Automated Generation of Mixed Integer Programming for Scheduling Problems based on Petri Nets

Ryosuke Ushijima[1] Andrea Veronica Porco[2], Hideki Kinjo[3] and Morikazu Nakamura[4]
[1,2,4]Department of Information Engineering, University of the Ryukyus
1 Senbaru, Nishihara, Okinawa 903-0213, Japan
[3]Department of Law and Economics, Okinawa University
555 Kokuba, Naha, Okinawa 902-8521, Japan
E-mail : [3]kin@okinawa-u.ac.jp, [4]morikazu@ie.u-ryukyu.ac.jp

Abstract: This paper proposes a scheme for automated generation of a mixed-integer programming for scheduling problems based on timed Petri nets. Our tool reads XML-based Petri net data, extract precedence and conflict relations among transitions and then generates a mixed integer programming for the target scheduling problem. Therefore, once users model their system with Petri nets, they can address the scheduling problem for efficient operations. For implementation, we use CPN Tools, a well-known tool for editing, simulating and analyzing Colored Petri nets, for modeling. Users can model their systems with CPN Tools and then generate a mixed-integer programming problem for scheduling.

Keywords— **Petri net, Scheduling Problem, Mathematical Programming**

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 characteristics 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 because of NP-hardness of them[7], [8], [9], [6]. The problems are also valuable in practice since the problems are applicable in a broad range of fields. 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[10] and Gurobi Optimizer[11] and also freeware tools. However, only big companies benefit from this optimization approach. The reason for limited usage is not only an economical reason, but also usability. Users need to formulate their problems firstly as mathematical programming problems with mathematics. Therefore, an only limited quantity of users can utilize these tools.

In this paper, we present automated generation of mixed-integer programming for scheduling problems 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 researches [3], [4], [5].

2. Timed Petri Nets

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 weights 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 a matrix. 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 at M when $M(p) \geq Pre(p, t), \forall p \in \bullet t$ and a transition 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. In Petri net models, a transition t corresponds to an event, and its firing represents the occurrence of the event in the system. Dynamical behavior of a system by event occurrence can be represented by dynamic changing of token distribution by firing in Petri nets.

For quantitative analysis of the dynamical behavior of a system, many researchers introduced *time* to Petri nets. We can categorize the way 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, the ED, is such that a transition cannot be fired for a given period after enabled [3], [4], [5]. In this paper, we consider timed Petri nets with FD because of 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 $t \in T$. Tokens are attached a time stamp in TS when they are 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 $\tau_j + D[t]$ to each output token.

More details are explained in the literature [1].

3. Scheduling Problems

A scheduling problem is a 6-tuple $SP = (TASK, RS, RR, PRE, RT, PT)$, where *TASK* is a set of tasks, *RS* is a set of resources, $RR : TASK \rightarrow RS$ is a function which maps a task to its required resource, $PRE \subseteq TASK \times TASK$ is the precedence relation between two tasks, *TS* is the time set, usually the natural number set or the non-negative real number set. $RT : TASK \rightarrow TS$ is a function to show the release time of a task, $PT : TASK \rightarrow TS$ is a function to return the processing time of tasks.

We assume in this paper the followings in the scheduling problem:

1. No resource can process more than one task at a time.
2. Each resource is always available for processing, that is, *no breakdown*.
3. Operations can not be interrupted untill their completion, that is, *no preemption*.
4. The processing times are known in advance and they are deterministic.

For the scheduling problem, we verified the feasibility of the problem [3], [6].

Proposition 1: A schedule is feasible if the following constraints are satisfied:

1. All the precedence relations are satisfied: $\forall(t,t') \in PRE, ct(t) \leq st(t')$.
2. The release time conditions are satisfied: $\forall t \in TASK, st(t) \geq RT(t)$.
3. There exist no resource conflicts: $\forall t,t' \in TASK, (ct(t) \leq st(t') \vee ct(t') \leq st(t))$ if $RR(t) = RR(t')$,

where $st(t)$ and $ct(t)$ represent the start time and the completion time of task t, respectively.

Proof: The proposition holds since the three constraints explain direcly the definition of the scheduling problem. ■

When we solve scheduling problems, we need to specify the objective function. Usually we minimize the makespan, that is, the longest completion time of any task in *TASK*.

4. Petri Net Models of Scheduling Problems

A scheduling problem $SP = (TASK, RS, RR, PRE, RT, PT)$ can be modeled by a timed Petri net $TPN = (P, T, Pre, Post, TS, D)$. Table 1 shows rough correspondence between *SP* and *TPN*. *TASK* corresponds to T in *TPN*. *RS* maps tasks to places named M_i, and so on. We can easily model a scheduling problem with a timed Petri net.

Table 1. Rough Correspondence between SP and TPN

Scheduling Problem SP	*Timed Petri Net TPN*
TASK	T
RS	*places* named M_i
RR	*Pre* and *Post* (Coloured Arcs)
PRE	*Pre* and *Post* (Black Arcs)
RT	*Additional Comments*
PT	D

Let us consider a jobshop scheduling problem in which 4 jobs $\{J_1, J_2, J_3, J_4\}$ and each job J_i has 3 tasks $t_{i,1}, t_{i,2}, t_{i,3}$, therefore, $TASKS = \{t_{i,j} | i = 1,2,3,4, j = 1,2,3\}$, $RS = \{M_1, M_2, M_3\}$, $PRE = \{(t_{i,1}, t_{i,2}), (t_{i,2}, t_{i,3}) | i = 1,2,3,4\}$, $RT(t) = 0, \forall t \in TASK$. The problem is regarded as a jobshop scheduling problem in general cases of RR, while it is called *flowshop* when $RR(t_{i,j}) = M_j, \forall i,j$.

Figures 1 and 2 show a timed Petri net model of flow-shop and job-shop scheduling problems, respectively. The natural number below a transition expresses its duration time. For example, the duration time of $t_{1,1}$ equals 3. In the figures, places named M_1, M_2, M_3 correspond to resources, that is, machines in the production systems. The number of each machine is just one as shown by the number of tokens in each resource place. Job $J_j, j = 1, ..., 4$ has three tasks denoted by transitions $t_{j,1}$, $t_{j,2}$, and $t_{j,3}$. Places $p_{j,i}$ and $p_{j,i+1}$ mean the pre-condition and the post-condition of task t_{ji}, respectively.

From this process, we can confirm that users can model scheduling problems with the timed Petri net much easier than mathematical ways. Moreover, Petri net models are intuitively understandable.

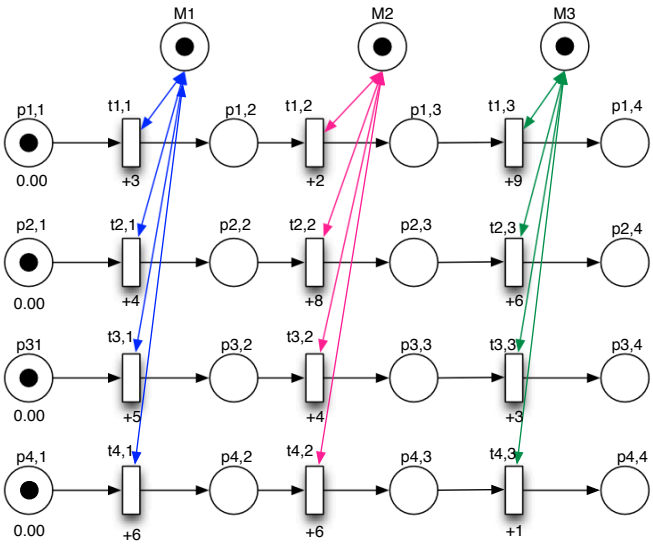

Figure 1. Flow-shop Scheduling Problem

5. Mathematical Programming Generation

We first generate resource information for each job $j, j = 1, 2, ..., n$, $R_j = (r_j^1, r_j^2, r_j^3, ..., r_j^m)$, where job j is composed of m tasks, and the i-th task of job j needs to use resource r_j^i.

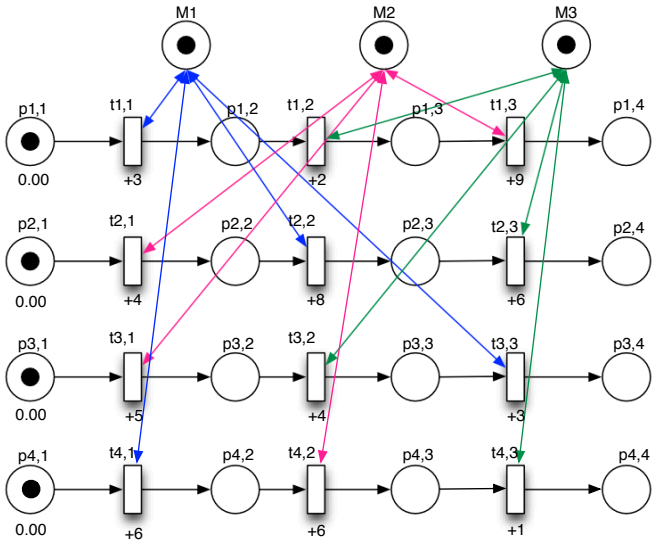

Figure 2. Job-shop Scheduling Problem

The resource information can be extracted from the Petri net structure.

We introduce real valued variables s_{j,r_j^i} to represent the starting time of job j's i-th task which requires resource r_j^i. Let us denote the duration time of task $t_{j,i}$ by $D[t_{j,i}]$. Therefore, the precedence relation among tasks in job j specified in the 1-st constraint of *Proposition 1* is represented by

$$s_{j,r_j^i} + D[t_{j,i}] \leq s_{j,r_j^{i+1}}, \tag{1}$$

$$\forall_j = 1, 2, ..., n, \forall_i = 1, 2, ..., m-1. \tag{2}$$

The constraints can be derived directly from the subnets corresponding to each job. Each of the subnets is a connected component of the graph constructed by removing the resource places and all the arcs connected to the places.

The second constraint of *Proposition 1* is simple. If we need to set available starting time $et_{i,j}$ for task i, j, the following constraints are added.

$$s_{j,r_j^i} \geq et_{i,j}, \forall_{i,j} \tag{3}$$

The third constraint of *Proposition 1* requires no conflict usage for all the resources. That is, each machine should be used by a single task at a time. By introducing binary variable $x^i_{j_k,j_h}, \forall i, \forall j_k, j_h \in \mathbf{J} \times \mathbf{J}$, we represent the order for the same resource usage between each pair of jobs.

$$s_{j1,i} \geq s_{j2,i} + D[t_{j2,i}], \quad \text{if} \quad x^i_{j1,j2} = 0 \tag{4}$$

$$s_{j2,i} \geq s_{j1,i} + D[t_{j1,i}], \quad \text{otherwise} \tag{5}$$

$$x^i_{j1,j2} \in \{0,1\}, \forall j1, j2, j1 < j2, \forall_i = 1, ..., m, \tag{6}$$

These constraints can be represented by the following linear equations by using a sufficiently large number H.

$$s_{j1,i} \geq s_{j2,i} + D[t_{j2,i}] - H \cdot x^i_{j1,j2} \tag{7}$$

$$s_{j2,i} \geq s_{j1,i} + D[t_{j1,i}] - H \cdot (1 - x^i_{j1,j2}) \tag{8}$$

$$x^i_{j1,j2} \in \{0,1\}, \forall j1 < j2, \forall_i = 1, ..., m. \tag{9}$$

For H, the following value can be used.

$$H = \sum_{j=1}^{n} \sum_{i=1}^{m} D[t_{i,j}] \tag{10}$$

We have all the constraints for feasible solutions specified in *Proposition 1* and we confirmed that the constraints are the same as the ones shown in [8], [9], [6].

Corollary 1: A schedule $s = (s_{i,j}, \forall i, j)$ is feasible for a given timed Petri net if the constraints shown in (1) to (3) and in (7) to (10) are satisfied.

6. Implementation with CPNTools

We implemented the automated mathematical programming generation based on CPN Tools[2]. Petri net models drawn with CPN Tools can be exported to XML documents. Figure 3 shows an example of the XML documents. The XML documents include not only their structural data but also attribute information such as time, arc weights, guard conditions, and functions. Therefore, we can easily construct mathematical programming problems of the scheduling problem from net models of scheduling problems.

Algorithm 1 shows the steps for automatic generation of a mixed integer programming for scheduling problems. We implemented the algorithm with a programming language Ruby 2.2.1.

Algorithm 1 SchedulingMIPGeneration

1: read the XML document.
2: extract the place and the transition IDs and their attribute information.
3: extract the arc IDs and their attribute information such as weight, link information.
4: construct *PRE* from the link information.
5: construct *RR* from the link information between a resource place and a transition.
6: generate all the precedence constraints $\forall t$ and $t' \in PRE$.
7: generate release time constraints if specified.
8: generate all no-resource-conflict constraints from *RR*.
9: generate the objective function.

Once we generate a mixed integer programming, we can solve the problem with a solver such as CPLEX and Gurobi Optimizer.

Figures 4 and 5 depict the Petri net models of the scheduled flow-shop and job-shop systems for the problems shown in Figs. 1 and 2, respectively.

XML

```
<place id="ID1412323818">
      ..........
<trans id="ID1412323829"
      ..........
<time id="ID1412323831">
   <text tool="CPN Tools"
         version="4.0.1">@+3</text>
      ..........
<arc id="ID1412324853"
          orientation="PtoT"
          order="1">
      ..........
   <transend idref="ID1412323829"/>
   <placeend idref="ID1412323818"/>
```

Figure 3. XML Document generated by CPN Tools

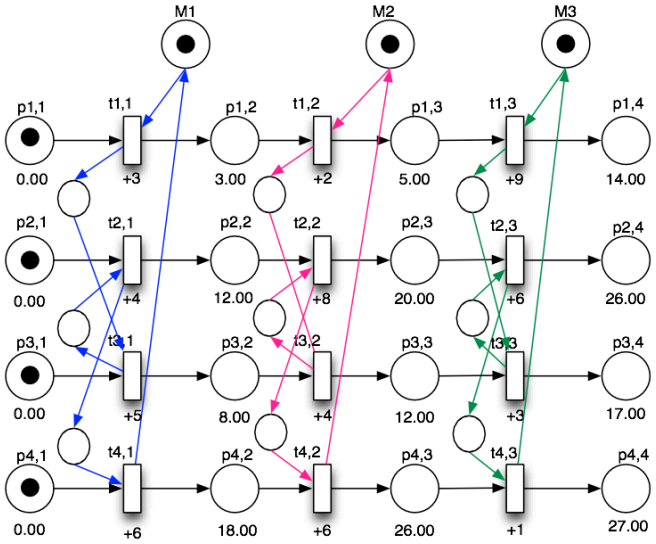

Figure 4. Net Model of Scheduled Flowshop System

7. Concluding Remarks

This paper proposed a scheme for automated generation of mixed-integer programming for scheduling problems based on Petri nets. Our developed tool reads XML-based Petri net data, analyzes precedence and conflict relations among transitions and then generates a mixed integer programming for the target scheduling problem. The scheduling problem we considered in this paper is limited by several assumptions. As future works, we will relax some of them for more practical usability.

References

[1] T. Murata, "Petri Nets: Properties, Analysis and Applications", Proceedings of the IEEE, Vol. 77, pp. 541-580, 1989.

[2] Kurt Jensen, Lars Michael Kristensen, Lisa Wells, "Coloured Petri Nets and CPN Tools for modeling and validation of concurrent systems, International Journal of Software Tools Technology Transfer, vol. 9, pp.213-254, 2007.

[3] W. M. P. van der Aalst, "Petri net based scheduling", Operations-Research-Spektrum, Volume 18, Issue 4, pp 219-229, 1996.

[4] Mušič, Gašsper, "Schedule optimization based on coloured Petri nets and local search", Proc. of the 7th Vienna International Conference on Mathematical Modelling, Mathematical Modelling , Volume 7, Part 1, pp.352-357, 2002.

[5] M.A. Piera, G. Mušič, "Coloured Petri net scheduling models: Timed state space exploration shortages", Mathematics and Computers in Simulation, Volume 82, Issue 3, Pages 428-441, 2011.

[6] Wen-Yang Ku, J. Christopher Beck, "Mixed Integer Programming models for job shop scheduling: A computational analysis", Computers & Operations Research, vol. 73, pp.165-173, 2016.

[7] Ahmet B. Keha, Ketan Khowala, John W. Fowler, "Mixed integer programming formulations for single machine scheduling problems", Computers & Industrial Engineering, vol. 56, pp. 357-367, 2009.

[8] Jason Chao-Hsien Pana, Jen-Shiang Chenb,"Mixed binary integer programming formulations for the reentrant job shop scheduling problem", Volume 32, Issue 5, pp.1197-1212, 2005.

[9] Débora P. Ronconi, Ernesto 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.

[10] CPLEX Optimizer, http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/.

[11] GUROBI Optimizer, http://www.gurobi.com.

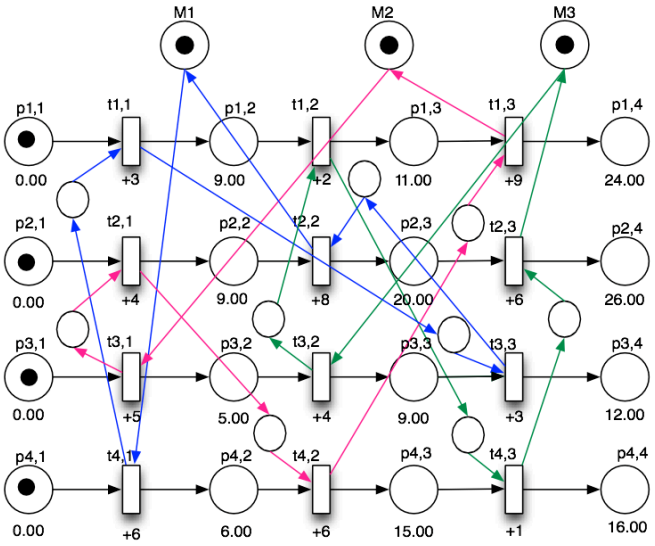

Figure 5. Net Model of Scheduled Jobshop System