moffitt,Peintner&Pollack 2005]
- P/A Graphs in [Bartak&Cepek 2007]
- HTN planning approaches [Georgievski&Aiello 2015]

[Beck&Fox 1999] Beck, J. C., and Fox, M. S. 1999. Scheduling alternative activities. In Proc. AAAI-99.

[Moffitt,Peintner&Pollack 2005] Moffitt, M. D.; Peintner, B.; and Pollack, M. E. 2005. Augmenting disjunctive temporal problems with finite-domain constraints. In Proc. AAAI-2005.

[Bartak&Cepek 2007] Bartak, R., and Cepek, O. Temporal networks with alternatives: Complexity and model. In Proc. FLAIRS-2007.

[Georgievski&Aiello 2015] Georgievski I., and Aiello M. HTN planning: Overview, comparison, and beyond. Artificial Intelligence Volume 222, May 2015, Pages 124-156.
Answer.
The reviewer suggested 4 important articles. We have carefully reviewed all of them and properly cited three of them: XorNode/PEX in [Beck&Fox 1999], DTPFD in [Moffitt,Peintner&Pollack 2005], and P/A Graphs in [Bartak&Cepek 2007]. With regard to the HTN planning approaches [Georgievski&Aiello 2015], we think it is not closely related to the main topic discussed in this paper. As a result, we haven’t cited it.

2.2 CP approaches

a) In the literature about CP approaches for scheduling problems and in particular as far as the integration of planning and scheduling is concerned, I think that CP Optimizer (the solver used in the study) has a particular role because it is the only CP framework that introduces and fully implements the notion of 'optional interval variable' which is central in the translation of the AND-OR graph. In fact the concept of 'optional interval variable' was introduced in [Laborie&Rogerie, 2008] for providing a generic tool to generalize the

existing approaches like XorNode, P/A Graphs or AND/OR graphs. Unlike many other CP frameworks, CP Optimizer is used to solve much more complex real industrial scheduling problems than just RCPSPs of typical jobshop scheduling problems (unlike mentioned in the paper). And many of these complex problems involve a mix of planning, resource allocation and scheduling. I would suggest the authors to read a recent overview of CP Optimizer
([Laborie&all,2018], in particular there is a small section on recent scheduling applications using CP Optimizer) and/or just search for ""CP Optimizer" OR "CPOptimizer" OR "ILOG CP"" on Google Scholar to have an idea of the range of different problems solved using this framework.

[Laborie&all,2018] Laborie P., Rogerie J., Shaw P. & Vilim P. IBM ILOG CP optimizer for scheduling. Constraints. April 2018, Volume 23, Issue 2, pp 210-250

b) I think the statement (end of section 2) that there is until now no literature reporting successful applications of CP for IPPS problems should be softened. Same think for the statement in section 1: "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." Furthermore, you cannot really say that the proposed IPPS problem is more complex than the RCPSP as it (currently) only handle disjunctive scheduling whereas the RCPSP works with cumulative resources which are, from this aspect, more general.

c) A detail: in the introduction: "Some global constraints like nooverlap(.) , alternative(.) and endbeforstart(.) provide substantial convenience to model scheduling problems, and efficient algorithms are available in literature to solve them (Barták, 2003).": these constraints, and in particular the alternative(.) one, are typically CP Optimizer constraints and I don't think they were mentioned in (Barták, 2003). Also, endbeforestart(.) is not a global constraint as it only holds between 2 interval variables.

Answer.

a) We would like to sincerely thank the reviewer for the profound comments and suggestions. The recommended paper [Laborie&all,2018] and other articles about CP Optimizer help us understand much more about the CP and CP Optimizer. We have carefully revised all statements about CP. We have updated the description about interval variables and constraints which are CP Optimizer's components. Besides, we stated that the IPPS is a complex problem in last version. We are sorry that we just compared it to typical JSPs and Flexible JSPs. With the help of the reviewer's comments, we noticed it was an imprecise statement. As a result, we have updated the description of the complexity of IPPS problem, related it to JPSs (in introduction).

b) We have softened the statement "that there is until now no literature reporting successful applications of CP for IPPS problems" at the end of Section 2. We just wanted to state that

there are no CP models for the specific IPPS problem described in this paper. We have noticed that such statement is misleading. We totally agree that RCPSP is more general. Besides, there are other categories of scheduling problems which are really complex. It is improper to make such comparison. As a result, we have removed the imprecise statements about the complexity of IPPS problem.

c) We are sorry that we didn't notice the differences between CP and CP Optimizer specific components. We have carefully reviewed the recent work on CP Optimizer and corrected the description about constraints built in CP Optimizer. It was an error to add the constraint endbeforestart(.) in the global constraint list. We have removed it.

3. Give a bit more detail about the CP Optimizer concepts

The CP model maybe a bit difficult to understand for a reader that do not know the main concepts of the CP Optimizer model. In particular:

- The interval variables could be introduced a little bit more formally by defining their domain (that include a specific 'absence' value), and also mention that here you create some (optional) interval variables by specifying there size (machine dependent size)
- The constraints used in the model could also be introduced more formally (in particular how they behave in case of absent intervals), there are not many of them: preof(x), endbeforestart(x,y), nooverlap({x_i}), alternative(x,{y_i})

Also, it would be useful to state somewhere that CP Optimizer is a "model&run" optimization engine, like in MIP solvers you only need to formulate the problem, then the resolution is performed by an automatic search. This is also a difference between CP Optimizer and other CP frameworks where you often need to design your own search strategy.

Answer.

Thanks very much for the suggestions. We have placed the introduction of CP Optimizer in several places of the paper.

1) In the introduction, we introduced the CP Optimizer as "a model-and-run optimization engine that provides intuitive modelling language for scheduling problems based on the built-in interval variable and constraints such as $nooverlap(\cdot)$, $alternative(\cdot)$ and $endbeforstart(\cdot)$ (Laborie, Rogerie, Shaw, & Vilím, 2018)." We also mentioned that the CP Optimizer is also able to perform automatic search with built-in algorithms.
2) We have adopted the definition of interval variable by Laborie & Rogerie (2008) together with the domain of intervals $dom(I) \subseteq \{\perp\} \cup \{[s,e) \mid s,e \in \mathbb{Z}, s \le e\}$.
3) We have stated in Section 3.2.1 that there are two types of intervals for the IPPS model: $I(o_{j,u})$ the interval for processing operation $o_{j,u}$ with undetermined size, and $I(o_{j,u},k)$ the interval for processing $o_{j,u}$ on machine k with fixed size equal to $\tau(o_{j,u},k)$.

4) In Section 3.2.2, we have briefly introduced the constraints $noverlap(C)$ with a set of intervals C as the input, $endbeforestart(I_1, I_2)$ with two intervals I_1 and I_2 as the inputs, and $alternative(I, C)$ with an interval I and a set of intervals C as the inputs.

5) In Section 3.2.3 for the case illustration, we have stated that the model is solved automatically by CP Optimizer. Besides, in the experimental configuration (Section 4), we have stated that the CP optimizer uses the newest iterative diving search method to perform aggressive dives in the search tree without backtrack (Laborie et al., 2018).

4. The version of CP Optimizer that is used in the experiments (12.5) is very old

This version is from 2012, so about 7 years old now. The CP Optimizer system is evolving quickly (latest version is 12.9), and in particular it made big progress on the capacity to provide optimality proofs. So it would really be interesting the re-evaluate the experiments using a more recent version.

Answer.

Thanks very much for the suggestion. We have updated the CP Optimizer to version 12.10 and re-evaluated the experiments. The results have been updated. At the meantime, we noticed that the CP optimizer uses the newest iterative diving search method to perform aggressive dives in the search tree without backtrack, which have been notified in the experiment configuration.

Other more detailed remarks/typos:

* In introduction: "Global constraints, as defined by van Hoeve and Katriel (2006), is a constraint that captures a relation between a non-fixed number." It seems that the end of the sentence is missing. I guess you want to say that a global constraint captures a relation between a non-fixed number of decision variables.

Answer.

We are sorry for the carelessness. Thanks to the reviewer's previous comments, we noticed that is might be improper to refer to the global constraints defined by van Hoeve and Katriel (2006) in the introduction, since our work is mainly based on the CP Optimizer. We have replaced the description about global constraints by the description about CP Optimizer in the introduction.

* In introduction in the same paragraph: "This paper reported" -> "reports"

Answer.

We are sorry for the misuse of grammatical tense. We have fixed this error.

* In introduction, last sentence: conclusion is section 6, not 7.

Answer.

Thanks for reminding the error. Since we have merged the Section 4 and 5, we have updated the description of the paper organization.

* In section 3.2.2, you introduce constraints on presence of intervals related with the selection of machines: (2) and (3). These constraints are not in the complete model ("The Graph-Based CP Model for IPPS"). I think that you should say somewhere that these constraints (2) and (3) are subsumed by the alternative(.) constraints (9). And it is indeed the right way to model machine allocation in the problem.

Answer.

Thanks for the comment. We have noticed the imperfection in our model. We notice that the constraint

$$alternative\left(I\left(o_{j,u}\right),\left\{I\left(o_{j,u},k\right)\mid k\in M\left(o_{j,u}\right)\right\}\right) \tag{1}$$

implies

$$presenceOf\left(I\left(o_{j,u}\right)\right)=\sum_{k\in M\left(o_{j,u}\right)} presenceOf\left(I\left(o_{j,u},k\right)\right) \tag{2}$$

As a result, constraint (2) is redundant. We have removed this constraint out of our model.

* I think that the part of section 4 that describes the "Applied Graph-Based CP model for IPPS" is mostly redundant with the general model provided in section 3.2.3. If you need to save some place, you could consider removing this part. But in this section, Fig 3 is really nice and helps understanding the formulation.

Answer.

Thanks for the suggestion. The Applied Graph-Based CP Model is an approach to instantiate the general model provided in Section 3.2.2. The instantiation is based on the type-definition for nodes. Although it does not contribute to the model directly, it provides an approach to formulate the model in a simpler way, which can make the implementation easier. As a result, we hope to keep the applied model.

* You should cite at some point in the article the recent paper [Laborie&all,2018] that provides a fairly complete overview of CP Optimizer for scheduling.

Answer.

We have carefully reviewed this paper and properly placed the citation. This paper has substantially elevated our understanding about the CP Optimizer and the CP framework. We are running out of words to express our gratitude to the reviewer.

* In the experimental section, I'm wondering why you used a FailLimit instead of a TimeLimit.

Answer.

In our opinion, the FailLimit is somewhat independent from the hardware environment in which the experiments are conducted. The FailLimit is more closely related to the problem scale, formulation, and search strategy. That is reason why we used FailLimit.

* Also, in the experimental section you say "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". While it is true that the search strategy is deterministic (this is required for a solver used in an industrial context), the search uses some randomized search and you can easily change the random seed used by the random generator (there is a search parameter RandomSeed) to produce run different executions (RandomSeed=1,2,...,n) and measure the robustness/stability of the resolution, just like for the other approaches.

Answer.

The reviewer's comment expands our horizon in the CP framework. We have carefully reviewed several important articles about the algorithms and search strategies implemented by the CP Optimizer. We have removed this misleading statement. Instead, we have stated that the CP Optimizer uses the newest iterative diving search method to perform aggressive dives in the search tree without backtrack (Laborie et al., 2018).

* In the references:
  - The reference to Bartak,Salido,Rossi 2010 is duplicated
  - The reference to de Mello, Sanderson 1986 is incomplete

Answer.

We used the Endnote to manage the citations and bibliography. We are sorry that we didn't notice the bugs. We have updated the references.

Reviwer#3:

This paper considers the integrated process planning and scheduling (IPPS) problem. The goal is using a graph-based constraint programming to solve the problem and to evaluate the model, a benchmark is used as comparison. The problem is quiet so interesting. I suggest accepting this paper with major revision. The reasons are presented as follows.

Major comments

1. Is that NP-hard problem? Please mention it.

Answer.

Yes. The IPPS problem discussed in this paper is NP-hard. It is known that traditional jobshop scheduling problem (JSP) is NP-hard (Sotskov and Shakhlevich (1995). The IPPS problem that combines partial process planning functions to jobshop type of scheduling problem is clearly NP-hard. We have stated the NP-hardness of the IPPS in the introduction.

2. Please show the advantage and disadvantage of the authors' model in the Section 6. Actually, the structure of manuscript is not so clear although it indeed has some contributions for the research area.

Answer.

We have rewritten important paragraphs and reorganized the manuscript. In current version, it has 5 sections: introduction, literature review, the model and a simple case, the experiments on benchmark problems, and conclusions & discussions. We have merged the previous Section 3 and 4.

We have rewritten the Section 5 for the conclusions and discussions. The proposed model handles the Type-2 IPPS problem with AND/OR graph as input. It directly projects the AND/OR graps to CP constraints. The AND/OR graph regulates both precedence and presence relations among nodes, which makes our model unique and distinguishable. As suggested by the reviewers, the limitation is notable. The assumptions on the IPPS problem could be relaxed to cater to more general scheduling problems.

3. How to have the lower bound in the Table 5? it should be mentioned clearly.

Answer.

The lower bounds are reported in existing literature. However, we cannot find its original source. In our research, we noticed that the lower bound is "the maximum of the smallest possible total processing times of each job". Let O_j^* be a set of operations applicable of fulfilling job j. The lower bound is calculated by

$$\max_{j\in J}\left\{\min_{all\ possible\ O_j^*}\left\{\sum_{o_{j,u}\in O_j^*}\min_{k\in M(o_{j,u})}\tau\left(o_{j,u},k\right)\right\}\right\} \tag{3}$$

We have included the lower bound calculation in Section 4 – Experiments. Thanks for the comments.

4. Is that benchmark with 24 problems generated by the authors or intended by other papers? Please mention it. If from the authors please let reader know how to generate? Is it appropriate for this problem?

Answer.

The 24 problems were generated by the Kim et al. (2003) which were used as benchmark problems in lasted decades. We have added Table 5 in current version to report the specification of the problems. This set of benchmark problems are purposely designed to test algorithms' performance in circumstances of different levels of flexibility. As a result, we think it is appropriate for this problem.

5. I don't think it is fair that the compared methods are metaheuristics. Please add some graph-based methods to be comparison. Qiao and Lv (2012)??

Answer.

We agree the reviewer's comments. However, to the best of the authors' knowledge, all existing models and exact algorithms targeted the Type-1 IPPS problem with enumerated process plans as input. This might be the first time that a model is proposed for the Type-2 IPPS problems. Qiao and Lv (2012) proposed a model which still formulates the Type-1 IPPS problem since they proposed an approach to prepare the enumerated process plans beforehand. As a result, we have to compare our approach to existing metaheuristics. Besides, we think it is somewhat meaningful to do this comparison, since metaheuristics and exact approaches have their own pros and cons. The comparison can provide a sense of performances of each approach.

We have compared our approach to the Improved GA (IGA) reported by Qiao and Lv (2012). But it is confusing that all citations downloaded from different sources present this reference as Lihong & Shengping (2012).

6. Please organize the paper let the reader can easier rewrite this model and use it to solve the similar problem.

Answer.

Thanks for the comment. We have made a great enhancement in the readability of this paper. We provided an approach to instantiate our model as well, which we hope will be helpful for the replication.

Minor comments

1. Please add some new related papers in Section 2.

Answer.

Thanks very much for the comment. We have reviewed the latest articles in this area and updated our references.

2. Please check the grammar for all manuscript.

For example,

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.

Answer.

We are sorry for the grammar errors. We have carefully checked the manuscript, including the piece mentioned by the reviewer.

3. Please provide other three existed algorithms' CPU time.

Answer.

The CPU time for the graph-based CP approach is included as a reference to show that our approach can solve the IPPS problem with reasonable computing effort. In the authors' opinion, the CPU time heavily depends on the hardware and software environment. It is quite sensitive to the real-time status of the computer. As a result, it is not quite worthy to compare the CUP times of different algorithms. As a result, we just included the CPU time for the proposed approach as a reference of consumption of computing resource.

4. Please mark the note if the problem is stopped in FailLimit. Is that a optimal solution if stopped in Fail Limit?

Answer.

Thanks for the comment. We have marked all optimal solutions in our table with a "*" mark. We have added Table 7 in current version which shows that the CP Optimizer continues exploring the solution space if the FailLimit is not reached, although optimal solutions might be obtained. Sometimes it is possible that the solution is not optimal when the search stops.

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

Integration of process planning and scheduling is to carry out the two functions simultaneously. This paper provides a graph-based constraint programming (CP) model based on the IBM ILOG CP Optimizer to solve the integrated process planning and scheduling (IPPS) problem in the jobshop environment. A tailored AND/OR graph structure is defined for the IPPS problem. The AND/OR graph defines not only the precedence relations but also the presence relations among nodes. Interval variables and scheduling-oriented constraints built in CP Optimizer are borrowed to project the AND/OR graph based IPPS problem to a concise CP model. An applied CP model based on type definition for nodes is provided for practical implementations. Incorporating minimization of makespan as the single objective, the graph-based CP model is tested on a set of benchmark problems. Experimental results show that the proposed approach outperforms compared algorithms in major IPPS instances. The performance of the proposed approach in solving IPPS instances of larger scales is even better.

Keywords

Integrated process planning and scheduling; constraint programming; CP Optimizer; graph-based modelling; AND/OR graph.

1. Introduction

Process planning and scheduling are two important functions in manufacturing. Process planning is to design a series of operations to finish all features of a product with consideration of manufacturing constraints and limited manufacturing resources. A process plan comprises the entire bundle of operations, operations' sequence constraints, and resource requirements. In conventional practices, process planning and scheduling are carried out sequentially. At first, a set of alternative process plans are generated at the process planning stage. The best process plan in terms of the planning goal is then selected. And then, scheduling is carried out to deal with the selected process plan and allocate manufacturing resources to all necessary operations. The sequential process planning and scheduling may ruin the global optimality of the planning work, since the two stages are usually implemented towards different goals. Worse still, conflicts may frequently occur. For instance, the process planning module cannot sense the real-time status of machines, in which case unavailable machines might be planned for some operations. It also implies that separately carrying out process planning and scheduling cannot flexibly respond to

changes in the dynamic manufacturing environment [1]. The IPPS is to perform process planning and scheduling simultaneously in a unified framework. There are several frameworks for the integration of process planning and scheduling, including an augmented scheduling system with embedded process planning module [2], a collaborative process planning and system [3], and the IPPS framework with AND/OR graph as input [4]. Modelling the process planning and scheduling as a single decision-making problem became the mainstream in recent years. The IPPS system takes a set of predetermined process plans as the input. It selects one process plan for each job and allocate machining resources simultaneously with regard to one or multiple common objectives. Due to the differences in organizing the predetermined process plans, the IPPS problems fall into two categories: the Type-1 IPPS problem that inputs enumerated process plans, and the Type-2 IPPS problem that inputs AND/OR graphs [5].

It is known that traditional jobshop scheduling problem (JSP) is NP-hard (Sotskov and Shakhlevich [6]. The IPPS problem that combines partial process planning functions to jobshop type of scheduling problem is clearly NP-hard. The problems' complexity of IPPS expands exponentially as more jobs, operations, or machines are involved. Existing articles reported a number of approximate approaches to get near-optimal solutions for the IPPS problem. Typical approximate approaches include a symbiotic evolutionary algorithm [4], an agent-based negotiation approach [7], a genetic algorithm [8], an improved genetic algorithm [9], an object-coding genetic algorithm [10] etc. Recently, two exact approaches, a mathematical modelling and a memetic algorithm [11], and a decomposition algorithm [5] for the IPPS were reported. However, both two approaches solve only the Type-1 IPPS problem. With regard to the Type-2 IPPS problem, packing all alternative process plans in a compact AND/OR graph structure can benefit the practical implementation and design of solution algorithms. The Type-2 IPPS problem needs more concern in current study. Until now, no available models can be found in literature for the Type-2 IPPS problem. More details about the two types of IPPS problems were discussed in in Barzanji et al.'s article [5].

Constraint programming (CP) is a powerful tool to solve 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 [12]. The IBM ILOG CP Optimizer is a model-and-run optimization engine that provides intuitive modelling language for scheduling problems based on the built-in interval variable and constraints such as $nooverlap(\cdot)$, $alternative(\cdot)$ and $endbeforstart(\cdot)$ [13]. The CP Optimizer is also able to perform automatic search with built-in algorithms. Existing literature reported constraint programming approaches for a wide categories of scheduling problems [14]. This paper will report a constraint programming approach for the Type-2 IPPS problem in the jobshop environment. A concise graph-based CP model for IPPS will be proposed, together with an applied model with node-type definition for the practical implementations.

This paper is organized as follows. In section 2, typical approximate algorithms for solving the IPPS problems are reviewed. The AND/OR graph representation and CP implementations in this field are discussed as well. The graph-based CP model for IPPS and a simple case are elaborated in Section 3. Section 4 illustrates experiment results on benchmark problems. Finally, Section 5 provides conclusions and discussions.

2. Related work

Representing alternative process plans with AND/OR graphs can be traced back to decades ago. Crowston [15] used AND/OR graphs in the scheduling problem to sketch the alternative jobs in manufacturing processes. Mello and Sanderson [16] proposed a compact AND/OR graph representation for the assembly plans to support the development of planning algorithms. Gillies and Liu [17] proposed two systems to schedule tasks with AND/OR precedence constraints on identical processors. Beck and Fox [18] provided a compact representation for scheduling alternative activities by integrating the XorNodes to indicate the start and end points of alternative activity paths. Barták and Cepek [19] proposed a general temporal network named P/A graph, a precedence graph with parallel and alternative branching. The alternative branching can be used to model the alternative manufacturing processes. A fan-like structure for the P/A graph was defined. Moffitt, Peintner [20] augmented the Disjunctive Temporal Problems by designating finite-domain component to each element of the problem. The finite-domain component reflects the choice among a set of options such as the possible venues for event scheduling problems.

Research in the IPPS field was mainly focused on the Type-1 IPPS . There exists several mathematical models for the Type-1 IPPS problem [9, 21, 22]. Type-2 IPPS organizes process plans in a compact graph structure, which provides flexibilities for the development of solution algorithms. Barzanji, Naderi [5] solved the multi-mode resource-constrained project scheduling problem with the consideration of alternative project structures. They constructed an AND/OR network for that problem and established a hybrid metaheuristic solution approach. The experimental results revealed the advantages of AND/OR network representation in solving the problem. However, until now the authors haven't found any models for the Type-2 IPPS problem.

With regard to the solution approaches, a number of typical heuristic algorithms have been successfully implemented to solve the IPPS problems. The heuristic approaches are usually not constructed to solve the mathematical models directly. Tao and Dong [23] applied a symbiotic evolutionary algorithm to solve the IPPS problem. In last decades, the genetic algorithms (GAs) have drawn massive attention in the IPPS problem domain. Successful GAs for the IPPS problem include a simulation based GA [4], a modified GA[24], an improved GA [25], an object-coding GA [9], etc. Zhang and Wong [10] combined the GA and variable neighbourhood search for the IPPS problem in a packaging machine workshop. An effective multi-objective GA was also proposed for the IPPS problem with multi-objectives [26]. Other approaches such as simulated annealing [27], particle swarm optimization [28], and ant colony optimization [29] were also applied to solve the IPPS problems, but benchmark tests show that GAs outperform the others. Besides, agent-based approaches were proposed for the IPPS problem. 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 [1, 30].

Constraint Programming (CP) have been applied to solve typical scheduling problems. Wong, Leung [31] 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. Baptiste, Le Pape [14] coupled an MIP model and a CP model to solve scheduling problems in two separate stages. Harjunkoski and Grossmann [32] proposed a CP-based approach for a multi-machine assignment scheduling problem. Some latest literature reported methods that combine CP and local search to increase the efficiency in solving complex

scheduling problems. For instance, Sadykov and Wolsey [33] reported an approach which coupled CP with the taboo search for the JSPs. Furthermore, Beck, Feng [34] established a theoretical framework that implies constraint satisfaction technologies for planning and scheduling problems. The integration of planning and scheduling was even considered. However, it is just a theoretical framework and cannot be used directly in practice. Bartak, Salido [35] established a more detailed model for general planning and scheduling problems with combined IP and CP approaches. Timpe [36] proposed a CP model for the flexible JPS with parallel batching machines. They also compared the CP model and an MIP model. The experiment results show that the CP outperforms the MIP in solving the given jobshop-type scheduling problem. Ham and Cakici [37] formulated a CP model and a MIP model for the flexible assembly job-shop scheduling problem in a dynamic manufacturing environment. It also revealed that the CP performs better than MIP in that problem category. Zhang and Wang [38] reported a standalone tiny CP model for the resource allocation and scheduling problem and solved with a recently-improved commercial CP solver. Experimental results showed that the CP model outperforms all previous approaches in the benchmark test of 335 instances. Laborie [39] proposed CP models to solve the Resource-constrained project scheduling problem in terms of minimizing project duration, and the extended benchmark tests showed that the CP is highly competitive compared to existing MIP models. Repeatedly researchers reported the outstanding performance of CP in solving scheduling problems.

As a powerful problem-solving and optimization tool, the CP has great potential to solve the IPPS problem. This paper will report a concise graph-based CP model with global constraints built in IBM ILOG CP Optimizer for the Type-2 IPPS problem.

3. The graph-based CP model 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. 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 set of operations. Processing operations of the same job may be restricted by precedence constraints. It is possible that an operation can be processed by different machines. Operations can be arranged in different orders to fulfil the same job [40].

3.1 An IPPS example and the 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 (F1.1, F1.2), a groove (F1.3), a hole (F1.4) and chamfer (F1.5). 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 (F2.1, F2.3), an outer diameter (F2.2), a slot (F2.7), a relief groove (F2.5), a thread (F2.6), and a chamfer (F2.4). 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 sequence constraints.

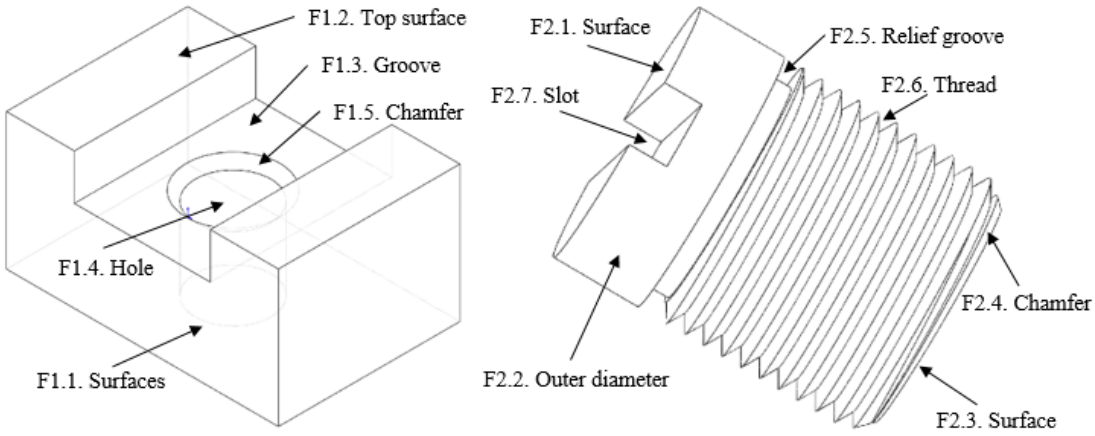

Fig. 1. An IPPS example with two parts.

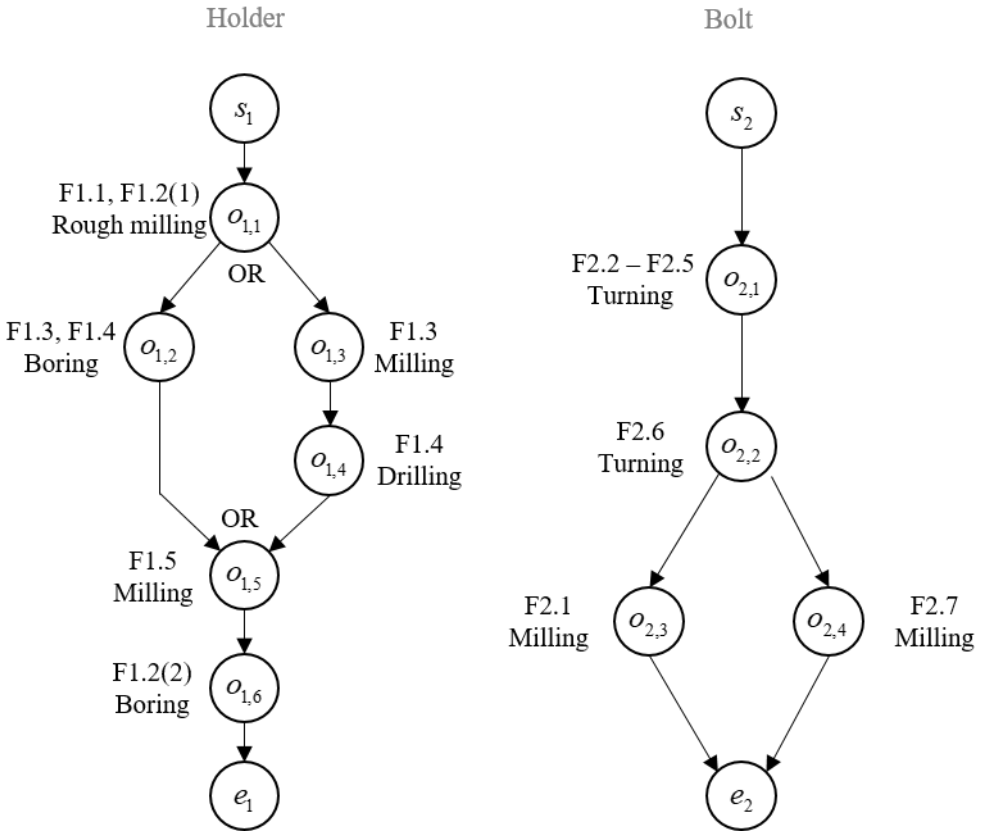

Fig. 2. The AND/OR graph for the IPPS example.

In Fig. 2, dummy nodes s_j and e_j are introduced to indicate the start and end points of job j. The operation $o_{1,1}$ is designed to finish Feature F1.1 and rough machining the top surface F1.2 with a single setup. Since the top surface requires a higher level of accuracy, another operation $o_{1,6}$ is designed to achieve it. Both the groove F1.3 and hole F1.4 can be done by either performing milling $O_{1,3}$ and drilling $o_{1,5}$ sequentially on a milling machine and driller

respectively, or performing a single operation $o_{1,2}$ on a boring machine with a single setup. Therefore, either the operation sequence $(o_{1,2})$ or $(o_{1,3}, o_{1,4})$ is enough to finish the two features. As a result, the two exclusive operation sequences have an or-relation. Reflected in the graph, there are two alternative paths after operation $o_{1,1}$. The alternative paths end at operation $o_{1,5}$ 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 sequence constraints.

Besides, for processing the bolt, either operation $o_{2,3}$ or $o_{2,4}$ can be started after finishing $o_{2,2}$. There is no sequence constraint between them. As the result, 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-relation. Since there are both or-relation and and-relation, the graph is defined as AND/OR graph [16, 30].

Kim, Park [4] used the AND/OR graph to represent their testbed for the Type-2 IPPS problem. Three types of flexibility are considered, which are

(1) Processing flexibility. It reflects alternative operation sequences for a job. As shown in Fig. 2, two alternative operation sequences $(o_{1,1}, o_{1,2}, o_{1,5}, o_{1,6})$ and $(o_{1,1}, o_{1,3}, o_{1,4}, o_{1,5}, o_{1,6})$ are both applicable of processing the handler. The two sequences have different sets of operations. No doubt the process sequences for the same job are mutually exclusive, that is, only one process sequence should be selected for a job and present in the final schedule.

(2) Operation flexibility. It is possible that more than one machine being 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.

(3) Sequencing flexibility. The same set of operations for the same job can be arranged in different orders. 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 both of them need to be processed and there is no sequence constraint between them.

Different types of JSPs take different combinations of flexibility. The IPPS problem is among the most difficult problems in this category for it takes all three types of flexibility. The IPPS discussed in this paper follows the general assumptions made for this category of problem.

Assumptions on IPPS problem [4]

(1) All operations are non-preemptive. A started operation cannot be interrupted until it is finished.

(2) Operations for the same job cannot be overlapped.
(3) A machine can perform at most one operation at a time, constrained by its capability.
(4) All jobs are released at time *zero*. Each operation can be started immediately after all its necessary predecessors are finished.
(5) Internal logistics and setup consumption are neglected or absorbed by processing times for operations.

The AND/OR graph representation enables establishing a concise and practical CP model for the IPPS problem. compared with existing models for the IPPS type of problems, for instance the one proposed by Kim, Park [4], the graph-based CP model is much simpler and easier to be implemented in industries. Later in this section, the way to project the graph representation to a CP-based IPPS model is elaborated.

3.2 Model formulation

The AND/OR graph in Fig. 2 has to be tailored beforehand. An alternative sub-sequence is called a branch in this paper, and all branches for the same set of features are called a group of peer branches. To clearly identify the start and end points of a group of peer branches, a pair of or-initiator and or-terminator are installed as shown in Fig. 3. Both or-initiator and or-terminator are dummy nodes. All dummy nodes do not consume manufacturing resources. Therefore, their processing times are equal to zero. Let manufacturing the handler and bolt be Job 1 and 2 respectively. Integer numbers beside nodes indicate the types of nodes, which will be elaborated in Section 3.2.3.

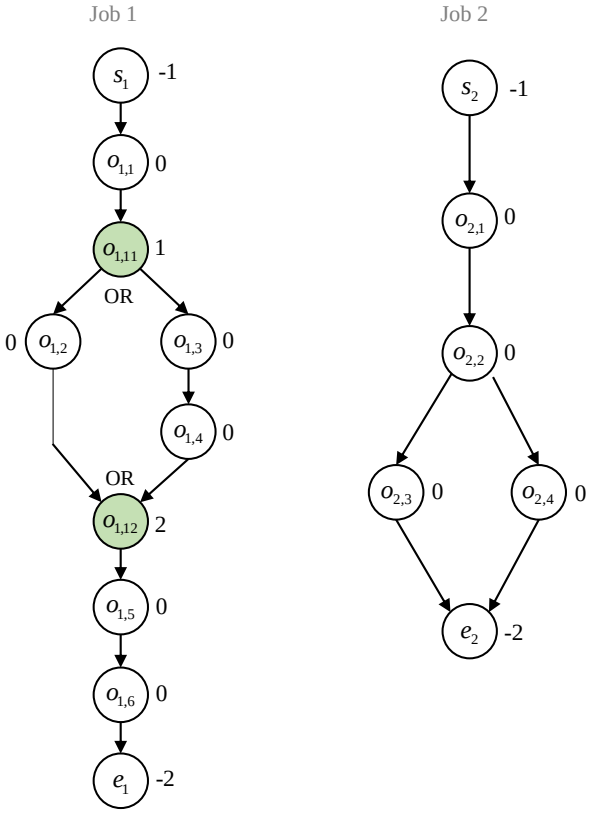

Fig. 3. Tailored AND/OR graph

The AND/OR graph $G=(O,A,L)$ for the IPPS problem comprises a set of nodes O representing operations, a set of arcs A representing sequence constraints, and a set of or-links L indicating the or-branches. A node is denoted by $o_{j,u}$ for which the subscript j indexes the job that the operation belongs to and u indexes the operation. An arc is denoted by $o_{j,u} \to o_{j,v}$, indicating that $o_{j,v}$ can only be started after finishing $o_{j,u}$. In this circumstance, $o_{j,u}$ is called an immediate predecessor of $o_{j,v}$ and $o_{j,v}$ is called an immediate successor of $o_{j,u}$. In almost all cases, the sequential relationship is only defined between two operations of the same job since jobs are usually independent from each other.

An or-link indicates the start points or endpoints of a group of alternative branches. Supposing that each branch has only one start node and one end node, an or-link can be represented by the set of start nodes or end nodes of the group of alternative branches. Taking the IPPS instance shown in Fig. 3 as an example, there are two or-links $OR\{o_{1,2},o_{1,3}\}$ and $OR\{o_{1,2},o_{1,4}\}$ which are the start-node set and end-node set of the two branches respectively. The prefix OR distinguishes the or-link from the common set. A hyperarc $o_{j,u_0} \to OR\{o_{j,u_1},o_{j,u_2},...,o_{j,u_n}\}$ can be defined where o_{j,u_0} is the node called or-initiator that initiates the group of or-branches. $o_{j,u_0} \to OR\{o_{j,u_1},o_{j,u_2},...,o_{j,u_n}\}$ can be regarded as a set of arcs $OR\{o_{j,u_0} \to o_{j,u_1}, o_{j,u_0} \to o_{j,u_2},...,o_{j,u_0} \to o_{j,u_n}\}$ for which the prefix OR indicates that the arcs in the set share an or-relation. Likewise, a hyperarc $OR\{o_{j,v_1},o_{j,v_2},...,o_{j,v_n}\} \to o_{j,v_0}$ can be defined where o_{j,v_0} is called an or-terminator that terminates the group of alternative or-branches. Similarly, the hyperarc $OR\{o_{j,v_1},o_{j,v_2},...,o_{j,v_n}\} \to o_{j,v_0}$ can be regarded as a set of arcs $OR\{o_{j,v_1} \to o_{j,v_0}, o_{j,v_2} \to o_{j,v_0},...,o_{j,v_n} \to o_{j,v_0}\}$. To make the AND/OR graph effective for the IPPS representation, follows assumptions must be made.

Assumptions on the AND/OR graph for IPPS problems

(1) The arcs that start from or go to a same node can have either and-relation or or-relation but not both.
(2) An or-branch has only one start node and one end node.
(3) The branches can be disconnected by removing the or-initiator and or-terminator.

Take the IPPS instance in Figure 3 as an example. Assumption (a) says that the node $o_{1,11}$ can

lead either the or-link $OR\{o_{1,2}, o_{1,3}\}$ or other regular nodes but not both. Assumption (b) says that each or-branch has only one start point, such as the node $o_{1,2}$ for the left branch and $o_{1,3}$ for the right branch. Assumption (c) forces that $o_{1,2}$ on the left branch cannot be connected to $o_{1,3}$ or $o_{1,4}$ on the right branch by any paths without passing the or-initiator $o_{1,11}$ or the or-terminator $o_{1,12}$.

Assumption (a) and (b) are to simplify the graph representation and modelling. For Assumption (a), it is always possible to isolate the and-relation and or-relation by adding a dummy or-initiator or or-terminator to enclose the group or-branches. For Assumption (b), it is always possible to add a single dummy start node or end node to an or-branch without numerically changing the IPPS instance. As a result, Assumption (a) and (b) are reasonably placed. Assumption (c) is to avoid logic errors for the construction of IPPS instance since operations on different peer or-branches are mutually exclusive. The three assumptions on the AND/OR graph for IPPS problems establish a one-on-one dependency between an or-initiator and an or-link (resp. an or-link and an or-terminator). Or say, there will be an or-link defined if and only if there is a preceding or-initiator or a succeeding or-terminator. In that case, it just needs to define the sets of or-initiators and or-terminals for the model. Table 1 summarizes problem-related elements and their notations.

Table 1. Elements and notations for the IPPS problem.

Notation	Description
j,k	Indexes for job and machine respectively.
u,v	Indexes for operations.
J	The set of all jobs for the IPPS instance $J=\{j \mid j=1,2,3,...,n\}$.
M	The set of all available machines $M=\{k \mid k=1,2,3,...,m\}$.
$o_{j,u}$	An operation of job j indexed by u.
$M(o_{j,u})$	The set of machines capable of processing $o_{j,u}$.
O	The set of all operations for the IPPS instance.
O_j	The set of all operations for job j, $O_j=\{o_{i,u} \in O \mid i=j\}$.
$(o_{j,u},k)$	A tuple to indicate processing operation $o_{j,u}$ by machine k.
P	The set of all defined tuples $(o_{j,u},k)$. $P=\{(o_{j,u},k) \mid o_{j,u} \in O, k \in M(o_{j,u})\}$.
$\tau(o_{j,u},k)$	The processing time of operation $o_{j,u}$ on machine k.

$o_{j,u} \rightarrow o_{j,v}$	An arc to define a sequence constraint between $o_{j,u}$ and $o_{j,v}$, indicating that $o_{j,u}$ is an immediate predecessor of $o_{j,v}$.
A	The set of all arcs.
s_j and e_j	A pair of dummy nodes indicating the start and end points of job j.
R^s, R^e	The set of or-initiators and the set of or-terminators respectively.

3.2.1 Decision variables

Generally speaking, the time interval is a piece of time during which the operation can be performed. Allocation of machining resource to an operation is to determine a time interval on a machine for an operation. In the IPPS problem domain, an interval can be optional due to the operation flexibility and processing flexibility. In the CP Optimizer, an interval I is a decision variable whose domain $dom(I)$ is

$$dom(I) \subseteq \{\perp\} \cup \{[s,e) \mid s,e \in \mathbb{Z}, s \le e\} \tag{1}$$

where $I = \perp$ means the interval is absent from the final schedule, s and e respectively represent the start and end of the time interval if the interval is present [41].

Specifically, there are two types of intervals designed for the IPPS model: $I(o_{j,u})$ the interval for processing operation $o_{j,u}$ with undetermined size, and $I(o_{j,u},k)$ the interval for processing $o_{j,u}$ on machine k with fixed size equal to $\tau(o_{j,u},k)$. The presence status of an interval I is captured by the binary variable

$$presenceOf(I) = \begin{cases} 0 & if\ I = \perp \\ 1 & otherwise \end{cases} \tag{2}$$

Specifically, the presence of time interval for operation $o_{j,u}$ is

$$presenceOf(I(o_{j,u})) = \begin{cases} 1 & \text{if } o_{j,u} \text{ is selected for job } j \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

By default, all intervals are optional.

3.2.2 The AND/OR graph-based CP model

The operations of the same job cannot be processed simultaneously, which means that intervals assigned to these operations cannot be overlapped. Let Q_j^J be the set of intervals for operations of job j, that is

$$Q_j^J = \{I(o_{j_0,u},k) \mid o_{j_0,u} \in O_j, k \in M(o_{j_0,u})\} \tag{4}$$

All time intervals in Q_j^J must be pairwise disjoint. Likewise, let Q_k^M be the set of intervals for the machine k, that is

$$Q_k^M = \left\{ I\left(o_{j,u}, k_0\right) \mid \left(o_{j,u}, k\right) \in P, k_0 = k \right\} \tag{5}$$

Variables and useful functions for the CP model are summarized in Table 2.

Table 2. Variables and functions for the CP model

Notation	Description
I	A time interval.
$start(I)$	A function to access the start point of time interval I.
$end(I)$	A function to access the endpoint of time interval I.
$size(I)$	A function to access the duration of time interval I.
$presenceOf(I)$	A variable to indicate the presence or absence of time interval I in the final schedule.
$I\left(o_{j,u}\right)$	A time interval variable for operation $o_{j,u}$. The start point, endpoint and size are undetermined.
$I\left(o_{j,u}, k\right)$	A time interval variable for the operation $\left(o_{j,u}, k\right)$. The size of $I\left(o_{j,u}, k\right)$ is equal to the processing time $\tau\left(o_{j,u}, k\right)$, and the start point and endpoint are undetermined.
Q_j^J	A set of time intervals for job j.
Q_k^M	A set of time intervals for machine k.
$c_{\max}$	The makespan.

Built-in constraints in CP Optimizer are borrowed to establish the model, including

- $nooverlap(C)$ with a set of intervals C as the input. It restricts that any pair of present intervals in C has no overlap.
- $endbeforestart\left(I_1, I_2\right)$ with two intervals I_1 and I_2 as the inputs. It restricts that the interval I_1 must end before I_2 starts in case that both I_1 and I_2 are present.
- $alternative(I, C)$ with an interval I and a set of intervals C as the inputs. The $alternative(\bullet)$ constraint synchronizes the interval I with exact one interval in C. The synchronization is threefold: the presence of interval I will force the presence of exact one

interval in C; if interval I is present, the start and end of I will be synchronized with the present interval in C; the absence of I will result in the absence of all intervals in C. It is to allocate machining recourse to each planning operation.

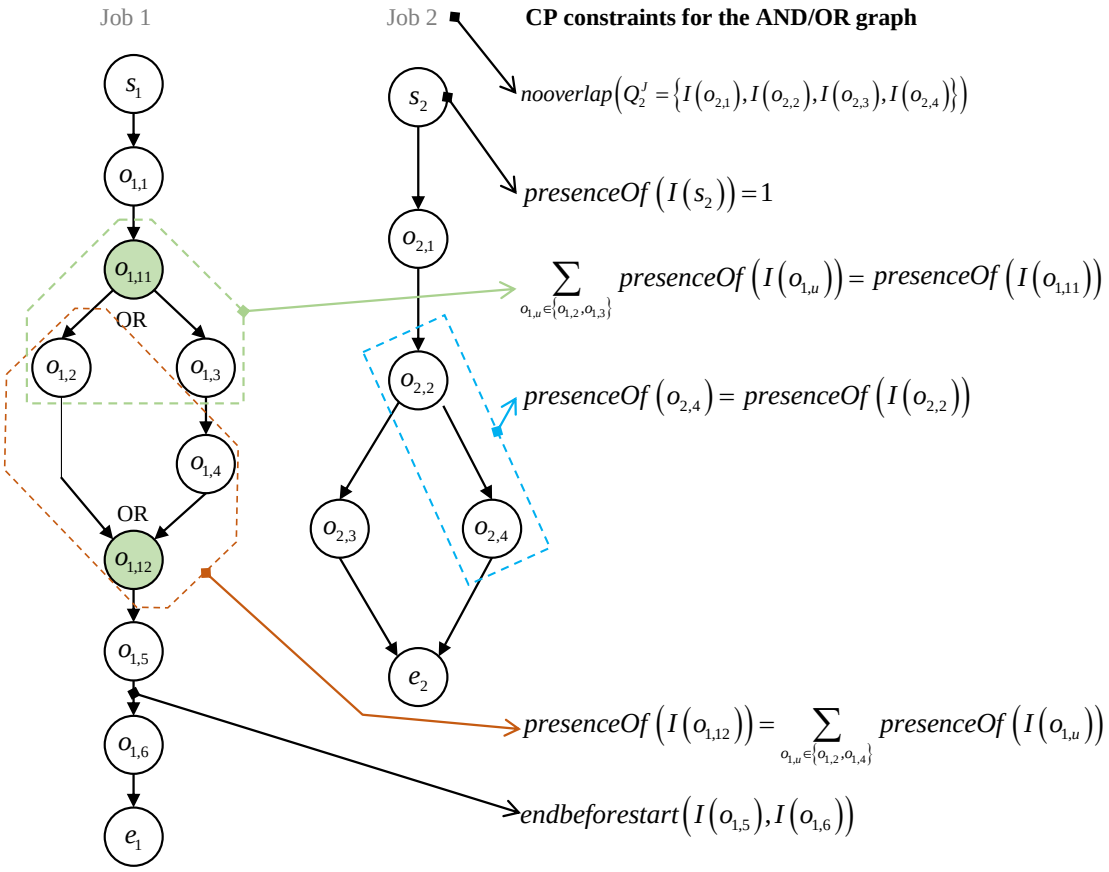

Fig. 3. Projecting the AND/OR graph structure to CP constraints.

Fig. 3 illustrates the projection from AND/OR graph structure to the CP constraints. Fig. 3 gives examples of CP constraints, which are explained below.

- $nooverlap\left(Q_2^J=\left\{I\left(o_{2,1}\right),I\left(o_{2,2}\right),I\left(o_{2,3}\right),I\left(o_{2,4}\right)\right\}\right)$. The intervals $I\left(o_{2,1}\right),I\left(o_{2,2}\right),I\left(o_{2,3}\right)$ and $I\left(o_{2,4}\right)$ for operations of job 2 cannot be overlapped.
- $presenceOf\left(I\left(s_2\right)\right)=1$. Job 2 must be started.
- $\sum_{o_{1,u}\in\left\{o_{1,2},o_{1,3}\right\}} presenceOf\left(I\left(o_{1,u}\right)\right)=presenceOf\left(I\left(o_{1,11}\right)\right)$ defines the presence dependency between the or-initiator $o_{1,11}$ and its succeeding or-link $OR\left\{o_{1,2},o_{1,3}\right\}$. The presence of $o_{1,11}$ implies the presence of its succeeding or-link.
- $presenceOf\left(o_{2,4}\right)=presenceOf\left(I\left(o_{2,2}\right)\right)$ deals with the presence dependency between two regular nodes both of which are neither or-initiator nor or-terminator. Connected by the arc

$o_{2,4} \to o_{2,2}$, the immediate successor $o_{2,2}$ must be present or absent together with its immediate predecessor $o_{2,4}$.

- $presenceOf\left(I\left(o_{1,12}\right)\right) = \sum_{o_{1,u} \in \{o_{1,2}, o_{1,4}\}} presenceOf\left(I\left(o_{1,u}\right)\right)$ defines the presence dependency between the or-link $OR\{o_{1,2}, o_{1,4}\}$ and its succeeding or-terminator. The presence of an or-terminator $o_{1,12}$ depends on the presence of its preceding or-link.
- $endbeforestart\left(I\left(o_{1,5}\right), I\left(o_{1,6}\right)\right)$ exerts the sequence constraint defined by the arc $o_{1,5} \to o_{1,6}$.

Putting in the constraints for machine capacity restriction and machine assignment for each operation, the CP model can be written as follows.

The Graph-Based CP Model for IPPS

$$\forall j \in J, \langle o_{j,u}, o_{j,v} \rangle \in A : endbeforestart\left(I\left(o_{j,u}\right), I\left(o_{j,v}\right)\right) \quad (6)$$

$$\forall j \in J : nooverlap\left(Q_j^J\right) \quad (7)$$

$$\forall v \in M : nooverlap\left(Q_v^M\right) \quad (8)$$

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

$$\forall o_{j,u_0} \in R^s : \sum_{o_{j,u} \in O, s.t. \langle o_{j,u0}, o_{j,u} \rangle \in A} presenceOf\left(I\left(o_{j,u}\right)\right) = presenceOf\left(I\left(o_{j,u_0}\right)\right) \quad (10)$$

$$\forall o_{j,v_0} \in R^e : presenceOf\left(I\left(o_{j,v_0}\right)\right) = \sum_{o_{j,v} \in O, s.t. \langle o_{j,v}, o_{j,v_0} \rangle \in A} presenceOf\left(I\left(o_{j,v}\right)\right) \quad (11)$$

$$\forall j \in J, \langle o_{j,u}, o_{j,v} \rangle \in A, o_{j,u} \notin R^s, o_{j,v} \notin R^e : presenceOf\left(I\left(o_{j,u}\right)\right) = presenceOf\left(I\left(o_{j,v}\right)\right) \quad (12)$$

$$\forall j \in J : presenceOf\left(I\left(s_j\right)\right) = 1 \quad (13)$$

$$Interval\ I\left(o_{j,u}\right)\ for\ \forall j \in J, \forall o_{j,u} \in O_j \quad (14)$$

$$Interval\ I\left(o_{j,u}, k\right)\ for\ \forall j \in J, \forall o_{j,u} \in O_j, \forall k \in M\left(o_{j,u}\right) \quad (15)$$

Constraint (6) is the precedence constraint between any pair of nodes connected by directed arcs. Constraint (7) and (8) reflect the no-overlapped assumptions for jobs and machines respectively. Constraint (9) forces that only one machine can be assigned to process an operation. Constraint (10) deals with the presence relationship between an or-initiator and the succeeding or-link. Likewise, constraint (11) takes care of the presence relationship between an or-terminator and its preceding or-link. Constraint (12) deals with the presence relationship between any pair of nodes connected by arcs and neither are or-initiator nor or-terminator. Constraint (13) points out that all

jobs have to be started. Constraint (14) and (15) define the two types of interval variables respectively.

Constraint (10) and (12) guarantee that once the selected process route goes into an or-branch, there will be no paths that lead the route to other alternative branches due to the assumption (3) on the AND/OR graph for IPPS problems number. Constraint (11) makes the process route continue after it leaves the or-branch. The constraints (10)-(12) guarantees that the graph-based CP model yields correct set of operations for all jobs.

3.2.3 A case and the model instantiation

This subsection proposes a way for the instantiation of the graph-based CP model based on type definition, together with the illustration of solving the IPPS instance given in Section 3.1. Suppose that there are 5 available machines, 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 data are given in Table 3. Corresponding graphic elements are given in Table 4. A dummy machine M0 is allocated for 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 of the IPPS example.

Operation $o_{j,u}$	Capable machines $M(o_{j,u})$	Processing time $\tau(o_{j,u},k)$	Operation $o_{j,u}$	Capable machines $M(o_{j,u})$	Processing time $\tau(o_{j,u},k)$
$o_{1,1}$	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. Graphic elements for the IPPS example.

Set	Data
J	$J=\{1,2\}$
M	$M=\{0,1,2,3,4,5\}$
O	$\{o_{1,1},o_{1,2},o_{1,3},o_{1,4},o_{1,5},o_{1,6},o_{2,1},o_{2,2},o_{2,3},o_{2,4},s_1,s_2,e_1,e_2,o_{1,11},o_{1,12}\}$
$M(o_{j,k})$	$M(o_{1,1})=\{3,4\}$, $M(o_{1,2})=\{4\}$, $M(o_{1,3})=\{3\}$, $M(o_{1,4})=\{1,2,5\}$, etc.

O_j	$O_1=\{o_{1,1},o_{1,2},o_{1,3},o_{1,4},o_{1,5},o_{1,6},o_{1,11},o_{1,12},s_1,e_1\}$, $O_2=\{o_{2,1},o_{2,2},o_{2,3},o_{2,4},s_2,e_2\}$
P	$P=\{(o_{1,1},3),(o_{1,1},4),(o_{1,2},4),(o_{1,3},3),(o_{1,4},1),...\}$
A	$A=\left\{\begin{array}{l} s_1\to o_{1,1},o_{1,1}\to o_{1,11},o_{1,11}\to o_{1,2},o_{1,11}\to o_{1,3},o_{1,2}\to o_{1,12},o_{1,3}\to o_{1,4},\\ o_{1,4}\to o_{1,12},o_{1,12}\to o_{1,5},o_{1,5}\to o_{1,6},o_{1,6}\to e_1,s_2\to o_{2,1},o_{2,1}\to o_{2,2},\\ o_{2,2}\to o_{2,3},o_{2,2}\to o_{2,4},o_{2,3}\to e_2,o_{2,4}\to e_2 \end{array}\right\}$
R^s & R^e	$R^s=\{o_{1,11}\}$, $R^e=\{o_{1,12}\}$.

There are 5 types of nodes, including the start nodes $\{s_j \mid j\in j\}$, end nodes $\{s_j \mid j\in j\}$, or-initiators R^s, or-terminators R^e, and regular nodes for operations. Assigning type numbers to distinguish the nodes' categories can simplify the formulation. Letting o_i be any node, type numbers can be assigned as follows.

$$T(o_i)=\begin{cases} -1 \ \textit{if } o_i \textit{ is a start node, that is } o_i\in\{s_j \mid j\in J\} \\ -2 \ \textit{if } o_i \textit{ is an end node, that is } o_i\in\{e_j \mid j\in J\} \\ 1 \quad \textit{if } o_i \textit{ is an or}-\textit{start node, that is } o_i\in R^s \\ 2 \quad \textit{if } o_i \textit{ is an or}-\textit{end node, that is } o_i\in R^e \\ 0 \quad \textit{otherwise} \end{cases} \tag{16}$$

Based on the Type definition, the graph-based CP model for IPPS problem can be rewritten as follows.

The Applied Graph-Based CP Model for IPPS

$$\forall\langle o_1,o_2\rangle\in A: endbeforestart(I(o_1),I(o_2)) \tag{17}$$

Constraint (7) and (8)

$$\forall o_i\in O: alternative(I(o_i),\{I(o_i,k)\mid k\in M(o_i)\}) \tag{18}$$

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

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

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

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

$$Interval\ I(o_i),\ Interval\ I(o_i,k)\ for\ \forall o_i\in O, k\in M(o_i) \tag{23}$$

The constraints listed in the Applied Graph-Based CP Model correspond to the constraints (6)–(13). The constraints (17) and (18)–(22) work on the same set of operations O. It enhances the consistency in the programming work and makes the proposed model easier to be implemented.

Taking minimizing the makespan as the single objective, that is

$$\min \max\left\{end\left(I\left(o_i\right)\right) \mid o_i \in O, T\left(o_i\right)=-2\right\} \tag{24}$$

the model is programmed with IBM ILOG CPLEX Optimization Studio and solved automatically by CP Optimizer. An optimal schedule generated in a run is sketched in Fig. 4, where all dummy operations are neglected. It is easy to verify that all 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.

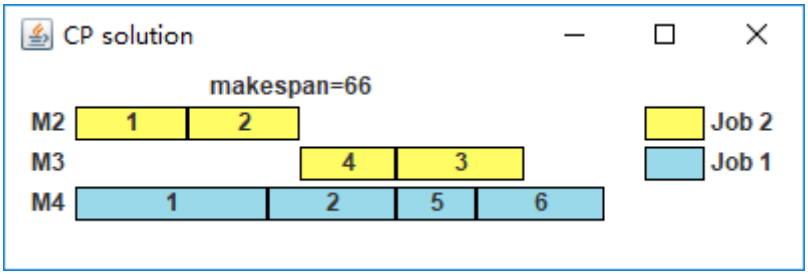

Fig. 4. An optimal schedule for the IPPS example.

4. Experiments

A testbed proposed by Kim, et al. is adopted for the experiments [42]. Purposely-designed 18 jobs consisting of around 300 operations are combined in different ways to generate 24 problems of different levels of flexibility. The problems' specification of the testbed is given in Table 5. The lower bound is the maximum of the smallest possible total processing time of each job. Let O_j^* be a set of operations applicable of fulfilling job j. A simple lower bound is calculated by

$$\max_{j \in J}\left\{\min_{all\ possible\ O_j^*}\left\{\sum_{o_{j,u} \in O_j^*} \min_{k \in M\left(o_{j,u}\right)} \tau\left(o_{j,u}, k\right)\right\}\right\} \tag{25}$$

The experiments are conducted on the Lenovo ThinkStation P720 equipped with 12-core Xeon® Silver 4116 CPU @2.1GHz. IBM ILOG CPLEX Optimization Studio version 12.10 is used to formulate the model. The CP Optimizer uses the newest iterative diving search method to perform aggressive dives in the search tree without backtrack [4]. Other settings are default.

Table 5. Specification of the benchmark problems

Problem	Job number	Number of jobs	Lower bound
1	1, 2, 3, 10, 11, 12	6	427
2	4, 5, 6, 13, 14, 15	6	343
3	7, 8, 9, 16, 17, 18	6	344
4	1, 4, 7, 10, 13, 16	6	306
5	2, 5, 8, 11, 14, 17	6	304
6	3, 6, 9, 12, 15, 18	6	427
7	1, 4, 8, 12, 15, 17	6	372
8	2, 6, 7, 10, 14, 18	6	342

9	3, 5, 9, 11, 13, 16	6	427
10	1, 2, 3, 5, 6, 10, 11, 12, 15	9	427
11	4, 7, 8, 9, 13, 14, 16, 17, 18	9	344
12	1, 4, 5, 7, 8, 10, 13, 14, 16	9	306
13	2, 3, 6, 9, 11, 12, 15, 17, 18	9	427
14	1, 2, 4, 7, 8, 12, 15, 17, 18	9	372
15	3, 5, 6, 9, 10, 11, 13, 14, 16	9	427
16	1, 2, 3, 4, 5, 6, 10, 11, 12, 13, 14, 15	12	427
17	4, 5, 6, 7, 8, 9, 13, 14, 15, 16, 17, 18	12	344
18	1, 2, 4, 5, 7, 8, 10, 11, 13, 14, 16, 17	12	306
19	2, 3, 5, 6, 8, 9, 11, 12, 14, 15, 17, 18	12	427
20	1, 2, 4, 6, 7, 8, 10, 12, 14, 15, 17, 18	12	372
21	2, 3, 5, 6, 7, 9, 10, 11, 13, 14, 16, 18	12	427
22	2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18	15	427
23	1, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18	15	372
24	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18	18	427

4.1 Experiment 1: global performance test

The *FailLimit* is a parameter built in IBM ILOG CP Optimizer that limit the number of failures allowed before terminating a search. Setting the *FailLimit* number usually depends on the problem scale. In this experiment, the *FailLimit* is set to be 500,000 for all problems to test the performance of the model with reasonable computing resources. The proposed model is compared to the symbiotic evolutionary algorithm (SEA) [13], the Improved Genetic Algorithm (IGA) [4], the Object-Coding Genetic Algorithm (OCGA) [9], and the Enhanced Ant Colony Optimization (E-ACO) [10].

Table 6. Makespans found by SEA, IGA, OCGA and graph-based CP (GCP).

Problem	Lower bound	SEA	IGA	OCGA	E-ACO	GCP	Best	Improved rate (%)	Runtime of GCP (s) †
1	427	437.6	**427***	**427***	427.1	**427***	–	–	0.62
2	343	349.7	344.5	343.5	343.1	**343***	GCP	0.03	3.2
3	344	355.2	351	346.4	345	**344***	GCP	0.29	2.5
4	306	306.2	307.4	310.1	307.6	**306***	GCP	0.07	3.43
5	304	323.7	**309.8**	323	319.6	318	IGA	-2.58	2.78
6	427	443.8	**427***	**427***	427.1	**427***	–	–	0.89
7	372	372.4	372.7	373.3	**372***	**372***	–	–	2.07
8	342	348.3	357	343.5	343.3	**343**	GCP	0.09	3.43
9	427	434.9	**427***	**427***	427.1	**427***	–	–	0.86
10	427	456.5	431.6	427.1	427.6	**427***	GCP	0.02	0.64
11	344	378.9	379.7	350.6	350.2	**344***	GCP	1.80	4.47
12	306	332.8	323.7	324.7	323.4	**318**	GCP	1.70	4.61
13	427	469	442.8	427.2	427.6	**427***	GCP	0.05	0.96

14	372	402.4	415.3	377.4	374.3	**372***	GCP	0.62	3.01
15	427	445.2	427.4	**427***	427.3	**427***	–	–	1.07
16	427	478.8	449.4	428.1	430.9	**427***	GCP	0.26	1.36
17	344	448.9	426	383.1	381.2	**344***	GCP	10.81	7.39
18	306	389.6	373.6	354.5	361.5	**344**	GCP	3.05	6.05
19	427	508.1	471.3	433.7	434.9	**427***	GCP	1.57	2.03
20	372	453.8	446.6	390.5	392.4	**372***	GCP	4.97	4.64
21	427	483.2	447.8	**427***	429.4	**427***	–	–	1.5
22	427	548.3	508.1	452.9	447.2	**427***	GCP	4.73	3.98
23	372	507.5	477.8	410	420.3	**378**	GCP	8.47	6.51
24	427	602.2	548.5	471.2	479.3	**436**	GCP	8.07	10.33

–: A dash implies that both one compared algorithm and the GCP obtain the optimal solution.

*: Optimal solution.

†: 's' is for seconds.

Table 5 reports makespans found by different approaches for the 24 problems. The first column is the problem number. Column two gives the lower bounds of each problem. Column 3 to 7 respectively list the mean makespans found by each algorithm in several runs. Best solutions are highlighted in bold. The improvement rate is the percentage of the makespan improved by GCP to the best solution found by compared algorithms. The runtimes of GCP for solving each problem in the given experimental environment is listed in the last column. The GCP outperforms the compared algorithms in solving 17 problems. For the other 6 problems, both the GCP and one compared algorithm reach the optimal solutions. In total, the graph-based CP approach finds optimal solutions for 18 problems. For simpler problems number 1 to 16, the improvement is not significant since lower bounds are already or nearly reached by the OCGA. For more complex problems number 17 to 24, the solutions' quality 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 experimental environment.

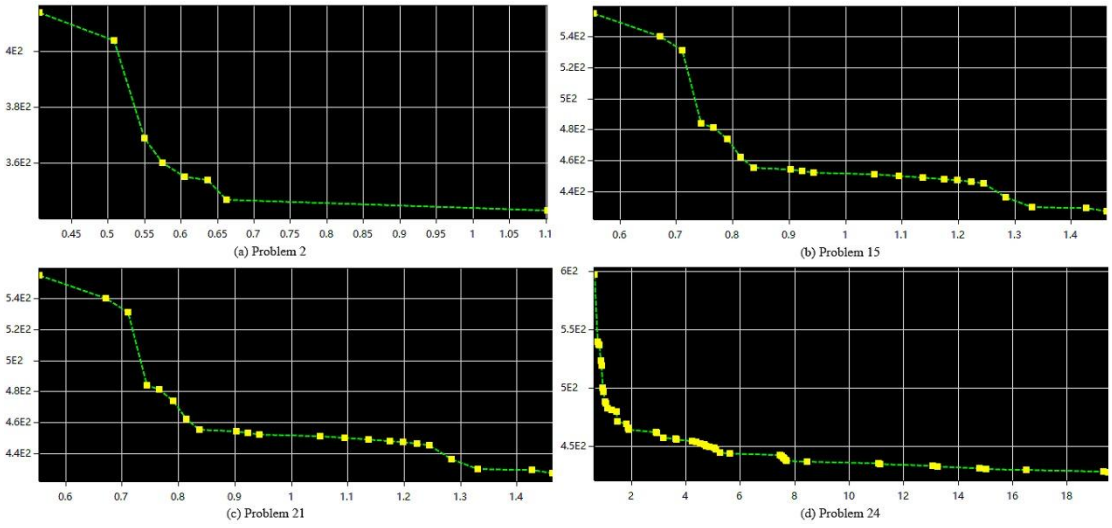

Fig. 5. The convergence procedure for solving problem 2, 15, 21 and 24

The search procedures for solving representative problems number 2, 15, 21 and 24 are depicted in Fig. 5, for which the horizontal axis represents the timeline and the vertical axis represents the makespan. The problem-scale extends from problem 2 to 24. It shows that the search dives quickly at the early stage to near optimal solutions for all four problems. For more complex problems 15, 21 and 24, the algorithm can continuously explore better solutions.

4.2 Experiment 2: extreme test

In this experiment, the graph-based CP model is allowed to run enough time to get solutions as good as possible. Best makespans found by the 5 algorithms are listed in Table 7.

Table 7. Comparison of best solutions

Problem	Lower bound	SEA	IGA	OCGA	E-ACO	GCP	Best	Improved rate (%)	Runtime of GCP (s)
1	427	428	**427***	**427***	**427***	**427***	–	–	0.25
2	343	**343***	**343***	**343***	**343***	**343***	–	–	>1000†
3	344	347	**344***	**344***	**344***	**344***	–	–	>1000
4	306	**306***	**306***	**306***	**306***	**306***	–	–	>1000
5	304	319	**304***	318	318	318	IGA	-4.40	>1000
6	427	438	**427***	**427***	**427***	**427***	–	–	0.96
7	372	**372***	**372***	**372***	**372***	**372***	–	–	>1000
8	342	343	**342***	343	343	343	IGA	-0.29	>1000
9	427	428	**427***	**427***	**427***	**427***	–	–	0.8
10	427	443	**427***	**427***	**427***	**427***	–	–	0.56
11	344	369	368	348	348	**345**	GCP	0.87	>1000
12	306	328	**312**	318	322	318	IGA	-1.89	>1000
13	427	452	429	**427***	**427***	**427***	–	–	0.95
14	372	381	386	**372***	373	**372***	–	–	>1000
15	427	434	**427***	**427***	**427***	**427***	–	–	1.45
16	427	454	433	**427***	429	**427***	–	–	1.69
17	344	431	415	370	377	**344***	GCP	7.56	>1000
18	306	379	364	351	357	**344**	GCP	2.03	>1000
19	427	490	450	**427***	431	**427***	–	–	1.82
20	372	447	429	384	386	**372***	GCP	3.23	>1000
21	427	477	433	**427***	428	**427***	–	–	1.53
22	427	534	491	446	444	**427***	GCP	3.98	3.81
23	372	498	465	394	413	**372***	GCP	5.91	>1000
24	427	587	532	458	460	**427***	GCP	7.26	22.93

*: Optimal solution.

†: The runtime is longer than 1000 seconds.

Compared to the results in the global performance test, the GCP slightly improves the makespans for problem 5, 12, 17, 18, 23 and 24. Besides, the GCP finds optimal solutions for three more

problems number 17, 23, and 24. It can be seen from Table 7 that GCP performs better than other 4 algorithms in solving more complex problems. It finds optimal solutions for the most complex problems 22 to 24 in reasonable times. One optimal schedule for problem number 24 is shown in Fig. 6.

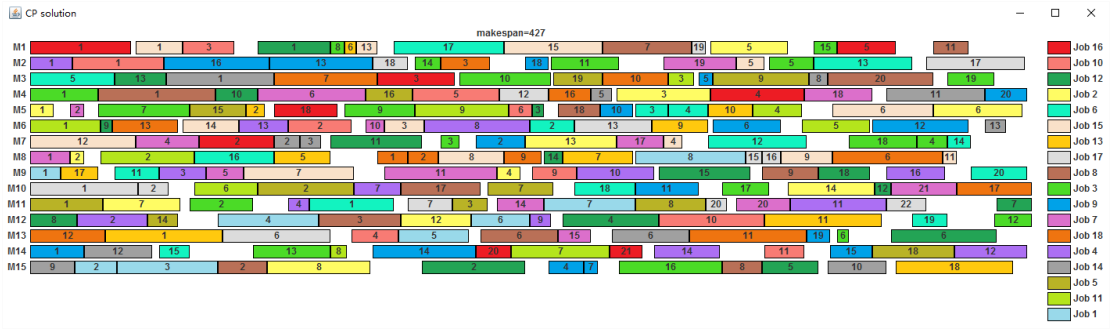

Fig. 6. An optimal schedule for problem 24 with makespan=427

5. Conclusions and discussions

This paper proposes a graph-based constraint programming model for the IPPS problem in the jobshop-type manufacturing environment. The IPPS problem takes AND/OR graphs as the input. Three types of flexibility namely processing flexibility, operation flexibility, and sequencing flexibility are considered.

Based on the interval variable, sequence structure, and scheduling-oriented constraints built in IBM ILOG CP Optimizer, a concise model is formulated for the problem, together with a simplified version based on type-definition for nodes. The experiments on a benchmark testbed show that the CP Optimizer can solve the graph-based CP model efficiently with built-in search algorithms.

It is promising to implement this model in industries based on the model-and-run CP optimizer. It is also promising to further develop the model to cater to the process planning and scheduling problems in other circumstances. For instance: the assumption on internal logistic and setup consumption can be relaxed to make the model more practical; the non-preemption assumption can be removed to build a more general model, for which the *cumulFunction* built in CP Optimizer can be used to restrict the resource usage. The problem relaxation and development of corresponding solution algorithms can be considered in future work. Besides, establishing agent-based framework for the rescheduling to cope with dynamics in manufacturing environment based on current AND/OR graph representation and the CP model is also important for both academic research and real-world implementation.

Acknowledgment

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

Reference

[1] Leung CW, Wong T, Mak K-L, Fung RY. Integrated process planning and scheduling by an agent-based ant colony optimization. Computers & Industrial Engineering. 2010;59(1):166-80.

[2] CHEN Q, KHOSHNEVIS B. Scheduling with flexible process plans. Production Planning & Control.

1993;4(4):333-43.
[3] Kempenaers J, Pinte J, Detand J, Kruth J-P. A collaborative process planning and scheduling system. Advances in Engineering Software. 1996;25(1):3-8.
[4] Kim YK, Park K, Ko J. A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling. Computers & Operations Research. 2003;30(8):1151-71.
[5] Barzanji R, Naderi B, Begen MA. Decomposition algorithms for the integrated process planning and scheduling problem. Omega. 2020;93:102025.
[6] Sotskov YN, Shakhlevich NV. NP-hardness of shop-scheduling problems with three jobs. Discrete Applied Mathematics. 1995;59(3):237-66.
[7] Wong T, Leung C, Mak K, Fung R. An agent-based negotiation approach to integrate process planning and scheduling. International journal of production research. 2006;44(7):1331-51.
[8] MORAD N, ZALZALA A. Genetic algorithms in integrated process planning and scheduling. Journal of Intelligent Manufacturing. 1999;10(2):169-79.
[9] Lihong Q, Shengping L. An improved genetic algorithm for integrated process planning and scheduling. The International Journal of Advanced Manufacturing Technology. 2012;58(5-8):727-40.
[10] Zhang L, Wong TN. An object-coding genetic algorithm for integrated process planning and scheduling. European Journal of Operational Research. 2015;244(2):434-44.
[11] Jin L, Zhang C, Shao X, Tian G. Mathematical modeling and a memetic algorithm for the integration of process planning and scheduling considering uncertain processing times. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture. 2016;230(7):1272-83.
[12] Rossi F, Van Beek P, Walsh T. Handbook of constraint programming: Elsevier; 2006.
[13] Laborie P, Rogerie J, Shaw P, Vilím P. IBM ILOG CP optimizer for scheduling. Constraints. 2018;23(2):210-50.
[14] Baptiste P, Le Pape C, Nuijten W. Constraint-based scheduling: applying constraint programming to scheduling problems: Springer Science & Business Media; 2012.
[15] Crowston WB. Decision CPM: Network Reduction and Solution. Journal of the Operational Research Society. 1970;21(4):435-52.
[16] Mello LSHd, Sanderson AC. AND/OR graph representation of assembly plans. IEEE Transactions on Robotics and Automation. 1990;6(2):188-99.
[17] Gillies DW, Liu JW-S. Scheduling tasks with AND/OR precedence constraints. SIAM Journal on Computing. 1995;24(4):797-810.
[18] Beck JC, Fox MS. Scheduling alternative activities. AAAI/IAAI: Citeseer; 1999. p. 680-7.
[19] Barták R, Cepek O. Temporal Networks with Alternatives: Complexity and Model. FLAIRS Conference2007. p. 641-6.
[20] Moffitt MD, Peintner B, Pollack ME. Augmenting disjunctive temporal problems with finite-domain constraints. Proceedings of the National Conference on Artificial Intelligence: Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999; 2005. p. 1187.
[21]
[22] Li X, Gao L, Shao X, Zhang C, Wang C. Mathematical modeling and evolutionary algorithm-based approach for integrated process planning and scheduling. Computers & Operations Research. 2010;37(4):656-67.
[23] Tao S, Dong ZS. Multi-mode resource-constrained project scheduling problem with alternative project structures. Computers & Industrial Engineering. 2018;125:333-47.

[24] Lee H, Kim S-S. Integration of process planning and scheduling using simulation based genetic algorithms. The International Journal of Advanced Manufacturing Technology. 2001;18(8):586-90.
[25] Shao X, Li X, Gao L, Zhang C. Integration of process planning and scheduling—a modified genetic algorithm-based approach. Computers & Operations Research. 2009;36(6):2082-96.
[26] Li X, Gao L, Pan Q, Wan L, Chao K-M. An effective hybrid genetic algorithm and variable neighborhood search for integrated process planning and scheduling in a packaging machine workshop. IEEE Transactions on Systems, Man, and Cybernetics: Systems. 2018;49(10):1933-45.
[27] Luo G, Wen X, Li H, Ming W, Xie G. An effective multi-objective genetic algorithm based on immune principle and external archive for multi-objective integrated process planning and scheduling. The International Journal of Advanced Manufacturing Technology. 2017;91(9-12):3145-58.
[28] Li W, McMahon CA. A simulated annealing-based optimization approach for integrated process planning and scheduling. International Journal of Computer Integrated Manufacturing. 2007;20(1):80-95.
[29] Guo Y, Li WD, Mileham AR, Owen GW. Applications of particle swarm optimisation in integrated process planning and scheduling. Robotics and Computer-Integrated Manufacturing. 2009;25(2):280-8.
[30] Wong TN, Leung CW, Mak KL, Fung RYK. Dynamic shopfloor scheduling in multi-agent manufacturing systems. Expert Systems with Applications. 2006;31(3):486-94.
[31] Wong T, Leung C, Mak K, Fung R. Integrated process planning and scheduling/rescheduling—an agent-based approach. International Journal of Production Research. 2006;44(18-19):3627-55.
[32] Harjunkoski I, Grossmann IE. Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. Computers & Chemical Engineering. 2002;26(11):1533-52.
[33] Sadykov R, Wolsey LA. Integer programming and constraint programming in solving a multimachine assignment scheduling problem with deadlines and release dates. INFORMS Journal on Computing. 2006;18(2):209-17.
[34] Beck JC, Feng TK, Watson JP. Combining Constraint Programming and Local Search for Job-Shop Scheduling. Informs Journal on Computing. 2011;23(1):1-14.
[35] Bartak R, Salido MA, Rossi F. Constraint satisfaction techniques in planning and scheduling. Journal of Intelligent Manufacturing. 2010;21(1):5-15.
[36] Timpe C. Solving planning and scheduling problems with combined integer and constraint programming. Or Spectrum. 2002;24(4):431-48.
[37] Ham AM, Cakici E. Flexible job shop scheduling problem with parallel batch processing machines: MIP and CP approaches. Computers & Industrial Engineering. 2016;102:160-5.
[38] Zhang S, Wang S. Flexible assembly job-shop scheduling with sequence-dependent setup times and part sharing in a dynamic environment: Constraint programming model, mixed-integer programming model, and dispatching rules. IEEE Transactions on Engineering Management. 2018;65(3):487-504.
[39] Laborie P. An update on the comparison of MIP, CP and hybrid approaches for mixed resource allocation and scheduling. International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research: Springer; 2018. p. 403-11.
[40] Kreter S, Schutt A, Stuckey PJ. Using constraint programming for solving RCPSP/max-cal. Constraints. 2017;22(3):432-62.
[41] Qiao L, Lv S. An improved genetic algorithm for integrated process planning and scheduling. Int J

Adv Manuf Technol. 2012;58(5-8):727-40.
[42] Laborie P, Rogerie J. Reasoning with Conditional Time-Intervals. FLAIRS conference2008. p. 555-60.