International Symposium on Scheduling 2011
July 2-4, 2011, Osaka, Japan
4A1

PETRI NET DECOMPOSITION APPROACH FOR AGV SYSTEMS TO MINIMIZE THE DEVIATION OF DELIVERY TIME AND TOTAL TRANSPORTATION TIME

Shuhei Eda and Tatsushi Nishi
Mathematical Science for Social Systems
Graduate School of Engineering Science, Osaka University
1-3 Machikaneyama-cho, Toyonaka, Osaka 560-8531, Japan
eda@inulab.sys.es.osaka-u.ac.jp, nishi@sys.es.osaka-u.ac.jp

Toshisada Mariyama, Satomi Kataoka, Kazuya Shoda and Katsuhiko Matsumura
Daifuku Institute of Technology and Training Co., LTD.,
1225 Nakazaiji, Hino-cho, Gamo-gun, Shiga 529-1692, Japan
{toshisada_mariyama, satomi_kataoka, kazuya_shoda, katsuhiko_matsumura}@ha.daifuku.co.jp

Abstract

In this paper, we propose an optimization algorithm for the routing of AGV systems to achieve the minimization of the deviation of delivery time and the minimization of the total transportation time. The dispatching and conflict-free routing problem for AGVs is represented as an optimal firing sequence problem for Petri Net. A Petri Net decomposition approach is applied to solve the multi-objective optimization problem efficiently. The effectiveness of the proposed method is compared with that of the conventional method. Computational results show the effectiveness of the proposed method.

Keywords: Petri Net, AGV routing, multi-objective optimization, conflict-free routing, decomposition

1. Introduction

Multiple automated guided vehicles (AGVs) are widely used for transportation systems in semiconductor fabrication bays, container terminals, and production systems. In the preceding study, it has been required to derive collision-free routing to minimize the total transportation time with a shorter computation time(Dotoli and Fanti 2004)(Liao, Jeng, Zhou 2004). However, the delivery time, which is the time from a loading point to an unloading point, is often deviated from the scheduled time in order to avoid the deadlock or collision of vehicles. Considering the simplicity of making the master production schedule, the delivery time should be as equal as possible. Hence, it is difficult to estimate the delivery time to avoid the congestion of vehicles. In this paper, we propose an optimization algorithm for the routing of AGV systems to achieve the minimization of the deviation of delivery time and the minimization of the total transportation time. The transportation system is treated as a discrete system model and it is represented by the timed Petri Net. We describe the Petri Net decomposition algorithm(Tanaka, Nishi, Inuiguchi 2009) for solving this problem.

The rest of the paper consists of the following sections. Section 2 states the problem definition. Section 3 describes the modeling for task assignment and AGV routing problems. Section 4 presents the Petri Net decomposition approach and an efficient algorithm for solving subproblems. Computational experiments are demonstrated in Section 5. Finally, Section 6 concludes the paper.

2. Problem Statement

The AGVs transportation system is described by a set of nodes and edges. Each node represents a place where an AGV can stop or turn. Each edge represents a unidirectional or bidirectional lane. The following conditions are assumed for the traveling of AGVs.

- Two or more than two AGVs cannot travel on a node (resource constraint on each node).
- Two or more than two AGVs cannot travel on an edge (resource constraint on each edge).
- The loading time and unloading time is 1 time period. The turning time is negligible compared with one traveling time.

The following conditions are assumed for task assignments.

- A set of tasks are given at the same time.
- Each task denotes the request for transportation from the starting node to the ending node for loading or unloading.
- More than one task can be allocated to each AGV. There can be free AGVs whith no task.

In this study, there are some objective functions to optimize. Assume that there are p objective functions $f_1(x), \ldots, f_p(x)$. For the multi-objective optimization problems, x' is a Pareto optimal if there is no x satisfying $f_k(x) \leq f_k(x') \ \forall k(1 \leq k \leq p)$. The problem is composed of two objective functions, J_1 and J_2. J_1 is for

the minimization of the deviation of delivery time. J_2 is for the minimization of total transportation time. To obtain the set of Pareto optimal solution, the weighting parameter method is used. It enables us to output a Pareto set by setting the appropriate parameter μ to each objective function. The weighted sum of the objective function J is minimized.

3. Petri Net Modeling for AGV systems

3.1 Modeling for Task Assignnment and Routing Problem for AGV Systems

For modeling AGV system, place p_{v_i,s_j} represents the existence of AGV v_i on node s_j. The firing of each transition t_{v_i,s_a,s_b} represents the traveling for AGV v_i on a lane between the two adjacent nodes s_a and s_b. The resource place $p_{0,s_j}(1 \leq j \leq n)$ is introduced to avoid collision on each node and on each edge. A task can be modeled as follows. Place $p_{u_i,0}$ denotes the condition that task u_i is not assigned to any AGVs. Place p_{u_i,v_j} represents that condition that AGV v_j has not completed the traveling to a loading point. Place p_{u_i,v_j+1} represents that AGV v_j is transporting and the traveling to an unloading point is not completed for task u_i. $p_{u_i,1}$ denotes the condition that task u_i is completed. $t_{u_i,v_j,0}$ indicates the event that task u_i is assigned to AGV v_j, and $t_{u_i,v_j,1}$ means the event that AGV v_j executes its task u_i (loading or unloading products). Fig. 1 shows the Petri Net model for 2 tasks and 2 AGVs.

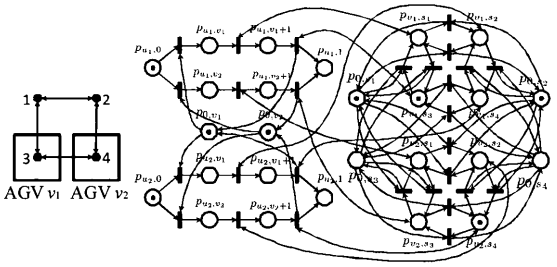

Fig. 1 Petri Net model for 2 Tasks and 2 AGVs routing problem

3.2 Optimal Firing Sequence Problem and Multi-objective optimization problem

The problem modeled by Petri Net is treated as a optimal firing sequence problem. Here, we define the optimal firing sequence problem. The multi-objective optimization problem is formulated. The multi-objective optimal firing sequence problem for Petri Net is defined as follows. Given $PN = (P, T, w, M_0)$, final marking $M_f : P \rightarrow (\mathbb{Z}^+ \cup \{*\})$ ($*$ is a don't care term), total time horizon $N_t \in \mathbb{N}$, and the objective fuctions $J_1, \cdots, J_p$, the optimal firing sequence problem is to find a feasibe set of firing vectors $(r_0, r_1, \ldots, r_{N_t-1}) \in (\{0,1\}^{|T|})^{N_t}$ satisfying $M_{N_t} = M_f$ to minimize $J_1, \cdots, J_p$. In our problem J_1 is the sum of the total traveling time, and J_2 is the deviation of delivery time from an initial position to the destination.

3.3 Formulation of Optimal Firing Sequence Problem

$P_{u_i} = \{p_{u_i,0}, p_{u_i,1}, p_{u_i,v_j}, p_{u_i,v_j+1}\}(1 \leq j \leq m; 1 \leq i \leq l)$ is the set of places representing the assignment of task u_i. Let M_k be the marking at time k, $M_{u_i}^k \in \mathbb{Z}^{+|P_{u_i}|}$ be the column vector for $p \in P_{u_i}$. $M_{pick}^{u_i}$ is a final marking for the loading of task u_i where $M_{pick}^{u_i}(p_{u_i,v_j+1}) = 1$. $M_f^{u_i}$ is a final marking for the unloading of task u_i where $M_f^{u_i}(p_{u_i,1}) = 1$. The other places for $M_{pick}^{u_i}$, $M_f^{u_i}$ are don't care term $(*)$. $\delta_{u_i,k} \in \{0,1\}$ takes the value of 1 if unloading of the task u_i is not finished, if completed 0. $\delta_{u_i,k}^{pick} \in \{0,1\}$ takes the value of 1 if loading of the task u_i is not finished , if completed 0. The delivery time for u_i is $\sum_{k=0}^{N_t}(\delta_{u_i,k} - \delta_{u_i,k}^{pick})$, and the total traveling time is $\sum_{k=0}^{N_t} \delta_{u_i,k}$. Therefore, the objective function with respect to the minimization of the deviation of delivery time is J_1, and the minimization of the total traveling time is J_2. These functions are formulated as follows.

$$J_1 = \sum_{i=1}^{l} |\sum_{k=0}^{N_t}(\delta_{u_i,k} - \delta_{u_i,k}^{pick}) - \bar{v}| \quad (1)$$

$$J_2 = \sum_{i=1}^{l}\sum_{k=0}^{N_t} \delta_{u_i,k} \quad (2)$$

$\bar{v}$ in J_1 is an average of the delivery time. And it is written as

$$\bar{v} = \frac{1}{l}\sum_{i=1}^{l}\sum_{k=0}^{N_t}(\delta_{u_i,k} - \delta_{u_i,k}^{pick}) \quad (3)$$

$\delta_{u_i,k}, \delta_{u_i,k}^{pick}$ are formulated as

$$\delta_{u_i,k} = \begin{cases} 1 & (M_k^{u_i} \neq M_f^{u_i}) \wedge (\delta_{u_i,k-1} = 1) \\ 0 & (M_k^{u_i} = M_f^{u_i}) \vee (\delta_{u_i,k-1} = 0) \end{cases} \quad (4)$$

$$\delta_{u_i,k}^{pick} = \begin{cases} 1 & ((M_{pick}^{u_i})^{\mathrm{T}} M_k^{u_i} = 0) \wedge (\delta_{u_i,k-1}^{pick} = 1) \\ 0 & ((M_{pick}^{u_i})^{\mathrm{T}} M_k^{u_i} = 1) \vee (\delta_{u_i,k-1}^{pick} = 0) \end{cases} \quad (5)$$

If the task is not assigned to the AGV in the time horizon, J_1 can be minimized. However, this solution is impractical. Hence, (6) should be added into the constraints.

$$\sum_{i=1}^{l}(M_f^{u_i})^{\mathrm{T}} M_{N_t}^{u_i} = l \quad (6)$$

Moreover, the firing condition and state equation are formulated as

$$M_k - (A^-)^{\mathrm{T}} r_k \geq 0 \quad (7)$$

$$M_{K+1} = M_k + (A^+ - A^-)^{\mathrm{T}} r_k \quad (8)$$

The optimal firing sequence problem can be formulated as an MILP(Mixed Integer Linear Programming) problem. The formulations are as follows:

$$\min_{\{r_k\}} J \tag{9}$$

$$J = \mu J_1 + (1-\mu) J_2 \tag{10}$$

s. t. (1), (2), (3), (4), (5), (6), (7), (8)

$$0 \leq \mu < 1 \tag{11}$$

$$\delta_{u_i,k},\ \delta^{pick}_{u_i,k} \in \{0,1\} \tag{12}$$

4. Petri Net Decomposition Method

4.1 Decomposition Approach

The optimal firing sequence problem is computationally extensive to solve when the number of tasks and AGVs are increased. Therefore, the entire Petri Net is decomposed into several subnets by the Petri Net decomposition approach. The Petri Net is decomposable if the following two conditions are satisfied.

- The total objective function J is additive for each subnet $u_i (1 \leq i \leq \mathcal{M})$, which means that $J = \sum_{i=1}^{\mathcal{M}} J_{u_i}$. Each value of J_{u_i} should depend on only index i.
- The final marking is not defined for the duplicated place P_R, that is, the final marking for place $p \in P_R$ is do not care term $(*)$.

The entire Petri Net of task assignment and routing problems for l tasks and m AGVs can be decomposed into independent subnets if the several places are duplicated. The l independent subnets for the assignment of task u_i, and m independent subnets for the routing of AGVs are generated. $\mathcal{M}$ is a sum of l and m, subnet u_i $(1 \leq i \leq l)$ is for the task, u_i $((l+1) \leq i \leq \mathcal{M})$ is for the AGV. The transition set T is decomposed into subsets , T_{u_i} as (13) where $A \sqcup B$ denotes the disjoint union of A and B for any sets.

$$T = T_{u_1} \sqcup T_{u_2} \sqcup \cdots \sqcup T_{u_m} \tag{13}$$

The place set is decomposed into the subsets P_{u_i} and P_R by

$$P = P_{u_1} \sqcup P_{u_2} \sqcup \cdots \sqcup P_{u_m} \sqcup P_R \tag{14}$$

where P_{u_i} satisfies (15), P_R satisfies (16), $OUT(p)$ is the set of the output transitions for place p and $IN(p)$ is the set of the input transitions for place p.

$$P_{u_i} = \{p \mid IN(p) \subseteq T_{u_i}, OUT(p) \subseteq T_{u_i}\} \tag{15}$$

$$P_R = P \setminus (P_{u_1} \sqcup P_{u_2} \sqcup \cdots \sqcup P_{u_m}) \tag{16}$$

The firing condition and the state equation for PN are derived. $r^{u_i}_k \in \{0,1\}^{|T_{u_i}|}$ is a column vector comprising $r_k(t)$ for $t \in T_{u_i}$. $M^{u_i}_k \in \mathbb{Z}^{+|P_{u_i}|} and M^R_k \in \mathbb{Z}^{+|P_R|}$ is a column vector comprising $M_k(p)$ for $p \in P_{u_i} and p \in P_R$. The firing condition for the PN is described as

$$M^{u_i}_k - (A^-_{u_i})^{\mathrm{T}} \geq 0 \quad (1 \leq i \leq \mathcal{M}) \tag{17}$$

$$M^R_k - (B^-)^{\mathrm{T}} r_k \geq 0 \tag{18}$$

by an appropriate integer matrix $A^+_{u_i} \in \mathbb{Z}^{+|T_{u_i}| \times |P_{u_i}|}$, $A^-_{u_i} \in \mathbb{Z}^{+|T_{u_i}| \times |P_{u_i}|}$ and

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

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

The state equation for the PN can be expressed by

$$M^{u_i}_{k+1} = M^{u_i}_k + (A^+_{u_i} - A^-_{u_i})^{\mathrm{T}} r^{u_i}_k \tag{19}$$

$$M^R_{k+1} = M^R_k + (B^+ - B^-)^{\mathrm{T}} r_k \tag{20}$$

From this approach for the places and transitions, PN is decomposed into the subnets $PN^{u_i} = (P_{u_i} \cup P_{R_{u_i}}, T_{u_i}, w^{u_i}, M^{u_i}_0)$ where $P_{R_{u_i}}$ is a set of places which corresponds to P_R. The marking of the subnet PN^{u_i} is defined as $M^{u_i}_k : (P_{u_i} \cup P_{R_{u_i}}) \to \mathbb{Z}^+$. $M^{R_{u_i}}_k \in \mathbb{Z}^{+|P_{R_{u_i}}|}$ is a column vector $M^{u_i}_k(p)$ involved in $p \in P_{R_{u_i}}$. The firing condition and state equation for PN^{u_i} can be written with (17), (19), (21), and (22).

$$M^{R_{u_i}}_k - (B^-_{u_i})^{\mathrm{T}} r^{u_i}_k \geq 0 \tag{21}$$

$$M^{R_{u_i}}_{k+1} = M^{R_{u_i}}_k + (B^+_{u_i} - B^-_{u_i})^{\mathrm{T}} r^{u_i}_k \tag{22}$$

Since the final marking for PN is do not care term $(*)$ regarding the set of P_R, the final marking for PN^{u_i} is formulated as

$$M^{u_i}_f(p) = M_f(p) \ (\forall p \in P_{u_i}) \tag{23}$$

The initial marking is formulated in the same way.

$$M^{u_i}_0(p) = M_0(p) \ (\forall p \in P_{u_i}) \tag{24}$$

Let us introduce an example of the application of the Petri Net decomposition approach. If this approach is applied to Fig. 1, it is decomposed into 4 subnets as shown in Fig. 2. Since each AGV subnet u_i $(l+1 \leq i \leq \mathcal{M})$ does not have

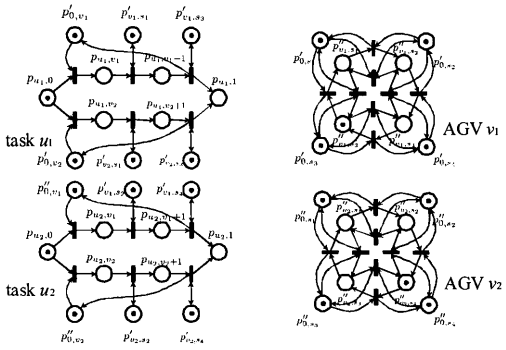

Fig. 2 Decomposed subnets for 2 Tasks and 2 AGVs routing problem

the final marking, $\delta_{u_i,k} = 0$ $(l+1 \leq i \leq \mathcal{M})$ is always true. In this case, each AGV travels wastefully. To avoid useless traveling for each AGV, the objective function for

each AGV subnet is defined as the distance from the current position. The objective function J_{u_i} is written by

$$J_{u_i} = \sum_{k=0}^{N_t} \delta'_{u_i,k} \ (l+1 \leq i \leq \mathcal{M}) \tag{25}$$

where $\delta'_{u_i,k}$ is a function which satisfies

$$\delta'_{u_i,k} = \begin{cases} 1 & (M_k^{u_i} \neq M_{k-1}^{u_i}) \\ 0 & (M_k^{u_i} = M_{k-1}^{u_i}) \end{cases} \ (M_f^{u_i} \in \{*\}^{P_{u_i}}) \tag{26}$$

4.2 Approximation of the Objective Function

Each task subnet u_i $(l+1 \leq i \leq \mathcal{M})$ has the final marking $M_f^{u_i}(p), M_{pick}^{u_i}(p) \in \mathbb{Z}^+$ as described in section 3.3. The objective function for the task subnet is defined as:

$$J_{u_i} = \mu | \sum_{k=0}^{N_t} (\delta_{u_i,k} - \delta_{u_i,k}^{pick}) - \bar{v}| + (1-\mu) \sum_{k=0}^{N_t} \delta_{u_i,k} \tag{27}$$

where μ is a weighting parameter. However, J_{u_i} has the average value of $\bar{v}$. The value of $\bar{v}$ cannot be computed by the optimization of each subnet u_i by the Petri Net decomposition approach. It does not hold the decomposable condition. Therefore, the average $\bar{v}$ is modified to a constant value $D \in Z^+$ and the value is estimated during the search of the Petri Net decomposition approach. (1) is reformulated as

$$J_1 = \sum_{i=1}^{l} | \sum_{k=0}^{N_t} (\delta_{u_i,k} - \delta_{u_i,k}^{pick}) - D| \tag{28}$$

The setting of D is explained in section 4.3. Due to the modification, the decomposable condition can be satisfied, and the objective function for the task subnet is defined as follows:

$$J_{u_i} = \mu | \sum_{k=0}^{N_t} (\delta_{u_i,k} - \delta_{u_i,k}^{pick}) - D| + (1-\mu) \sum_{k=0}^{N_t} \delta_{u_i,k} \tag{29}$$

By using this method, the objective functions to solve are (28) and (29).

4.3 Optimization Algorithm

The algorithm of the Petri Net decomposition method is described as follows.

STEP 1 Initialization

The initial value of D is set to the average estimated delivery time for all the tasks.

STEP 2 Initial Optimization

The number of iterations is $N := 1$. The subproblems formulated as (30) for subnet $u_j (j = 1, 2, \ldots, \mathcal{M})$ is solved by the algorithm explained in section 4.4. The optimal solution for each subproblem is regarded as a tentative solution.

$$\min_{\{r_k^{u_j}\}} \quad J_{u_j} \tag{30}$$
$$\text{subject to} \quad (17),\ (19),\ (21),\ (22)$$

STEP 3 Evaluation of Convergence

If the tentative solution $\hat{r}_k = [(\hat{r}_k^{u_1})^{\mathrm{T}}, (\hat{r}_k^{u_2})^{\mathrm{T}}, \ldots, (\hat{r}_k^{u_m})^{\mathrm{T}}]^{\mathrm{T}}$ satisfies (18), and the derived solution has not been updated from a previous solution, the tentative solution is regarded as a feasible and near optimal solution and go to STEP 7. Otherwise go to STEP 4

STEP 4 Re-optimization

The subnet for re-optimization is selected sequentially from u_i $(i \neq \mathcal{M})$ to u_{i+1} $(i \neq \mathcal{M})$ or $u_{\mathcal{M}}$ to u_1 if $i = m$. The re-optimization for subnet u_j is executed on the condition that the tentative solutions $\hat{r}_k^{u_i}$ $(u_i \neq u_j)$ derived at other subnets are constant value. The objective function for the re-optimization step is formulated as (31), where $\omega_{p,u_j}^{(N)}$ is the weighting factor for violating firing constraints with other subnets at Nth iteration.

$$\min_{\{r_k^{u_j}\}} \left(J_{u_j} + \sum_{k=0}^{N_t} \sum_{p \in P_{R_{u_i}}} \omega_{p,u_j}^{(N)} \alpha_{u_j,p,k}(r_k^{u_j}) \right) \tag{31}$$
$$\text{s. t.} \quad (17),\ (19),\ (21),\ (22)$$

The penalty function $\alpha_{u_j,p,k}(r_k^{u_j})$ is the number of tokens required to satisfy the firing condition by the firing vector $r_k^{u_j}$ at place $p \in P_R$ in time period k. Note that $\alpha_{u_j,p,k}(r_k^{u_j})$ $(l+1 \leq j \leq \mathcal{M})$ is the sum of $\alpha_{u_j,p,k-1}(r_k^{u_j})$ and the number of tokens required to be feasible at k.

STEP 5 Increasing Weighting Parameter for Penalty Function

The weighting factor for penalty function is updated by (32) and (33) using the derived solution $r_k^{u_j}$ and tentative solutions $\hat{r}_k^{u_i}$ $(i \neq j)$ for other subnets. $\Delta\omega$ is the parameter for updating weighting factors for the penalty function.

$$\omega_{p,u_j}^{(N+1)} = \omega_{p,u_j}^{(N)} + \Delta\omega \sum_{k=0}^{N_t} \alpha_{u_j,p,k}(r_k^{u_j}) \tag{32}$$

$$\omega_{p,u_i}^{(N+1)} = \omega_{p,u_i}^{(N)} \quad (i \neq j) \tag{33}$$

STEP 6 Update of Tentative Solution

The derived solution $r_k^{u_j}$ at Step 4 is regarded as a tentative solution. Update $\hat{r}_k^{u_j} := r_k^{u_j}$, then $N := N+1$, and return to Step 3.

STEP 7 Evaluation

Let c $(0 \leq c < C)$ define $c \in Z^+$ where C is a parameter involved in completing the algorithm. If D is not updated from the value of STEP 1, $J = J', c = 0$, $D := D + 1$, and return to STEP 2. Otherwise, compare J with J', and if $J > J', c = 0$, $J = J', c = 0$,

$D := D + 1$ and return to STEP 2. When $J \leq J'$, c is defined as c+1. Then, if $c < C, D := D + 1$ and return to STEP 2. If $c = C$, output J, and the algorithm is finished.

4.4 Solving Subproblems

The subproblems (30) and (31) are an integer programming problem which is also difficult to solve for integer programming techniques. The subproblem can be solved by Dijkstra's algorithm. Let $\mathcal{P}^j (0 \leq j \leq l)$ denote the set of states. h_k is a function which takes the value of 1 if the marking has reached the final marking, if not zero.

$$\mathcal{P}^j = \{\sigma |\ \sigma = (M_k^{u_j}, k, h_k);\ M_k^{u_j} \in R(PN^{u_j}, k);\ 0 \leq k \leq N_t;\ h_k \in \{0,1\};\ h_k = 0 \vee (M_k^{u_j} = M_f^{u_j})\} \quad (34)$$

If the marking $M_k^{u_j}$ is the same as the final marking, two types of states, $(M_k^{u_j}, k, 1)$, and $(M_k^{u_j}, k, 0)$ are generated. $(M_k^{u_j}, k, 1)$ is the state where the marking is the final marking and all of the transitions are completed. On the other hand, $(M_k^{u_j}, k, 0)$ is the state where the marking is the final marking but the transitions are not completed. Once the state $(M_k^{u_j}, k, 1)$ is generated, no other states can be reachable except $(M_{k+1}^{u_j}, k+1, 1)$ $(M_{k+1}^{u_j} = M_f^{u_j})$ from the state. The cost function $d_{u_j,a_k,a_{k+1}} (1 \leq j \leq l)$ from a state $a_k = (M_k^{u_j}, k, h_k) \in \mathcal{P}^j$ to a state $a_{k+1} = (M_{k+1}^{u_j}, k+1, h_{k+1}) \in \mathcal{P}^j$ takes the value as follows.

$$d_{u_j,a_k,a_{k+1}} = \begin{cases} \mu |\sum_{n=0}^{k+1} (\delta_{u_{j,n}} - \delta_{u_{j,n}}^{pick}) - D| + (1-\mu) \sum_{n=0}^{k+1} \delta_{u_j,n} \\ + \sum_{p \in P_{R_{u_j}}} \omega_{p,u_j}^{(N)} \alpha_{u_j,p,k}(r_k^{u_j}) \quad ((h_k = 0) \wedge (h_{k+1} = 1)) \\ 0 \quad \text{(otherwise)} \end{cases}$$

$d_{u_j,a_k,a_{k+1}} \geq 0$ must be satisfied in Dijkstra's algorithm. On the other hand, subnet u_j $(l+1 \leq j \leq \mathcal{M})$ does not have a final marking. Therefore, let $\mathcal{P}^j$ and $d_{u_j,a_k,a_{k+1}}$ $(l+1 \leq j \leq \mathcal{M})$ be defined as

$$\mathcal{P}^j = \{\sigma |\ \sigma = (M_k^{u_j}, k);\ 0 \leq k \leq N_t\} \quad (35)$$

$$d_{u_j,a_k,a_{k+1}} = \delta'_{u_j,k} + \sum_{p \in P_{R_{u_j}}} \omega_{p,u_j}^{(N)} \alpha_{u_j,p,k}(r_k^{u_j}) \quad (36)$$

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 $\mathcal{P}^j$ is the set of nodes and $d_{u_j,a_k,a_{k+1}}$ is the length of nodes.

Table 1 Comparison of objective function for 10 Tasks and 10 AGVs problems

CASE	J_1		J_2	
	Conv.	Proposed	Conv.	Proposed
1	0	0	319	485
2	3.2	0	341	494
3	4.8	0	371	513
4	1.8	0	334	524
5	9.6	0	376	554
Avg.	3.88	0	348.2	514

5. Computational experiments

5.1 Comparison with the conventional method

In this section, the computational experiments are conducted to compare the performance between the proposed method and the conventional method. The proposed method aims for minimizing the deviation of delivery time and the conventional method aims to minimize the total transportation time. In the experiments, all tasks have the same distance (20 times) although each task has a different loading and unloading point. This is because it is easy to verify how the proposed method can effectively minimize the deviation of delivery time compared to the conventional method. The number of tasks and AGVs are 10, and the distance between loading point and unloading point is 20. 5 cases are executed where the initial position of AGVs, the loading point and the unloading point of tasks are different with each case. The parameters are set $\Delta w = 0.3, C = 1, N_t = 100$ for all problems. The transportation system is Fig. 3. Intel(R) Core(TM) i7 CPU 860 @2.80GHz with 3.71GB memory is used for computation.

Table 1 presents the value of the objective function $J_1 and J_2$ between the proposed method (Proposed) and conventional method (Conv.). Table 2 presents computation time (CPU) and the maximum delivery time (Max DT).

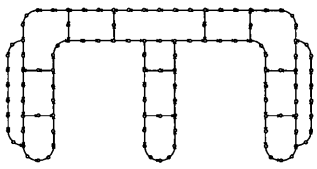

Fig. 3 Transportation system with 115 nodes and 128 edges

Table 1 shows that the value of J_2 for the conventional method is smaller than that for the proposed method. However, J_1 for the proposed method is smaller than that for the conventional method. The results imply that the proposed method can generate routes that can minimize the deviation of delivery time. Table 2 shows that the average CPU and the maximum delivery time of the proposed method is shorter than those of the conventional method. In other

Table 2 Comparison of CPU time and maximum delivery time for 10 Tasks and 10 AGVs problems

CASE	CPU [s]		Max. DT	
	Conv.	Proposed	Conv.	Proposed
1	530.38	220.03	21	21
2	304.67	130.39	22	21
3	451.26	177.5	23	22
4	289.38	168.45	22	22
5	555.11	153.95	25	21
Avg.	426.16	170.06	22.6	21.4

words, the proposed method can generate the solutions to avoid the congestion in a shorter CPU compared to the conventional method. The above discussion indicates that the proposed method can minimize the deviation of the delivery time. And the proposed method has the effect to minimize the maximum delivery time. It is not difficult to estimate the delivery time when the proposed method is executed. Therefore, it can decrease wasteful time and cost.

5.2 Comparison with nearest neighborhood method

The performance of the proposed method is compared with a heuristic method called nearest neighborhood method (NN method). The NN method is constructed so that the tasks are assigned to the AGV which has the least estimated traveling time to complete the task. The estimated traveling time to complete the task E_{v_j} can be computed by

$$E_{v_j} = \max\{\tau_{free}^{v_j}, \tau_{now}\} - \tau_{now} + \tau_{min}^{v_j} \qquad (37)$$

where $\tau_{free}^{v_j}$ is the estimated completion time for AGV v_j to complete all the tasks without considering collision with other AGVs. τ_{now} is the current time, and $\tau_{min}^{v_j}$ is the minimum traveling time from the ending node for the last task executed by AGV v_j to the starting node of newly generated task without considering collision with other AGVs. In our problem definition, $\tau_{now} = 0$ because all tasks are generated at time 0. The proposed method and the NN method are constructed to minimize the deviation of delivery time. All tasks have the same distance (10 unit time). The number of tasks and AGVs are 15. Five cases are executed with different initial positions of AGVs and different loading and unloading points of tasks in each case. Table 3 presents the value of the objective functions J_1 and J_2 for the proposed method (Proposed) and the NN method (NN). Table 4 presents the computation time (CPU) and the maximum delivery time (Max DT). Table 3 shows that the value of J_2 for the NN method is smaller than that for the proposed method. However, J_1 for the proposed method is smaller than that for the NN method. The results demonstrate that the proposed method can generate solutions that can minimize the deviation of delivery time. Furthermore, the maximum delivery time for the proposed method is shorter than that for the NN method from Table 4. This implies that the proposed method

Table 3 Comparison of objective function for 10 Tasks and 10 AGVs problems

CASE	J_1		J_2	
	NN	Proposed	NN	Proposed
1	5.60	0.00	433	552
2	3.73	2.00	394	568
3	0.00	0.00	456	506
4	13.87	0.00	308	596
5	5.60	0.00	599	545
Avg.	5.76	0.40	438	553.4

Table 4 Comparison of CPU time and maximum delivery time for 10 Tasks and 10 AGVs problems

CASE	CPU [s]		Max. DT	
	NN	Proposed	NN	Proposed
1	114.31	3235.14	14	11
2	247.41	1041.44	14	13
3	1730.98	1039.48	11	12
4	183.06	960.97	18	12
5	350.77	2251.22	14	11
Avg.	525.31	1705.65	14.2	11.8

can minimize the deviation of the delivery time although the computation time is longer than that of the NN method.

6. Conclusions and Future Works

In this paper, we have presented the Petri Net decomposition method for the routing of AGV systems to achieve the minimization of the deviation of the delivery time and the minimization of the total transportation time. The computational experiments demonstrate the effectiveness of the proposed method. The maximum delivery time of the solution for the proposed method is shorter than that for the NN method. One of our future work is to simulate more practical problems with large scale AGVs systems.

References

Dotoli, M. and Fanti, M. P. (2004). Coloured timed Petri net model for real-time control of automated guided vehicle system, *International Journal of Production Research*, Vol. 42, No. 9, pp. 1787–1814.

Liao, D.Y., Jeng, M., Zhou, M. (2004). Petri net modeling and Lagrangian relaxation approach to vehicle scheduling in 300mm semiconductor manufacturing, *Proceedings of International Conference on Robotics and Automation*, pp. 5301–5306.

Tanaka, Y., T. Nishi, M. Inuiguchi. (2009). Petri Net decomposition method for simultaneous optimization of task assignment and routing for AGVs, *Transactions of ISCIE*, Vol. 22, No. 5, pp. 191–198.