LETTER *Special Section on Mathematical Systems Science and Applications*

Automatic Generation of Mixed Integer Programming for Scheduling Problems Based on Colored Timed Petri Nets

Andrea Veronica PORCO[†], **Ryosuke USHIJIMA**[†], ***Nonmembers***, ***and*** **Morikazu NAKAMURA**[††a)], ***Member***

SUMMARY This paper proposes a scheme for automatic generation of mixed-integer programming problems for scheduling with multiple resources based on colored timed Petri nets. Our method reads Petri net data modeled by users, extracts the precedence and conflict relations among transitions, information on the available resources, and finally generates a mixed integer linear programming for exactly solving the target scheduling problem. The mathematical programing problems generated by our tool can be easily inputted to well-known optimizers. The results of this research can extend the usability of optimizers since our tool requires just simple rules of Petri nets but not deep mathematical knowledge.

key words: *scheduling problem, mixed integer programming, Petri nets, colored timed Petri net, automatic generation*

1. Introduction

Petri nets are a well-known mathematical modeling language for concurrent systems, where the concurrent systems include much variety of systems such as parallel/distributed systems, network systems, production systems, collaborative robots, and many others [1]. Petri nets are mathematically powerful for analysis of modeled systems and are also a graphically understandable for the system's structure and behavior. Once we know a limited number of simple rules on Petri nets, we can start system modeling.

Scheduling problems are important research topics in operations research and computer science, where many researchers investigate algorithms to solve exactly or approximately scheduling problems with considering their NP-hardness of them [2]–[5]. Scheduling problems are also valuable in practice since the problems are applicable in a broad range of fields [6], [7].

Recent advancement of optimization algorithms makes us solve exactly scheduling problems of practical size even if the problem is NP-hard. There are very efficient commercial optimization tools, such as CPLEX [8] and Gurobi Optimizer [9] and also freeware tools. However, limited users get benefits from this optimization approach. The reason for limited usage is not only an economic reason but also usability. Users need to formulate their problems firstly as mathematical programming problems, which requires deep knowledge of mathematics. Therefore, an only limited quantity of users can utilize these tools.

In this paper, we present automatic generation of mixed-integer programming for scheduling problems of multiple resources by making use of Petri nets. Users just need to model their target system with Petri nets and set necessary information for operations. Our proposed tool generates the mathematical programming problem for scheduling of the system, and then we utilize some optimization tool to solve the generated problem.

There are many Petri Net based scheduling studies, such as [10]–[12]. However, none of them treated automatic generation of mathematical programming for scheduling problems.

2. Preliminaries

A Petri net is a 4 tuple $PN = (P, T, Pre, Post)$ where $P = \{p_1, p_2, ..., p_n\}$ and $T = \{t_1, t_2, ..., t_m\}$ are a set of places and a set of transitions, respectively. *Pre(p, t)* and *Post(p, t)* express the weight on the arc from place p to transition t and from transition t to place p, respectively.

A marking $M^{tr} = (M(p_1), M(p_2), ..., M(p_n))$ represents a token distribution on places, that is, $M(p_i)$ is the number of tokens in place p_i. Here tr shows the transpose of the matrix. Token distributions show states of the system. Therefore, the initial marking M_0 shows the initial state of the corresponding system. We call p an input place of t when $Pre(p, t) > 0$ and an output place when $Post(p, t) > 0$. Transition t is enabled under some marking M_i when $M_i(p) \geq Pre(p, t), \forall p \in \bullet t$ and transition t can be fired when it is enabled, where $\bullet t$ shows the set of all the input places of t. On t's firing, $Pre(p, t)$ of tokens in each input place p should be removed and $Post(p, t)$ of tokens are added to each output place p. A transition corresponds to an event, and its firing represents an occurrence of the event in the system. The dynamical behavior of a system can be represented by changing of token distribution by firing in the Petri net model.

For quantitative analysis of a dynamical behavior of a system, many researchers introduced *time* to Petri nets. We can categorize the ways of timing into three types, FD (Firing Duration), HD (Holding Duration), and ED (Enabling Duration). The FD is to assign time to transitions, where the firing of transition takes time. The HD is referred as place time Petri nets, where tokens cannot be used for firing for a particular period after located in the place. The last one,

Manuscript received April 30, 2017.
Manuscript revised August 29, 2017.

[†]The authors are with Graduate Course of Science and Engineering, University of the Ryukyus, Okinawa-ken, 903-0213 Japan.
[††]The author is with Computer Science and Intelligent Systems, Dept. of Eng., University of the Ryukyus, Okinawa-ken, 903-0213 Japan.
a) E-mail: morikazu@ie.u-ryukyu.ac.jp (Corresponding author)

DOI: 10.1587/transfun.E101.A.367

the ED, is such that a transition cannot be fired for a given period after enabled [10]–[12]. In this paper, we consider timed Petri nets with FD because of its intuitively easiness.

Timed Petri nets are a six-tuple $TPN = (P, T, Pre, Post, TS, D)$, where TS is a set of time stamps. Usually we use the set of positive real numbers, and $D : T \rightarrow TS$ is a function to show the duration time of transition $t \in T$. A time stamp is attached to a token when the token is generated. In the timed Petri net, transition t is enabled at time τ when each input place of t has more than or equal to $Pre(p, t)$ tokens and its time stamp is no more than τ. By firing of t at time τ, the token distribution should be changed according to the same rule of the Petri net described above except that we attach the time stamp $\tau + D[t]$ to each output token.

Moreover, in colored Petri net, another extended Petri net, each token has values called *color*. On firing, the values of the produced tokens for the output places are calculated based on values of tokens of the input places. Colors of tokens in the input places can be preconditions for firing. Therefore, colored Petri nets have very strong modeling power. More details are explained in the literature [1], [13].

The scheduling problem is to determine the starting time of each task to optimize the given objective function by ordering tasks which use the same resource while satisfying all the precedence relations.

A scheduling problem can be seen as a 6-tuple $SP = (TASK, RS, RR, PRE, RT, PT)$, where $TASK$ is a set of tasks, RS is a set of resources, $RR : TASK \rightarrow 2^{|RS|}$ is a function which maps a task to an available resource set, $PRE \subseteq TASK \times TASK$ is the precedence relation between two tasks, $RT : TASK \rightarrow TS$ is a function to show the release time of a task, $PT : TASK \times RS \rightarrow TS$ is a function to return the processing time of tasks when we assign an available resource, where TS is the time length, usually the natural number set or the non-negative real number set.

3. Timed Petri Net Model for Scheduling Problems

3.1 Assumptions for Scheduling Problems

In this paper, we treat scheduling problems under the following assumptions. These are an extension from our previous work [14] since we can now allow multiple resources for each resource type. Therefore, this paper can cover multi-processor scheduling problems, multi-machine job-shop scheduling.

1. No resource can process more than one task at a time.
2. Multiple resources may be available for each type of resources, that is, they have the same functionality but may have difference capabilities.
3. Each task can be processed by a single resource.
4. Each resource is always available for processing, that is, *no breakdown*.
5. Operations can not be interrupted until their completion, that is, *no preemption*.
6. The processing times are known in advance and they are deterministic.

For the scheduling problem, we verified the feasibility of the problem [5], [10].

Proposition 1: A schedule is feasible if and only if the following conditions are satisfied:

1. All the precedence relations are satisfied.
2. The release time conditions are satisfied.
3. There exist no resource conflicts.

3.2 Modeling

In our approach, we model at first precedence relation between tasks with Petri net and then add resource information.

Petri net models for the precedence relation can be easily constructed from $TASK$ and PRE in a given scheduling problem $SP = (TASK, RS, RR, PRE, RT, PT)$. The net can be a sound workflow net when we add a single source p_i and a single sink p_o[15]. Figure 1 shows an example in which the subnet drawn with black ink represents a sound workflow net.

The soundness ensures the followings:

1. Only the single source place includes a token at the initial state and only the single sink place has a token at the final state.
2. All the state reachable from the initial state can lead to the final state.

We call the Petri net model with a single source and a single sink as *process net*.

Additionally, we overlay the resource net obtained from $TASK$, RS and RR. Note that we can decompose the resources set RS into subsets RS_i such that $RS = \cup_i RS_i$ by their functionality.

For each subset RS_i we introduce a place rp_i, thus, we have $RP = \{rp_i | i = 1, 2, ..., r\}$, where r means the number of resource types. From the point of the colored Petri net, it means that we assign a color to each place in RP by a color function C:

$$C : RP \rightarrow ResourceType \quad (1)$$

$$ResourceType = \{RS_1, RS_1, ...RS_r\} \quad (2)$$

For each place $rp_i \in RP$, we locate initial tokens as follows:

$$M_0(p) = \begin{cases} \{UNIT\} & (p \text{ is source}) \\ \{rt_{i,1}, rt_{i,2}, ..., rt_{i,r_i}\} & (p = rp_i) \\ \emptyset & (\text{otherwise}) \end{cases} \quad (3)$$

where $UNIT$ shows a token without color, that is, a normal token, and r_i means the number of resources of the resource type i.

By referring to the resource requirement of each task, RR, we can connect from each transition $t \in T$ to rp_i, and vise versa. We denote the set of arcs added here by $\hat{Pre}$ and

$\hat{Post}$ to differentiate from *Pre* and *Post* in the process net.

Moreover, let each token have a color representing its capability to calculate the duration time when the corresponding resource is assigned to a transition:

$$D : T_i \times RS_i \rightarrow TS \tag{4}$$

where T_i is the subset of T such that $RR(t) = RS_i, \forall t \in T_i$. The timestamp of all the tokens produced by firing can be calculated by adding this duration time to the starting time of the firing.

Finally, we obtain a colored timed Petri net for the scheduling problem sp, $CTPN = (P \times RP, T, Pre \cup \hat{Pre}, Post \cup \hat{Post}, TS, D, C)$.

Let us consider a job-shop scheduling problem in which four jobs $\{J_1, J_2, J_3, J_4\}$ and each job has 3 tasks, therefore, $TASKS = \{t_i, |i = 1, 2, ..., 12\}$, $RS = \{ResourceType1, ResourceType2, ResourceType3\}$, $PRE = \{(t_i, t_{i+1}), (t_{i+1}, t_{i+2}), (t_{i+2}, t_{i+3}) | i = 0, 1, 2)\}$, $RT(t) = 0, \forall t \in TASK$.

Figure 1 shows a colored timed Petri net model of a scheduling problem, where the subnet drawn with black color shows the process net and the subnets colored with blue, green, and red correspond to the resource net. Note that each resource type may have multiple colored tokens, in which different colors in a resource place denote different capability but the same functionality.

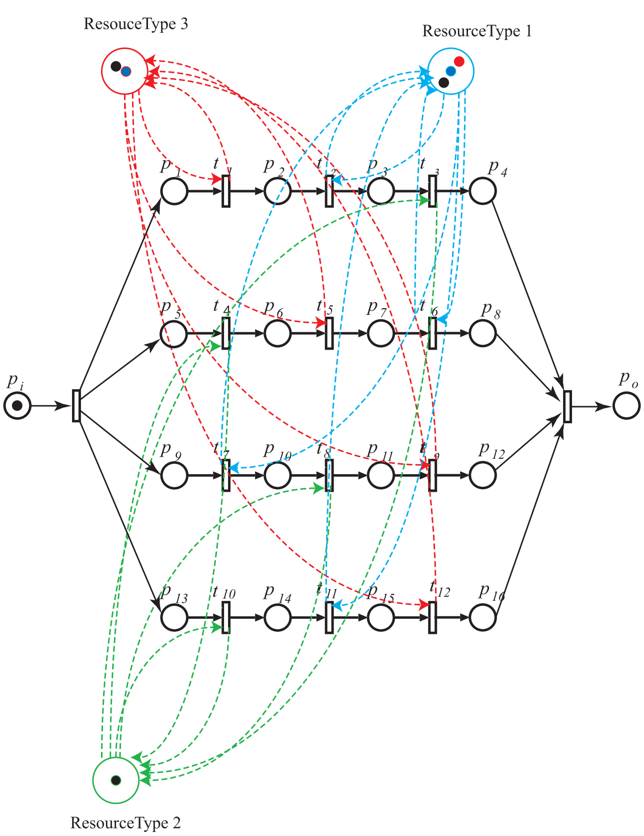

Fig. 1 Colored Timed Petri Net Model for a Scheduling Problem.

4. Extraction of Mixed Integer Programming from Colored Timed Petri Nets

In this section, we propose an algorithm to generate a mixed integer programming problem for MMASPs from TPNs.

4.1 Preparation

Input data for the algorithm is a timed Petri net model, $TPN = (P, T, Pre, Post, TS, D)$.

According to the discussion in Section 3, we generate the basic input data for scheduling problem, $SP = (TASK, RS, RR, PRE, RT, PT)$, as follows:

$$TASK = \{1, 2, ..., n\} \leftarrow T = \{t_1, t_2, ..., t_n\} \tag{5}$$

$$RS = \{1, 2, ..., |\cup_{rp \in RP} M[rp]|\} \tag{6}$$

$$RR(j) = \{1, ..., |M[rp]|\} \leftarrow \hat{Pre}(t_j, rp) \neq 0, \forall t_j \in T \tag{7}$$

$$PRE = \{(t_i, t_j) | \text{there exists } p \in P \text{ such that } Pre(p, t_j) \neq 0 \wedge Post(p, t_i) \neq 0\} \tag{8}$$

$$RT(j) = RT(t_j), \forall j \tag{9}$$

$$PT(j, k) = D(t_j, rt_{r,k}), k \in RR(j), \hat{Pre}(t_j, r) \neq 0, \forall t_j \in T \tag{10}$$

Let us define s_j and e_j to denote the start and end times of task $j, \forall j \in T$, respectively. Moreover, the binary variables x_j^k and $y_{i,j}$ are introduced as follows:

To represent the resource assignment, $\forall j \in TASK, \forall k \in RR(j)$,

$$x_j^k = \begin{cases} 1 & \text{if task } j \text{ is assigned to resource } k \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

To denote the order of the resource usage among tasks which use the same resource, $\forall (i, j) \in TASK \times TASK$,

$$y_{i,j} = \begin{cases} 1 & \text{if tasks } i \text{ and } j \text{ are assigned to the same resource and } i \text{ precedes } j \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

4.2 Constraints

To enforce the assignment of each task to exactly one resource, the following constraint is necessary.

$$\sum_{k \in RR(j)} x_j^k = 1, \forall j \in TASK \tag{13}$$

To state the starting and end time, the following constraint should be defined.

$$s_j + \sum_{k \in RR(j)} (PT(j, k) \cdot x_j^k) - e_j = 0, \forall j \in TASK \tag{14}$$

To ensure that processing of task j begins after processing of task i, if $y_{i,j} = 1$, the following constraint is defined.

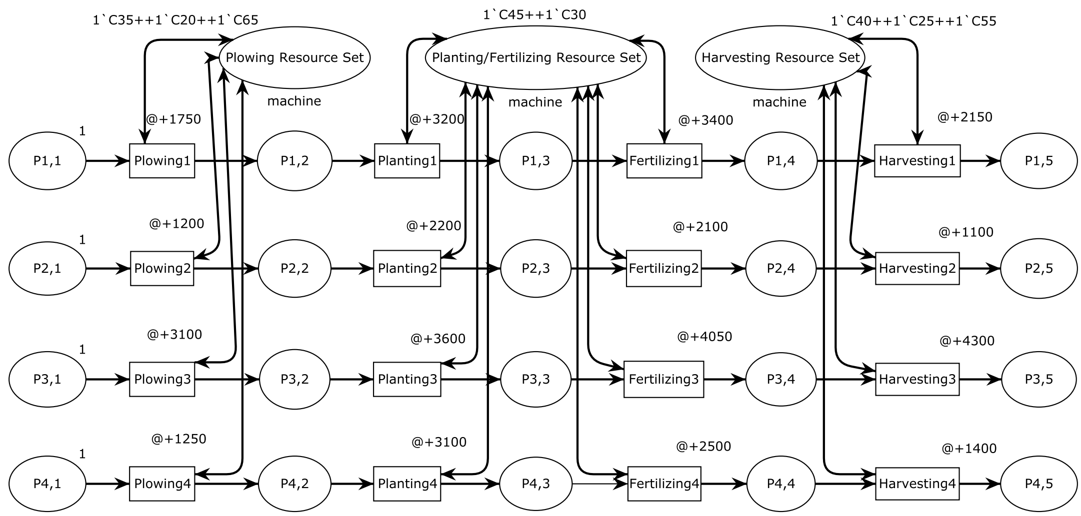

Fig. 2 Colored Timed Petri Net Model for Sugarcane Farming using CPN Tools 4.0.

Table 1 Petri Net Model for Case Study.

(a) Place and Initial Marking

Place	*Role*	*Initial Marking*
Plowing Resource Set	Resource Set for Plowing	C10, C20, and C80
Planting/Fertilizing Resource Set	Resource Set for Planting/Fertilizing	C30 and C40
Harvesting Resource Set	Resource Set for Harvesting	C40, C25, and C40
P *i*,1	Pre cond. for Plowing in Farm *i*	*UNIT*
P *i*,2	Post cond. for Plowing and Pre cond. for Planting in Farm *i*	ϕ
P *i*,3	Post cond. for Planting and Pre cond. for Fertilizing in Farm *i*	ϕ
P *i*,4	Post cond. for Fertilizing and Pre cond. for Harvesting in Farm *i*	ϕ
P *i*,5	Post cond. for Harvesting in Farm *i*	ϕ

Cx in *Initial Marking* means an attribution of the corresponding token and shows the specification (capability) of the resource.

(b) Transition

Transition	*Task*	*Task Size*
Plowing *i*	Plowing for Farm *i*	1,750, 1,200, 3,100, and 1,250 for Farm 1, 2, 3, and 4, respectively
Planting *i*	Planting for Farm *i*	3,200, 2,200, 3,600, and 3,100 for Farm 1, 2, 3, and 4, respectively
Fertilizing*i*	Fertilizing for Farm *i*	3,400, 2,100, 4,050, and 2,500 for Farm 1, 2, 3, and 4, respectively
Harvesting *i*	Harvesting for Farm *i*	2,150, 1,100, 4,300, and 1,400 for Farm 1, 2, 3, and 4, respectively

The average processing time for each task can be determined from the task size and the capacity of the assigned resource, Cx.

$$e_i - s_j + U \cdot y_{i,j} \leq U, \forall i, j \in \{(i, j) | i \neq j, r_i = r_j\} \quad (15)$$

where U is defined to represent a sufficiently large number.

To ensure that only one of two tasks i, j is processed before the other, the following constraint is defined.

$$y_{i,j} + y_{j,i} \leq 1, \forall i, j \in \{(i, j) | i \neq j, r_i = r_j\} \quad (16)$$

To ensure that if tasks i and j are assigned to resource k, then one must be processed before the other, the following constraint is defined.

$$x_i^k + x_j^k - y_{i,j} - y_{j,i} \leq 1, \forall i, j \in \{(i, j) | i \neq j, r_i = r_j\} \quad (17)$$

To guarantee that the sequencing variables $y_{i,j}$ and $y_{j,i}$ are zero if tasks i and j are assigned to different resources in the same resource group, the following constraint is necessary.

$$x_i^l + x_j^k + y_{i,j} + y_{j,i} \leq 2, \forall l, k, l \neq k, \forall i, j \quad (18)$$

To ensure the precedence relation between two tasks, the following constraint is defined.

$$s_j + \Big(\sum_{k \in RR(j)} (PT(j, k) \cdot x_j^k)\Big) \leq s_i, \forall (j, i) \in PRE \quad (19)$$

4.3 Objective Function

The objective function will try to minimize the makespan of the schedule in this paper even though other objectives are also available.

$$\min \max_{j \in TASK} e_j \quad (20)$$

For formulating the objective function as a linear function, we minimize a new variable *emax* (21) and add the linear constraints (22):

$$\min emax \quad (21)$$

$$emax \geq e_j, \forall j, (j, i) \notin PRE, \exists i \in TASK \quad (22)$$

We have all the constraints for feasible solutions specified in *Proposition 1*.

4.4 Algorithm and Implementation

The above subsections describe the steps of our automatic generation of mixed integer programming for scheduling problems and we can summarize the algorithm shown in Algorithm 1.

Algorithm 1 GenerateSchedulingMIP

Read timed Petri net model $TPN = (P, T, Pre, Post, TS, D)$
Convert TPN into $SP = (TASK, RS, RR, PRE, RT, PT)$ by (5)-(10)
Define real decision variables s_j and e_j for each $j \in TASK$
Define binary decision variables x_j^k for each $(j, k), j \in TASK$ and $k \in RR(j)$
Define binary decision variables $y_{i,j}$ for each (i, j) such that $i \neq j, (i, j) \in TASK \times TASK$
Generate all the constraints (13) - (19), and (22)
Generate objective function (21)

We implemented the algorithm with Ruby language (v. 2.2.1), where we utilize well-known CPN Tools [13] as a Petri net modeling tool. Petri net models drawn with CPN Tools can be exported to XML documents. The XML documents include not only their structural data but also attribute information such as time, arc weights, guard conditions, and functions. We omit the detail explanation of the implementation for the limited space.

5. Case Study

This section shows an example of a farm workflow scheduling where we model a sugarcane production process in four farms. We just model essential parts in the farm production process and readers may refer to our previous work for sugarcane workflow modeling details [6].

The sugarcane production process is composed of four serial tasks, plowing, planting, fertilizing, and harvesting. Each task requires a resource set for the corresponding work, and there may exist some available resource sets. Each resource set has its capability. Therefore, the working time for a task depends on the task size and the assigned resource set.

Figure 2 depicts a CPN model created by CPN Tools version 4.0. There are four farms and three resource places with initial resource sets (initial marking). Tables 1(a), 1(b) explain the details of the place and the transition sets, respectively. Note that the CPN model is based on the one shown in Fig. 1 though we omitted the source and the sink places in this modeling.

CPN Tools can output an XML document for the CPN model. Our developed program reads the XML file for the model shown in Fig. 2 and generates the mixed integer linear programming problem for its scheduling problem. Table 2 shows the number of variables and constraints for this example. Finally, gurobi optimizer solves the problem. Figure 3 is the Gantt chart representation of the schedule obtained by the optimizer.

Table 2 Size of Generated Mixed Integer Programming Problem.

Item	*Numbers*
Variables s_j	16
Variables e_j	16
Variables x_j^k (11)	40
Variables $y_{i,j}$ (12)	80
Variable *emax* (21)	1
Total Number of Variables	153
Constraints (13)	16
Constraints (14)	16
Constraints (15)	80
Constraints (16)	40
Constraints (17)	92
Constraints (18)	128
Constraints (19)	12
Constraints (22)	4
Total Number of Constraints	388

The numbers in the parentheses show the equation numbers.

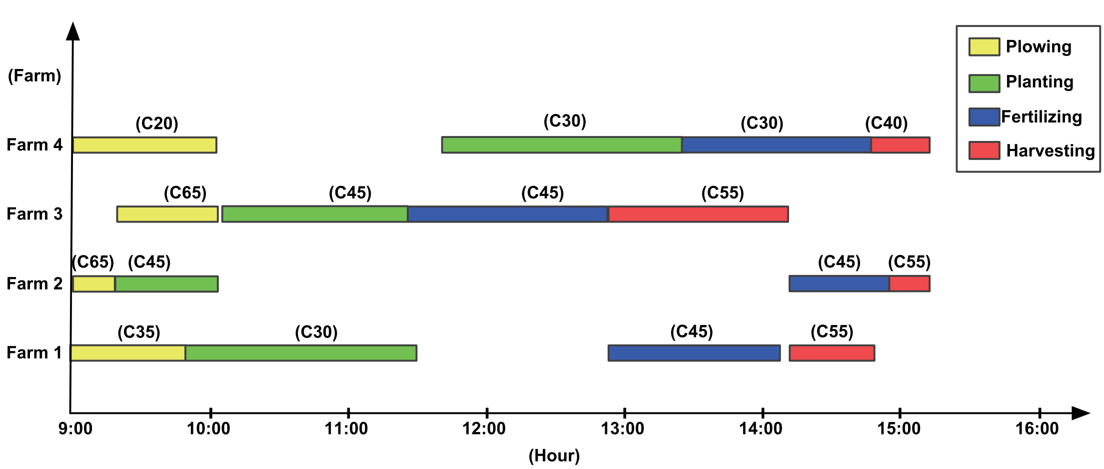

Fig. 3 Gantt Chart for the Obtained Farming Schedule by Gurobi Optimizer.

6. Concluding Remarks

This paper proposed a scheme for automatic generation of mixed-integer programming problems for scheduling with multiple resources based on colored timed Petri nets. Our developed tool reads Petri net data modeled by users, extracts the precedence and conflict relations among transitions, information on the available resources, and finally generates a mixed integer linear programming for exactly solving the target scheduling problem. The mathematical programing problems generated by our tool can be easily inputted to well-known optimizers. The results of this research can extend the usability of optimizers since our tool requires just simple rules of Petri nets but not deep mathematical knowledge.

As future works, we will relax some assumptions considered in this paper and treat uncertainty in optimization to get more practical usability.

References

[1] T. Murata, "Petri nets: Properties, analysis and applications," Proc. IEEE, vol.77, no.4, pp.541–580, 1989.
[2] A.B. Keha, K. Khowala, and J.W. Fowler, "Mixed integer programming formulations for single machine scheduling problems," Computers Industrial Engineering, vol.56, pp.357–367, 2009.
[3] J.C.-H. Pan and J.-S. Chen, "Mixed binary integer programming formulations for the reentrant job shop scheduling problem," Computers Operations Research, vol.32, no.5, pp.1197–1212, 2005.
[4] D.P. Ronconi and E.G. Birgin, "Mixed-integer programming models for flowshop scheduling problems minimizing the total earliness and tardiness," Just-in-Time Systems, Springer Optimization and Its Applications, pp.91–105, 2012.
[5] W.-Y. Ku and J.C. Beck, "Mixed integer programming models for job shop scheduling: A computational analysis," Computers Operations Research, vol.73, pp.165–173, 2016.
[6] S. Guan, M. Nakamura, T. Shikanai, and T. Okazaki, "Hybrid Petri nets modeling for farm work flow," Computers and Electronics in Agriculture, vol.62, no.2, pp.149–158, 2008.
[7] S. Guan, M. Nakamura, T. Shikanai, and T. Okazaki, "Resource assignment and scheduling based on a two-phase metaheuristic for cropping system," Computers and Electronics in Agriculture, vol.62, no.2, pp.181–190, 2009.
[8] CPLEX Optimizer, http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/
[9] GUROBI Optimizer, http://www.gurobi.com
[10] W.M.P. Van Der Aalst, "Petri net based scheduling," Operations-Research-Spektrum, vol.18, no.4, pp.219–229, 1996.
[11] G. Mušič, "Schedule optimization based on coloured Petri nets and local search," Proc. 7th Vienna International Conference on Mathematical Modelling, Mathematical Modelling, vol.7, Part 1, pp.352–357, 2002.
[12] M.A. Piera and G. Mušič, "Coloured Petri net scheduling models: Timed state space exploration shortages," Mathematics and Computers in Simulation, vol.82, no.3, pp.428–441, 2011.
[13] K. Jensen, L.M. Kristensen, and L. Wells, "Coloured Petri nets and CPN Tools for modeling and validation of concurrent systems," International Journal of Software Tools Technology Transfer, vol.9, no.3-4, pp.213–254, 2007.
[14] R. Ushijima, A.V. Porco, H. Kinjo, and M. Nakamura, "Automated generation of mixed Integer programming for job-shop scheduling problems based on Petri nets," The 31st International Technical Conference on Circuits/Systems, Computers and Communications" (ITC-CSCC), pp.287–290, Okinawa, Japan, 2016.
[15] W.M.P. van der Aalst, "The application of Petri-nets to workflow management," J. Circuits, Systems and Computers, vol.8, no.1, pp.21–66, 1998.