Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Lagrangian Relaxation Technique for Solving Scheduling Problems by Decomposition of Timed Petri Nets

Tatsushi Nishi * Kenichi Shimatani * Masahiro Inuiguchi *

* *Mathematical Science for Social Systems, Graduate School of Engineering Science, Osaka University, 1-3 Machikaneyama, OSAKA 560-8531 (Tel: +81-6-6850-6351), (e-mail: nishi@sys.es.osaka-u.ac.jp).*

Abstract: In this paper, we propose Lagrangian relaxation technique for solving scheduling problems by decomposition of timed Petri nets. The scheduling problem is represented by the transition firing sequence problem to minimize a given objective function. The timed Petri net is decomposed into several subnets so that the subproblem for each subnet can be easily solved by a shortest path algorithm. The optimality of solution can be evaluated by the duality gap derived by Lagrangian relaxation method. The performance of the proposed method is compared with the conventional optimization algorithm with penalty function method. The results show that the duality gap within 1.5% can be derived for AGVs routing problems. The effectiveness of the proposed method is demonstrated by comparing the performance between the conventional method.

1. INTRODUCTION

Petri net is a mathematical modeling tool for representing wide range of discrete event systems as graphical and algebraic description. As a general scheduling solver, scheduling methods combined with Petri net modeling and novel combinatorial optimization algorithm have received much attention. This paper concentrates on the development of general scheduling tool for scheduling problems represented by timed Petri nets. The dynamics for Petri nets are determined by finding transition firing sequence from an initial marking to a final marking. The problem to determine a reachable transition firing sequence from initial marking to final marking is an NP-hard combinatorial optimization problem. It is well known that the legal firing sequence problem for Petri nets is NP-complete (Watanabe *et al.* [1989]). The problem to determine an optimal transition firing sequence to minimize the given objective function is also NP-hard. Typical approach for scheduling Petri nets is heuristic search for finding optimal transition sequence(Lee and DiCeare [1994]). The primal difficulty associated with the optimization of Petri nets is caused by state explosion problem for handling large-scaled problems. Conventionally, several decomposition methods for Petri nets are proposed to reduce computational complexity. Teng and Zhang [1993] developed a decomposition procedure for analysis and simulation of manufacturing systems for Petri nets. He *et al.* [2000] proposed an automatic generation procedure of decomposed Petri net model using IDEF3 methodology and its scheduling method by simulated annealing method. Petri net decomposition and coordination approach and its application to AGV routing problems, flowshop scheduling problems [Nishi and Maeno 2007] have been constructed by a penalty function optimization method. However, the conventional methods cannot ensure the optimality of solution derived by the optimization algorithm because lower bound for the original problem cannot be obtained by the conventional approaches.

The objective of the paper is to develop effective solution method for scheduling problems represented as timed Petri nets to resolve state explosion problem by the novel decomposition approach. The Lagrangian relaxation technique is used to evaluate the optimality of the solution derived by the proposed method. The tight lower bound and upper bound are derived by the proposed approach. The proposed method is applied to solve routing problems for multiple automated guided vehicles which is one of large scale optimization model by timed Petri nets.

The main contribution of the paper is stated as follows. We present a general decomposition and coordination method for solving scheduling problems represented by timed Petri net. The use of the Petri net decomposition procedure leads to derive separable formulation of Lagrangian function, which enables us to derive a lower bound to evaluate the optimality of solution by the duality gap between the lower and upper bounds. The reachability of the decomposed subnet is discussed from the setting of initial number of tokens for the decomposed subnet. Secondly, we give the proof that the subproblem is solved by Dijkstra's algorithm in polynomial computing time.

The paper is organized by the following sections. In Section 2, we present a mathematical modeling and formulation of scheduling problems represented by timed Petri nets. In Section 3, the timed Petri net is decomposed into several subnets. In Section 4, we propose the Lagrangian relaxation method for solving transition firing sequence problem by decomposition of timed Petri nets. Compu-

⋆ This work was supported in part by 2007 Tateishi Science and Engineering Foundation

tational experiments for examples are demonstrated on AGVs routing problems and the performance of the proposed method is compared with the conventional optimization algorithm with penalty function method in Section 5. In Section 6, we conclude our research and discuss some future works.

2. MODELING SCHEDULING PROBLEMS BY TIMED PETRI NETS

In this section, scheduling problems are represented as an transition firing sequence problem for timed Petri nets.

2.1 Definition of t-timed Petri Nets

Transition-timed Petri nets are used for modeling timed events for representing scheduling problems. A parameter $\theta(t)$ is introduced to describe transition firing time duration. In this model, firing transition is continued during $\theta(t)$ time period after the tokens of the input place are removed from the input place when a transition t is fired. If the firing is completed, the tokens are placed into the output place. Throughout the paper, we use the following notations. $\mathbb{R}$ is set of real values, $\mathbb{N}$ is set of non-negative integers. A^B represents set of vector with elements of A whose dimension is $|B|$. $A^{B\times C}$ represents set of vector with elements of A whose dimension is $|B| \times |C|$. A^{T} is transposed matrix of A.

The notation of TPN, marking, firing vector, and firing condition are defined as follows.
Definition 1: Transition-timed Petri net TPN is defined as $TPN = (P, T, w, M_0, \theta)$ where P is finite set of places; $P = \{p_1, p_2, \cdots, p_{|P|}\}$, T is finite set of transitions; $T = \{t_1, t_2, \cdots, t_{|T|}\}$, w is incident relationship between places and transitions; $w : (P\times T)\cup(T\times P) \to \mathbb{N}$, and M_0 is initial marking; $M_0 : P \to \mathbb{N}$, $\theta : T \to \mathbb{N} - \{0\}$ is the function of transition sets to time duration. For $p \in P, t \in T$, $w(p,t) = 0$ $(w(t,p) = 0)$ indicates that there is no arc from p to t (from t to p). The value $w(p,t) > 0 (w(t,p) > 0)$ indicates the weight on arc from p to t (from t to p). $t' \in T$ is an input transition for place p' if $w(t', p') > 0$.
Definition 2: A firing vector $r_k : T \to \{0, 1\}$ is a vector with 0-1 binary valuables whose elements take the value of if a transition is fired at time $k \in \mathbb{N}$ and zero otherwise. A marking vector $M_k : P \to \mathbb{N}$ is a vector whose elements are the number of tokens. These variables can be defined as column vector $M_k \in \mathbb{N}^P, r_k \in \{0,1\}^T$ satisfying $(M_k)_i = M_k(p_i)$ $(\forall p_i \in P), (r_k)_j = r_k(t_j)$ $(\forall t_j \in T)$.
Definition 3: The incident matrices $A^+_{TPN} \in \mathbb{N}^{T\times P}$, $A^-_{TPN} \in \mathbb{N}^{T\times P}$ are defined as $(A^+_{TPN})_{ji} = w(t_j, p_i)$, $(A^-_{TPN})_{ji} = w(p_i, t_j)$.
Definition 4: $c_k(t)$ represents the residual time for firing duration. $c_k \in \mathbb{N}^{\mathrm{T}}$ is a column vector of $c_k(t)$ for $t \in T$. The firing condition for TPN is given by

$$M_k - (A^-_{TPN})^{\mathrm{T}} r_k \geq 0 \tag{1}$$

$$(c_k)^{\mathrm{T}} \cdot r_k = 0 \tag{2}$$

where $r_k \in \{0, 1\}$ is a firing vector which takes a value of 1 if transition is fired at time k and otherwise takes a value of 0. If a feasible firing vector r_k is given, the marking M_k is changed into the marking M^+_k.

$$M^+_k = M_k - (A^-_{TPN})^{\mathrm{T}} r_k \tag{3}$$

The residual time is assigned if a transition is fired at time k.

$$c^+_k(t) = \begin{cases} \theta(t) & (r_k(t) = 1) \\ c_k(t) & (\text{otherwise}) \end{cases} \tag{4}$$

$w(t,p)$ tokens are added to the output place for transition t if $c^+_k(t) = 1$.

$$M_{k+1} = M^+_k + (A^+_{TPN})^{\mathrm{T}} e_k \tag{5}$$

$$e_k(t) = \begin{cases} 1 & (c^+_k(t) = 1) \\ 0 & (\text{otherwise}) \end{cases} \tag{6}$$

$c^+_k(t)$ is changed by the following equation in time $k + 1$.

$$c_{k+1}(t) = \begin{cases} c^+_k(t) - 1 & (c^+_k(t) \geq 1) \\ 0 & (\text{otherwise}) \end{cases} \tag{7}$$

2.2 Decomposable formulation of optimal transition firing sequence problem

The problem to determine a feasible firing sequence to minimize the given objective function for TPN is defined in this section. Given $TPN = (P, T, w, M_0, \theta)$, a final marking $M_f : P' \to \mathbb{N}$ $(P' \subset P)$, time horizon $N_t \in \mathbb{N} - \{0\}$, and objective function $J : (\{0,1\}^T)^{N_t} \to \mathbb{R}$, the problem is to derive a set of firing vector $(r_0, r_1, \ldots, r_{N_t-1}) \in (\{0,1\}^T)^{N_t}$ to minimize J satisfying $c_{N_t} = 0 \wedge M_{N_t}(p) = M_f(p)$ $(\forall p \in P')$. We call the problem as an optimal transition firing sequence problem.

The decomposable TPN satisfies the following three conditions.

(i) The entire system consists of several entities. The dynamics of each entity u_i $(1 \leq i \leq m)$ is represented by each subnet described by TPN.
(ii) The entire objective function $J = \sum_{i=1}^m J_{u_i}$ can be written by the sum of each objective function $J_{u_i}(r^{u_i}_0, r^{u_i}_1, \ldots, r^{u_i}_{N_t})$ for each entity u_i $(1 \leq i \leq m)$.
(iii) The final marking $M_f(p)$ is not defined for any place $p \in P_R$. P_R is defined in Section 3.1.

The entire system consists of entities $u_1,\ u_2, \ldots, u_m$. $M^{u_i}_f$ is the final marking for entity u_i. A variable $\delta_{u_i,k}$ indicates 0 if the marking for entity u_i has reached the final marking $M^{u_i}_f$ and otherwise 1. The sum of total transition time from initial marking to the final marking for entity u_i can be written as $\sum_{k=0}^{N_t} \delta_{u_i,k}$. Here, N_t is sufficiently large time horizon to complete all of the transitions for the entities. The variable $\delta_{u_i,k}$ satisfies the following equation.

$$\delta_{u_i,k} = \begin{cases} 1 & (M^{u_i}_k \neq M^{u_i}_f) \\ 0 & (M^{u_i}_k = M^{u_i}_f) \end{cases} \tag{8}$$

Under the assumption that the objective function is additive for each entity u_i, the optimal transition firing sequence problem can be formulated as the following equations.

$$\min_{\{r_k\}} \sum_{i=1}^{m} J_{u_i}(\{\delta_{u_i,k}\}) \tag{9}$$

subject to (1) – (9)

where J_{u_i} is the given objective function depending on variable $\delta_{u_i,k}$ defined for entity u_i, and $\delta_{u_i,k} \in \{0,1\}$, $r_k \in \{0,1\}^T$.

3. LAGRANGIAN RELAXATION METHOD FOR SOLVING OPTIMAL TRANSITION FIRING SEQUENCE PROBLEM BY DECOMPOSITION OF PETRI NETS

3.1 Decomposition of Petri nets

In this section, we propose the Lagrangian relaxation method for solving optimal transition firing sequence problem by the decomposition of TPN. The transition firing sequence problem (P) is a combinatorial optimization problem which is extremely difficult to solve in realistic computing time. The Lagrangian relaxation method removes the complicating constraints from constraints and replaces with a penalty term in the objective function of the original problem(Fisher [1981]). It is difficult to select which constraints should be relaxed or not from the set of constraints for problem (P). To reduce the complexity, we first apply a general decomposition scheme to decompose timed Petri Nets (TPN) into several subnets.

The set of transitions T is divided into the set of transitions T_{u_i} for each entity u_i by (10). Here, $A \sqcup B$ indicates disjoint union of the set A and the set B.

$$T = T_{u_1} \sqcup T_{u_2} \sqcup \cdots \sqcup T_{u_m} \tag{10}$$

The set of places P is divided into P_{u_i} and P_R. P_{u_i} is the set of places where both of the input transition of the place and the output transition of the place are the elements of the set of T_{u_i}. P_R is the set of places where the place is not the element of P_{u_i}. Thus, the set of places can be divided into subsets by (11).

$$P = P_{u_1} \sqcup P_{u_2} \sqcup \cdots \sqcup P_{u_m} \sqcup P_R \tag{11}$$

where P_{u_i} and P_R are defined as the following equation. $OUT(p)$ is the set of output transitions for place p, $IN(p)$ is the set of input transitions for place p.

$$P_{u_i} = \{\ p \mid IN(p) \subseteq T_{u_i},\ OUT(p) \subseteq T_{u_i}\} \tag{12}$$
$$P_R = P \backslash (P_{u_1} \sqcup P_{u_2} \sqcup \cdots \sqcup P_{u_m}) \tag{13}$$

$r_k^{u_i} \in \{0,1\}^{T_{u_i}}$ is the column vector comprising $t \in T_{u_i}$ for $r_k(t)$. $M_k^{u_i} \in \mathbb{N}^{P_{u_i}}$ is the column vector comprising $M_k^{u_i} \in \mathbb{N}^{P_{u_i}}$, and $M_k^R \in \mathbb{N}^{P_R}$ is the column vector comprising $p \in P_R$ for $M_k(p)$. Thus, the firing condition for TPN can be written as (14) and (15) by $A_{u_i}^+ \in \mathbb{N}^{T_{u_i} \times P_{u_i}}$, $A_{u_i}^- \in \mathbb{N}^{T_{u_i} \times P_{u_i}}$, and

$$B^- = [(B_{u_1}^-)^{\mathrm{T}}, \ldots, (B_{u_m}^-)^{\mathrm{T}}]^{\mathrm{T}} \quad (B_{u_i}^- \in \mathbb{N}^{T_{u_i} \times P_R})$$
$$B^+ = [(B_{u_1}^+)^{\mathrm{T}}, \ldots, (B_{u_m}^+)^{\mathrm{T}}]^{\mathrm{T}} \quad (B_{u_i}^+ \in \mathbb{N}^{T_{u_i} \times P_R})$$

$$M_k^{u_i} - (A_{u_i}^-)^{\mathrm{T}} r_k^{u_i} \geq 0 \ (1 \leq i \leq m) \tag{14}$$
$$M_k^R - (B^-)^{\mathrm{T}} r_k \geq 0 \tag{15}$$
$$(c_k^{u_i})^{\mathrm{T}} r_k^{u_i} = 0 \quad (1 \leq i \leq m) \tag{16}$$

The state equation can be written as (17)–(20).

$$M_k^{u_i+} = M_k^{u_i} - (A_{u_i}^-)^{\mathrm{T}} r_k^{u_i} \tag{17}$$
$$M_{k+1}^{u_i} = M_k^{u_i+} + (A_{u_i}^+)^{\mathrm{T}} e_k^{u_i} \tag{18}$$
$$M_k^{R+} = M_k^R - (B^-)^{\mathrm{T}} r_k \tag{19}$$
$$M_{k+1}^R = M_k^{R+} + (B^+)^{\mathrm{T}} e_k \tag{20}$$

where $e_k^{u_i} \in \{0,1\}^{T_{u_i}}$ is the column vector satisfying (6) for each $t \in T_{u_i}$. Consider a TPN model illustrated in Fig. 1(a). The transitions t_1, t_2 represent the dynamics of entity u_1, and transitions t_3, t_4 represent the dynamics of entity u_2. The set of transitions T are decomposed as $T = T_{u_1} \cup T_{u_2}$ where $T_{u_1} \cap T_{u_2} = \phi$. The set of places whose input and output transitions are the elements of T_{u_i} is defined as $P_{u_i} (1 \leq i \leq 2)$. The set of places which are not belong to P_{u_1} and P_{u_2} is defined as P_R. Here, $P_{u_1} = \{p_{u_1,s_1},\ p_{u_1,s_2},\ p_{u_1,s_3}\}$, $P_{u_2} = \{p_{u_2,s_4},\ p_{u_2,s_2},\ p_{u_2,s_5}\}$, $P_R = \{p_{o,s_2}\}$. The resource places are duplicated as P_{o,s_2} into P'_{o,s_2} and P''_{o,s_2}. The entire Petri net is decomposed into two subnets as illustrated in Fig. 1(b).

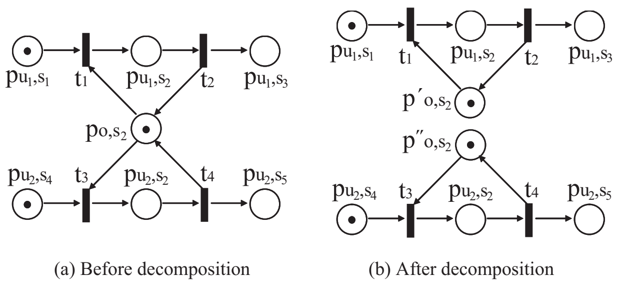

Fig. 1. An example of decomposition for timed Petri net

From the decomposition scheme, TPN is decomposed into each subnet $TPN^i = (P_{u_i} \cup P_{Ru_i}, T_{u_i}, w^{u_i}, M_o^i)$ for entity u_i. P_{Ru_i} is the set of places which has one-to-one mapping with P_R. M_o^i is the initial marking defined as (22). w^{u_i} can be expressed by the bijection $\zeta : P_{Ru_i} \to P_R$,

$$\begin{aligned} w^{u_i}(p,t) &= w(p,t) \quad (\forall p \in P_{u_i}; \forall t \in T_{u_i}) \\ w^{u_i}(t,p) &= w(t,p) \quad (\forall p \in P_{u_i}; \forall t \in T_{u_i}) \\ w^{u_i}(p,t) &= w(\zeta(p),t) \quad (\forall p \in P_{Ru_i}; \forall t \in T_{u_i}) \\ w^{u_i}(t,p) &= w(t,\zeta(p)) \quad (\forall p \in P_{Ru_i}; \forall t \in T_{u_i}) \end{aligned} \tag{21}$$

The aggregated TPN model $TPN^D = (P_D, T, w^D, M_o^D)$ is defined for all of the decomposed subnets for TPN^i $(1 \leq i \leq m)$. By using the aggregated model, the initial marking and the final marking for the decomposed subnet TPN^i are also defined.

$$\begin{aligned} P_D &= \bigcup_{i=1}^{m} (P_{u_i} \cup P_{Ru_i}) \\ w^D(p,t) &= w^{u_i}(p,t) \ (\forall p \in P_{u_i} \cup P_{Ru_i}; \forall t \in T_{u_i}; \forall i) \\ w^D(t,p) &= w^{u_i}(t,p) \ (\forall p \in P_{u_i} \cup P_{Ru_i}; \forall t \in T_{u_i}; \forall i) \\ M_0^D(p) &= M_0^i(p) \ (\forall p \in P_{u_i} \cup P_{Ru_i}; \forall i) \end{aligned} \tag{22}$$

Since P_R is not included in the domain P' for the final marking for TPN from the decomposable condition (3), the final marking $M_f^i : P_{u_i} \cap P' \to \mathbb{N}$ for TPN^i and the final marking for $M_f^D : P^D \cap P' \to \mathbb{N}$ can be written as the following equations.

$$M_f^i(p) = M_f(p) \quad (\forall p \in P_{u_i} \cap P') \tag{23}$$
$$M_f^D(p) = M_f(p) \quad (\forall p \in P^D \cap P') \tag{24}$$

$M_k^i : P_{u_i} \cup P_{Ru_i} \to \mathbb{N}$ is the marking for TPN^i, and $M_k^{Ru_i} \in \mathbb{N}^{P_{Ru_i}}$ is the column vector comprising $M_k^i(p)$ for $p \in P_{Ru_i}$. The firing condition for TPN^i can be described as (14), (15), and (25). The state equation for TPN^i can be written as (17), (20), and (26).

$$M_k^{Ru_i} - (B_{u_i}^-)^{\mathrm{T}} r_k^{u_i} \geq 0 \tag{25}$$
$$M_{k+1}^{Ru_i} = M_k^{Ru_i} + (B_{u_i}^+)^{\mathrm{T}} e_k^{u_i} - (B_{u_i}^-)^{\mathrm{T}} r_k^{u_i} \tag{26}$$

4. LAGRANGIAN RELAXATION METHOD

Lagrangian relaxation method is widely used for obtaining lower bound for combinatorial optimization problems which has been applied to solve complex scheduling problems recently. The optimality of solution can be evaluated by using duality gap. The method is used for solving optimal transition firing sequence problems for Petri nets.

4.1 Separable formulation of Lagrangian function

The firing condition for P_{Ru_i} is removed and added to the objective function multiplied by non-negative Lagrangian multiplier vector $\lambda_k \in \mathbb{R}_+^{P_{Ru_i}}$. The Lagrangian relaxation problem is decomposed into several subproblems. The Lagrangian function can be defined by the following equation.

$$L = \sum_{i=1}^{m} J_{u_i} + \sum_{k=0}^{N_t} \lambda_k^{\mathrm{T}} \{(B^-)^{\mathrm{T}} r_k - M_k^R\} \tag{27}$$

From the state equation of (20), we can obtain $M_k^R = M_{k-1}^R + (B^+)^{\mathrm{T}} e_k - (B^-)^{\mathrm{T}} r_k = M_{k-2}^R + (B^+)^{\mathrm{T}}(e_{k-2} + e_{k-1}) - (B^-)^{\mathrm{T}}(r_{k-2} + r_{k-1}) = \cdots = M_0^R + (B^+)^{\mathrm{T}} \sum_{l=0}^{k-1} e_l - (B^-)^{\mathrm{T}} \sum_{l=0}^{k-1} r_l$. Thus, $M_k^R = M_0^R + (B^+)^{\mathrm{T}} \sum_{l=0}^{k-1} e_l - (B^-)^{\mathrm{T}} \sum_{l=0}^{k-1} r_l$ can be derived. By using the equation, the Lagrangian function can be rewritten as:

$$\begin{aligned} L = & \sum_{i=1}^{m} J_{u_i} - \sum_{k=0}^{N_t} \lambda_k^{\mathrm{T}} M_0^R \\ & + \sum_{k=0}^{N_t} \lambda_k^{\mathrm{T}} [(B^-)^{\mathrm{T}} r_k - \{(B^+)^{\mathrm{T}} \sum_{l=0}^{k-1} e_l - (B^-)^{\mathrm{T}} \sum_{l=0}^{k-1} r_l\}] \\ = & \sum_{i=1}^{m} L_{u_i} - \sum_{k=0}^{N_t} \lambda_k^{\mathrm{T}} M_0^R \end{aligned} \tag{28}$$

where $L_{u_i} = J_{u_i} + \sum_{k=0}^{N_t} \lambda_k^{\mathrm{T}} [(B_{u_i}^-)^{\mathrm{T}} r_k^{u_i} - \{(B_{u_i}^+)^{\mathrm{T}} \sum_{l=0}^{k-1} e_l^{u_i} - (B_{u_i}^-)^{\mathrm{T}} \sum_{l=0}^{k-1} r_l^{u_i}\}]$. From the above formulation, the Lagrangian relaxation problem to minimize L can be reduced to the minimization problem defined by (RP_{u_i}) as the following equation for each L_{u_i} for entity u_i.

$$\begin{aligned} &(RP_{u_i}) \ \min L_{u_i}, \quad \text{where } L_{u_i} = J_{u_i} + \\ &\sum_{k=0}^{N_t} \lambda_k^{\mathrm{T}} \{(B_{u_i}^-)^{\mathrm{T}} r_k^{u_i} - (B_{u_i}^+)^{\mathrm{T}} \sum_{l=0}^{k-1} e_l^{u_i} + (B_{u_i}^-)^{\mathrm{T}} \sum_{l=0}^{k-1} r_l^{u_i})\} \\ &\text{s. t. } (14), (17), (25), (26) \end{aligned} \tag{29}$$

4.2 Solving subproblem

The subproblem of (29) is an integer programming problem. The subproblem can be formulated by Dijkstra's algorithm. For example, J_{u_j} is the total time for AGV u_i required for marking $M_k^{u_j}$ to transit from the initial marking $M_o^{u_j}$ for multiple AGV routing problems. Let P_s^j denote the set of states. h_k is a function which indicates 1 if the marking has reached to the final marking and zero otherwise.

$$\begin{aligned} P_s^j = \{\sigma \mid \sigma = (M_k^j,\ k,\ h_k); k \in \mathbb{N};\ 0 \leq k \leq N_t; \\ h_k \in \{0,1\};\ h_k = 0 \vee (M_k^{u_j} = M_f^{u_j} \to 1)\} \end{aligned} \tag{30}$$

If the marking $M_k^{u_j}$ is the same as the final marking, two types of states, $(M_k^j, k, 1)$ and $(M_k^j, k, 0)$ are generated. $(M_k^j, k, 1)$ is the state that the marking is identical with the final marking and all of the transitions are completed. On the other hand, $(M_k^j, k, 0)$ is the state that the marking is identical with the final marking but the transitions are not completed. Once the state $(M_k^j, k, 1)$ is generated, no other states can be reachable except $(M_k^j, k+1, 1)$ from the state.

The cost function from a state $\sigma_a = (M_k', k, h_k) \in P_s^j$ to a state takes the value of 1 if $h_k = 0$ (the marking is not the final one) and zero otherwise when $h_k = 1$ (the marking is the same as the final one). The transition cost from state σ_a to state σ_b can be represented as the following equation from the constraints stated above.

$$\begin{aligned} &d_{u_i,a,b} = \delta_{u_i,k} \\ &+ \sum_{p \in P_R} \lambda_{p,k} \{(B_{u_i}^-)^{\mathrm{T}} r_k^{u_i} - (B_{u_i}^+)^{\mathrm{T}} \sum_{l=0}^{k-1} e_l^{u_i} + (B_{u_i}^-)^{\mathrm{T}} \sum_{l=0}^{k-1} r_l^{u_i}\} \end{aligned}$$

The Dijkstra's algorithm for solving subproblem is described as the following steps.

Step 1. The initial condition is set as $\sigma_0 = (M_0^j, 0, 0)$, $\mathcal{P}_o = P_s^j - \{\sigma_o\}, G(\sigma_0) = 0, g_0(\sigma) = \infty\ (\forall \sigma \in \mathcal{P}_o), k = 0$.
Step 2. if $\mathcal{P}_k = \phi$, then the algorithm is completed.
Step 3. For every state σ, the following recursion is calculated.

$$g_{k+1}(\sigma) = \min\{g_k(\sigma),\ G(\sigma_k) + d_{u_j,\sigma_k,\sigma}\} \quad (\forall \sigma \in P_k)$$

Step 4. Find a state σ_{k+1} where $G(\sigma_{k+1}) = g_{k+1}(\sigma_{k+1}) = \min_{\sigma \in \mathcal{P}_k} g_{k+1}(\sigma)$ and $\mathcal{P}_{k+1} = \mathcal{P}_k \backslash \{\sigma_{k+1}\}$.
$k := k + 1$ and return to Step 2.

If the cost function is a monotonic function with respect to the number of states, the subproblem can be formulated as a shortest path problem that can be solved by Dijkstra's algorithm where P_s^j is the set of nodes, $d_{a,b}$ is the length of nodes. The computational complexity for Dijkstra's algorithm is $O(m^2)$ where m is the number of nodes.

4.3 Optimization algorithm using Lagrangian relaxation

The optimization algorithm using Lagrangian relaxation is explained in this section.

Step 1. Initialization
The number of iteration is initialized as $N := 1$. The value of Lagrangian multipliers is also initialized ($\lambda_{p,k}^{(N)} = \{0\}$).

Step 2. Solving subproblem
Each subproblem (RP_{u_i}, $i = 1, \dots, m$) is sequentially solved from $RP_{u_1}, RP_{u_2}, \dots, RP_{u_m}$. The value of L obtained from the solution of Lagrangian relaxation problem, gives the lower bound LB for the original problem as the following equation.

$$LB = \sum_{i=1}^{m} L_{u_i}(\tilde{r}_k^{u_i}) - \sum_{k=0}^{N_t} \lambda_k^{\mathrm{T}} M_0^R \tag{31}$$

$$\tilde{r}_k^{u_i} = \arg \min_{\{r_k^{u_i}\}} L_{u_i} \tag{32}$$
$$\text{s. t.}\quad (14), (17), (25), (26)$$

Step 3. Construction of a feasible solution
The solution derived at step 2 is not always feasible. Some firing constraints may be violated. To ensure the generation of feasible solution, the procedure to construct a feasible solution (See section 4.4) is executed. The upper bound UB of original problem is updated by the value of objective function for the derived feasible solution.

Step 4. Evaluating convergence
The difference between the upper bound and the lower bound is less than pre-determined value, the algorithm is completed.

Step 5. Updating Lagrange multipliers
The Lagrangian multipliers are optimized by subgradient method. The subgradient of Lagrangian function can be calculated from the tentative solution of each subproblem by (33).

$$g_k = (B^-)^{\mathrm{T}} \tilde{r}_k - M_k^R \tag{33}$$

$\tilde{r}_k$ is the firing vector derived at step 2. The value of Lagrangian multiplier is updated by (34).

$$\lambda_{p,k}^{(N+1)} = \max\{0,\ \lambda_{p,k}^{(N)} + \alpha \frac{UB - LB}{|g_k|^2} g_k\} \tag{34}$$

α is a positive constant. $N := N + 1$ and return to Step 2.

4.4 Construction of a feasible solution

In most cases, the solution derived at step 2 of the algorithm in section 4.3 is infeasible for the entire system. To ensure the generation of feasible solution, the following procedure is executed during the iteration if the solution derived at step 2 is infeasible. The key idea of the algorithm is that conflicts between the entities are resolved by successively delaying the firing time for each entity if the firing constraints are violated. r_k is the firing vector obtained at step 2. $\tau_k^{u_i}$ is the firing delay time for entity u_i.
[Procedure for constructing a feasible solution]

Step 1 $\tau_k^{u_i} := 0$ $(0 \le k \le N_t, 1 \le i \le m)$, $k := 0$, $F := \emptyset$

Step 2 $\tilde{r}_k := [(\tilde{r}_k^{u_1})^{\mathrm{T}}, (\tilde{r}_k^{u_2})^{\mathrm{T}}, \dots, (\tilde{r}_k^{u_m})^{\mathrm{T}}]^{\mathrm{T}}, \tilde{r}_k^{u_i} = 0$ if $u_i \in F$, $\tilde{r}_k^{u_i} = r_{k-\tau_k^{u_i}}^{u_i}$ $u_i \notin F$. If $M_k^R - (B^-)^{\mathrm{T}} \tilde{r}_k \ge 0$, then go to Step 4, otherwise go to Step 3.

Step 3 $G := \{u_i \mid \exists p \in P_{inf}, B_{u_i}^-(p)^{\mathrm{T}} r_k^{\tilde{u}_i} \ge 1\}$ where $P_{inf} := \{p \mid M_k^R(p) - (B^-(p))^{\mathrm{T}} \tilde{r}_k < 0\}$, $M_k^R(p)$ is the element of M_k^R for place p, $B_{u_i}^-(p)^{\mathrm{T}}, B^-(p)^{\mathrm{T}}$ represents the row vector corresponding to place p for $(B^-)_{u_i}^{\mathrm{T}}$, $(B^-)^{\mathrm{T}}$, respectively. The entities which have the minimum delay are selected as S_k such that $S_k := \{u_i \mid \tau_k^{u_i} = \min_{u_j \in G} \tau_k^{u_j}\}$. An entity is selected from S_k. $\tau_k^{u_l} := \tau_k^{u_l} + 1$, $F := F + \{u_l\}$, return to Step 2.

Step 4 $\tau_{k+1}^{u_i} := \tau_k^{u_i} (1 \le i \le m)$, $k := k + 1$, $F := \emptyset$, if $k = N_t$, the algorithm is completed. Otherwise return to Step 2.

Note that the procedure can derive a feasible solution on the assumption that there exists at least a feasible solution to satisfy firing condition and state equation. If even a feasible solution does not exist for the problem, the procedure is terminated with $P_{inf} \neq \phi$.

5. COMPUTATIONAL EXPERIMENTS

5.1 Application to muitiple AGVs routing problems

The performance of the proposed method is investigated for routing problems for automated guided vehicles in transportation system with 143 nodes which is illustrated in Fig. 2. The routing problem for AGVs is to determine a feasible routing for multiple AGVs to minimize the total transportation time satisfying the condition that the AGVs do not collide with other AGVs on each node and each edge when the starting and ending node for each AGV are given. The formulation of routing problem for multiple AGVs is described in Nishi *et al.* [2005]. The difficulty to find an optimal routing to minimize total transportation time is much more increasing when the number of AGVs is increased with a fixed number of nodes because the deadlock and interferences with AGVs are also increased.

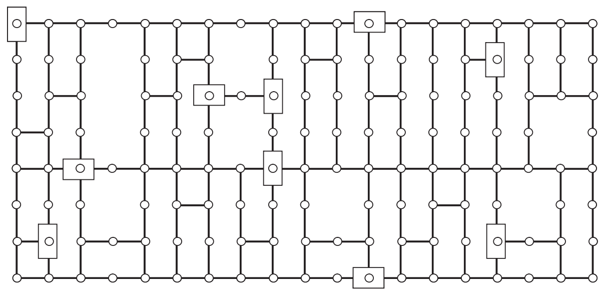
Fig. 2. An example of vehicle scheduling problem for 10 AGVs

To evaluate the performance of the proposed algorithm, the exact solution should be derived. However, branch and bound method cannot derive exact optimal solution for practical size of problems in realistic computation time. Therefore, the performance of the proposed algorithm is compared with a conventional algorithm [Nishi and Maeno 2007]. The conventional algorithm is based on the penalty optimization method where the infeasibility of violating firing condition is included in the objective function. The weighting factor for penalty function is gradually increased at each iteration of solving subproblem. For the conventional penalty function method, the solution of the subproblem cannot provide a lower bound of the original problem. Thus, the optimality of solution derived by the conventional method cannot be evaluated.

The transportation system consists of 143 nodes and 190 edges. The number of places, transitions are $|P| = mn + n$, $|T| = 2em$ where m is the number of AGVs, n is the number of nodes and e is the number of edges for the layout model. 15 cases of the problems are created when the number of AGVs is changed from 4 to 15. For 15 AGVs problems, $|P| = 2,288$, $|T| = 5,700$. For the decomposed subnets derived by using the proposed decomposition scheme, the subnets are evaluated as $|P_{u_i} \cup P_{Ru_i}| = n + n = 286$, $|T_{u_i}| = 2e = 380$. Therefore, the proposed decomposition method can reduce the state explosion problem caused by the increase of the size of Petri nets.

For each case of the problem, five problems are solved for randomly generated requests. The time horizon is $N_t = 60$, $\alpha = 1$. The average of total computation time, average duality gap, performance ratio are summarized in Table 1. The duality gap (DGAP) is calculated by DGAP [%] $= (UB - LB)/LB \times 100$.

The program is coded as Visual C++ (R) 6.0. A personal computer with Pentium IV 3GHz with 512 Mbyte memory, windows XP operating system is used for computation. The parameters for the proposed method are set up as $N_t = 60$ and $\alpha = 1$. The results are summarized in Table 1.

Table 1. Comparison of performance between LR and penalty function method (PM)

	LR method		PM method	
$\|V\|$	CPU time	DGAP	CPU time	Solution ratio
4	0.09	0.67	0.06	1.00
6	0.44	1.36	0.07	1.00
8	0.68	0.99	0.13	1.00
10	1.69	0.91	0.20	1.00
12	4.72	0.56	0.23	0.95
14	6.89	0.52	0.31	0.95
15	12.81	0.79	0.37	0.97

(CPU time: [sec.]. DGAP: average gap: [%])

The average computation times for both methods are increasing with respect to the increase of the number of AGVs because the routing problems with a large number of AGVs within a pre-defined number of nodes are increasingly difficult. The average performance of the proposed method is the same as that of the penalty function method when the number of AGVs is 4 to 10. However, for 10 to 15 AGVs problems, the average performance of the proposed method (LR) is better than that of the penalty function method (PM) even though the computation time for the proposed method is more than 10 times larger than that of the penalty function method. This is because constructing a feasible solution (Step 3 of the algorithm in Section 4.3) is added at each iteration of the algorithm. The reason why the proposed algorithm can derive better solution than the penalty function method is that the penalty function method (PM) may converge to a bad local optimal solution when the weighting factor is extremely large. For large-sized problems, a large number of iterations are needed to derive a feasible solution for the penalty function method. The average duality gap for the proposed method is within 2% and the worst duality gap is 5.5% for all cases. Therefore, the proposed method can derive a near-optimal solution for AGVs routing problems.

6. CONCLUSION AND FUTURE WORKS

In this paper, Lagrangian relaxation method for solving AGV routing problems by decomposition of timed Petri nets has been proposed. The decomposition procedure and the formulation of Lagrangian function have been developed. The entire Petri nets are decomposed into several subnets by using the decomposition procedure. The main advantage of the methodology is that the proposed method can evaluate the optimality of performance by obtaining lower bounds by Lagrangian relaxation. The benefits of the decomposition scheme represented by timed Petri nets is that the proposed methodology can easily extend to AGV routing problems with various constraints such as loading, unloading, buffering, or coordination with material handling machines based on the proposed decomposition scheme. The further research is to study the computational complexity with the overall coordination algorithm and to apply the proposed method for dynamic environment for practical use.

REFERENCES

H. Darabi, M.A. Jafari, S.S. Manapure. Finite automata decomposition for flexible manufacturing systems control and scheduling, *IEEE Transactions on System, Man, and Cybernetics - Part C*, volume 33, pages 168–175, 2003.

M.L. Fisher, The Lagrangian Relaxation Method for Solving Integer Programming Problems, *Management Science*, volume 27-1, pages 1–18, 1981.

D.W. He, B. Strege, H. Tolle, A. Kusiak. Decomposition in automatic generation of Petri nets for manufacturing system control and engineering, *International Journal of Production Research*, volume 38, pages 1437–1457, 2000.

D.Y. Lee, F. DiCesare. Scheduling flexible manufacturing systems using Petri nets and heuristic search, *IEEE Transactions on Robotics and Automation*, volume 10, pages 123–132, 1994.

T. Nishi, M. Ando, M. Konishi. Distributed route planning for multiple mobile robots using an augmented Lagrangian decomposition technique, *IEEE Transactions on Robotics*, volume 21, pages 1191–1200, 2005.

T. Nishi, R. Maeno. Petri Net Modeling and Decomposition Method for Solving Scheduling Problems, *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, pages 262–271, 2007.

S.H. Teng and J. Zhang. A Petri net-based decomposition approach in modelling of manufacturing systems, International Journal of Production Research, volume 31, pages 1423–1439, 1993.

T. Watanabe, Y. Mizobata, K. Onaga, Time Complexity of Legal Firing Sequence and Related Problems of Petri Nets, The Transactions of the IEICE, volume E-72, pages 1400–1409, 1989.