Distributed Control of Flexible Transfer System (FTS) Using Learning Automata

Toshio FUKUDA*1, Kosuke SEKIYAMA*2, Yoshiaki HASEBE3, Yasuhisa HASEGAWA*3 Susumu SHIBATA*4, Hironobu YAMAMOTO*4 and Yuji INADA*4

*1Center for Cooperative Research in Advanced Science and Technology, School of Engineering, Nagoya University, Furo-cho, Chikusa-ku, Nagoya 464-8603, JAPAN Phone: +81-52-789-4481, Fax: +81-52-789-3909 Email:{fukuda@mein,hasebe@robo.mein,yasuhisa@mein}.nagoya-u.ac.jp
*2Department of Mechano-Informatics and Systems, The Science University of Tokyo, Suwa College 5000-1 Toyohira, Chino, Nagano 391-0292 JAPAN Phone: +81-266-73-1201 ex. 335 Fax: +81-266-73-1230 Email: sekiyama@ss.suwa.sut.ac.jp
*4Mechatronics Technology Section Advanced Industrial Systems Technology Dept. Technology Research & Development Center Toyo Engineering Corporation 8-1, Akanehama 2-chome Narashino-shi, Chiba, 275-0024 JAPAN Email:{s-shibata,yamamoto,inada-y}@is.toyo-eng.co.jp Tel: +81-47-454-1794, Fax: +81-47-454-163

Abstract
This paper proposes a flexible transfer system (FTS) as one of the self-organizing manufacturing systems. The FTS is composed of autonomous robotic modules, which transfer a palette carrying an object. Through the self-organization of a multi-layered strategic vector field corresponding to a task, the FTS can generate quasi-optimal transfer path in fully distributed way. We apply the learning automata for path generation algorithm. Simulation is conducted to evaluate the basic performance of the system and the results show the feasibility of application. Also, the developed hardware is explained.

1. Introduction
The large-scale production systems, such as mobile production line and heavy industrial plants are still based on the centralized control. In the centralized control systems, optimization of the productivity or efficiency is expected in a rather stationary environment, while several problems have been pointed out with regard to fault-tolerance, flexibility, and expansion of the system configurations to cope with a dynamical change of work environment. These problems will become more conspicuous for further complicated and larger systems. Therefore, a distributed autonomous system with a capability of self-organization has been required, because it is superior to the centralized system in the features of load distribution and higher adaptability[1].

In this research, we propose Flexible Transfer System (FTS) as an example of the self-organizing production system, which is shown in fig.1. The FTS transfers a pallet on an autonomous robotic module, where an assembling part is put. Based on the local communication between the connected modules and local information processing, a successful transferring path is generated.

The configuration of modules can be changed according to a kind of task. For example, a situation can be considered where a part of module is broken down, or a configuration has to be modified for the maintenance of a module. Due to the self-organizing capability, the FTS can cope with the changes without suspending a function of the whole system. The purpose of self-organizing transferring path is attaining a quasi-optimized state, which means that it is not optimized in the sense of the centralized control based on the global information. Where, the central issue is how to control the direction of the self-organization for a quasi-optimized state. The conventional work on the distributed control and self-organizing production system is as follows. The FTS using GA realizes suitability for scheduling transportation program[2][3]. In Flat Transportation of
Micro system research, Micro Actuator responds to system order autonomously[4][5]. Mobile Robot makes the way around using Fuzzy Rule and Learning Automata[6].

In this paper, we present a distributed control method using the learning automata for the FTS. As a first stage of the work, we simulated a process of self-organization of the particular transferring path for an assembling task, and evaluated a basic performance of the system. The algorithm can be easily implemented to the developed experimental system as shown in fig.1.

2. Flexible Transfer System (FTS)
2.1 Configuration of the FTS

The FTS is composed of many autonomous transfer modules, which system configuration is depicted in fig.2. Any four sides of the module can be connected with another transfer module and the connected modules are communicable through Ethernet. Each module recognizes an existence of the palette with equipped proximity switches, and it transfers a palette to the neighboring module driven by the Omni-wheels. The direction of transfer is restricted in the x-y direction, in order to guarantee that a module can correctly recognize the palette on it. Figure 3 illustrates the surface of the robotic module and the bottom of palette. The palette carries an object to be assembled. A data carrier chip is installed in the center of the bottom side of palette where the machining information of the object is written down. When the module recognizes a palette on it with proximity switches, it will read the task information of the object from the data carrier, and decides the direction to transfer the palette.

2.2 Basic Approach for the FTS Control

Each module has no global information such as a configuration of the whole modules, or the location of itself. What the module can obtain is the peripheral condition of itself through the communication. However, the FTS will generate a feasible path from random state, in the fully distributed way. The central idea of FTS is a self-organization of multi-layered strategic vector fields. Each module has an action strategy, which is defined as a vector value. Local interactions among the modules develop the global stochastic vector field, which corresponds to a particular machining task. Once a strategy filed is generated, the palette can reach to the desired machining site by referring the vector field wherever it may be placed. The concept is depicted in fig.5. For generation of the strategy field, we apply the learning automata, as described in the following section.

3. Distributed Control of FTS by Learning Automata
3.1 Learning Automata

In this section, we describe a brief introduction of the learning automata[7][8]. The learning automata are defined by a set of six variables of the real number, which indicates a set of input to the automaton as shown in fig.5. That is given as,

\[ (X, \Omega, A, \pi(t), F, G) \] (1)
Hence, $\Omega = \{a_1, a_2, \ldots, a_r\}$ is a state set of the automaton, and $A = \{a_1, a_2, \ldots, a_r\}$ is a set of the output from the automaton.

Also, $\pi(t) = (\pi_1(t), \pi_2(t), \ldots, \pi_n(t))$ is a state set of the automaton, and $\pi = (a_1, a_2, \ldots, a_r)$ is a set of the output from the automaton.

$G$ is an algorithm to update a state vector. If a response to the automaton is given by $r(t)$ for an action $a(t)$, $\pi(t)$ is updated based on the followings

$$\pi(t+1) = F(\pi(t), \alpha(t), X)$$  \hspace{1cm} (2)

$G$ is the output function which gives a correspondence between the state and the action, but here, it is sets of $r$ dimensional vectors, given by

$$g_i = (g_{i1}, g_{i2}, \ldots, g_{ir}) (i = 1, 2, \ldots, s)$$  \hspace{1cm} (3)

Then, an action $a_i$ is determined based on the stochastic vector $g_i$ for respective state $a_i$. For this automaton, environment is characterized by unknown probabilistic distribution $\theta_i$ on $X$, corresponding to $a_i$.

3.2 Self-Organizing Algorithm of Strategic Vector Field

3.2.1 Strategic Vector Field

We apply the learning automaton to the path organization of the FTS. Let be $p = (x, y)$ a position of the module, and $E$ the set of the position. Also, the peripheral condition of the module (existence of another palette) is defined by $Env(x, y)$, which can be recognized by the communication between the modules. The internal state of the module is represented as $h(x, y, s, t)$, where $s$ and $t$ indicates the kind of task, and time step respectively. The set of $h(x, y, s, t)$ gives a strategy vector field to determine where to transfer the palette. $Env(x, y)$ and $h(x, y, s, t)$ correspond to $\Omega$ and $\pi(t)$ in the section 3.1.

Now, we define the set of neighborhood of the module as follows,

$$p(i) = \{(x+1, y), (x, y-1), (x-1, y), (x, y+1)\}$$  \hspace{1cm} (4)

where, $i$ indicates an element of the set. The strategy vector of the module located at $p$ stochastically determines the direction of transfer of the palette, which is now expressed as follows,

$$h_p(s, t) = (b_0(t), b_1(t), b_2(t), b_3(t))^T$$  \hspace{1cm} (5)

3.2.2 Learning Algorithm of Strategy Vector

When a module successfully transfers the palette to the machining site of corresponding task, it will receive a constant reward from the machining site. However, since the FTS is fully distributed system, modules, which are not connected to the machining site, cannot explicitly know what is a correct behavior. Therefore, we introduce a reward variable $r_p(s, t)$ for each module corresponding to the kind of task $s$ ($A, B, C$). When a module $m[a]$ transfers a palette to the module $m[b]$, $m[a]$ will receive a reward from $m[b]$, which value is a constant ratio of $r_p(s, t)$ of $m[b]$. This procedure is expressed by[9],

$$r_p(s, t) = (1 - k) r_p(s, t) + k r_p(\text{trans})$$  \hspace{1cm} (6)

$k = \text{const}$

Through this procedure, a field of the reward is organized, where a module closer to the machining site will have a larger reward value. This implies that a gradient of the reward field is generated. Hence, we define a vector, which element is a relative value between the neighboring modules of position $p$. This is given by,

$$r = \frac{1}{r_p(s, t)} \left( r_p(s, t) \right)^T$$  \hspace{1cm} (7)

Using (6), the reward vector is defined as follows,

$$R_p(s, t) = (1 - a) r + \frac{a}{\sum_{i=1}^{4} r_p(s, t)}$$  \hspace{1cm} (8)

$a = \text{const}$

Where, the second term is four-dimensional vector of normalized random numbers. This is introduced to restrict initial fluctuations due to the excessive dependence to the reward values in the initial stage of the learning process. The reward vector is essential because it gives the direction of self-organization of the strategy vectors. Now, the strategy vector (5) is updated based on the following rule,

$$h_p(s, t) = F([l + IR_p(s, t)] h_p(s, t))$$  \hspace{1cm} (9)
Where, I is unit matrix of order 4, which is simply introduced to correct the dimension of vector. Also, F is an operator to normalize norm of the vector.

3.2.3 Collision Avoidance

Since the FTS transfers multiple palettes in a fully distributed way, there is a case where the designated module of transference is occupied by the other palette. Therefore, the occupation of the module should be confirmed before the transference, by way of the communication between the modules. According to the negotiation, the strategy vector has to be modified if needed, to avoid a collision. The occupation is checked as follows,

\[ \delta_{p(t)}(t) = \begin{cases} 1 & p_{(t)} : empty \\ 0 & p_{(t)} : occupied \end{cases} \]  

Based on (10),

\[ \delta_p(t) = (\delta_{p(t)}(t), \delta_{p(c_2)}(t), \delta_{p(c_3)}(t), \delta_{p(c_4)}(t)) \]  

\[ Env_p(t) = h\delta_p(t) \]  

By use of (12), the strategy vector is modified such that the module will not transfer the palette to the occupied module, which is given by

\[ h_p(s,t+1) = F(Env_p(t) - h_p(s,t+1)) \]  

The actual palette transference is performed based on (13).

4. Simulation Experiments

We exemplify the FTS with the proposed algorithm and evaluate the basic performance of the system. Figure 6 illustrates the supposed task environment in the simulation. From the entrance, a palette is thrown into the system. As is mentioned in section 2, the information of the object is contained as to what kind of task should be executed in the data carrier of the palette. The marked A, B, and C in fig.6 indicates the machining site for such a particular task. In this example, the object on the palette is needed three kinds of task. If a machining schedule is designated as \( A \rightarrow B \rightarrow C \) in the data carrier, the palette has to be transferred through a feasible path. When the task is completed, the palette will exit from the machining site (Exit).

4.1 Generation of Strategic Vector Field

We observe how the strategic vector will evolve in the model. For this simulation, following assumptions are set,

1. 5 × 5 modules
2. 4 directions for the transfer
3. Synchronized transfer
4. 3 machining sites
5. 1 time step for a palette transfer

Hence, the synchronized transfer (3) as a whole system is basically not required, but it is assumed for just simplification and for the future plan of fusion with another algorithm, which is undergoing work. Each module functions as a learning automaton, and self-organizes a strategic vector field for a feasible transfer path according to the placement of the machining sites. Each module will get the task information of the object on the palette. According to the task information, corresponding strategic field is referred. At the initial condition, the strategic vector field is randomized. So, the motion of the palette exhibits a random walk. However, as the learning process proceed, the vector field converges to an effective state. Figure 6 shows an example of the generated path of the palette, which corresponds to fig.6. From the left side of entrance, a palette is transferred into the system, where three kinds of machining tasks are planned but its order is not strictly defined. The palette will exit from a machining site (Exit).
4.2 Estimation of the Basic Performance

We evaluate the basic performance of the FTS. Apparently, there will be a limitation in the number of palettes that the FTS can deal with at the same time. For typical two cases, simulations are performed. In both cases, the task is set to visit two machining sites of three.

4.2.1 The case where the number of palette is constant

Firstly, let us consider the case where the number of palettes is constant in the FTS. This corresponds to the situation that only the object is thrown to and sent out from the system, but palettes just go around on the modules carrying the object.

Now, assume that there are $k$ palettes on the transfer modules. Simulations were performed for the following cases,

<table>
<thead>
<tr>
<th>Case</th>
<th>Number of Palettes</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>$k = 1$</td>
</tr>
<tr>
<td>A2</td>
<td>$k = 2$</td>
</tr>
<tr>
<td>A3</td>
<td>$k = 3$</td>
</tr>
<tr>
<td>A4</td>
<td>$k = 4$</td>
</tr>
</tbody>
</table>

Figure 9 shows the results for the respective cases. In fig.9, the horizontal axis indicates the trial number of palette, which transferred the object, and the vertical axis indicates the required time step for the completion of tasks. For all cases, the initial condition of the strategic vector field is random state. In case 1, because there is always only one palette on the modules, the optimal vector field is generated without difficulties. In case 2, although it takes longer time step for convergence of the strategic vector field, very efficient transfer is realized after convergence of the vector field likewise case 1. In case 3, since three palettes exit on the modules, sometimes congestion takes place, which may hinder smooth transfer in the narrow working space. Because the FTS continuously generates vector field, moderate congestion can be resolved. As shown in fig.9 (d), the system fails to deal with more than 4 palettes in this configuration. Since each palette has a different task, the occupation of the path by the another palette often disturbs successful organization of vector filed. We can notice that, until around 150 time steps, the FTS can modify the strategic vector very well, the vector field lose high flexibility after long time steps. This is a fundamental problem of the current stage of proposed algorithm. We are aware of how this takes place. It is due to saturation of the reward vector, not because of the learning ability of the algorithm. Now, we are working on this problem from various approaches. The average required time steps for respective cases are summarized in table 1.

<table>
<thead>
<tr>
<th>Case</th>
<th>Required Time Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>8.81Step</td>
</tr>
<tr>
<td>A2</td>
<td>11.99Step</td>
</tr>
<tr>
<td>A3</td>
<td>22.19Step</td>
</tr>
<tr>
<td>A4</td>
<td></td>
</tr>
</tbody>
</table>

Table 1 Average task completion time (1)

In the next, we examine the case where the palette is thrown at the constant rate to the FTS. Now, let us consider the case where a palette is thrown once per $k$ steps in the system, as follows,

<table>
<thead>
<tr>
<th>Case</th>
<th>Number of Steps Per Palette</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td>$k = 5$</td>
</tr>
<tr>
<td>B2</td>
<td>$k = 6$</td>
</tr>
<tr>
<td>B3</td>
<td>$k = 7$</td>
</tr>
<tr>
<td>B4</td>
<td>$k = 8$</td>
</tr>
</tbody>
</table>

Figure 10 shows the results for the respective cases. Likewise fig.9, the horizontal and vertical axis is the trial number of palette, and the task completion time for the
palette respectively. Also, the initial state of the strategic vector is randomized. In case B1, as shown in fig. 10(a), the transfer cannot catch up with the speed of input rate. For the palette to visit two machining sites, at least 8 steps are required (see fig. 8), and the self-organization requires many steps at the first stage. Therefore, the system will fall into a deadlock in the first stage before the vector field is generated. In case B2 and B3, although the initial stage of the task execution requires long time steps, the developed vector field can cope with an amount of task efficiently. In case B4, the performance is almost optimal. Also, we can see that the overall performance is very good once after the strategic vector is self-organized. However, we see that conditions at the initial stage should be moderate until the vector field is established, although the system can manage to solve the congestion of the path. The average required time steps in the simulation are shown in table 2. According to table 2, there is a little difference in the performance where the strategic vector is successfully generated.

<table>
<thead>
<tr>
<th>CaseB1</th>
<th>CaseB2</th>
<th>CaseB3</th>
<th>CaseB4</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.52 Step</td>
<td>11.81 Step</td>
<td>10.31 Step</td>
<td></td>
</tr>
</tbody>
</table>

Table 2 Average task completion time (2)

Fig. 10 Task completion time for each palette (2)

5. Summary

In this paper, we proposed the flexible transfer system (FTS) composed autonomous robotic transfer modules. Despite each module can deal with only local information, the global transfer path can be generated for the task configurations. This is achieved by introducing the multi-layered strategic vector field. According to the machining information, the vector field is selected and referred for the desired path. We apply the learning automata to the self-organization of the vector field. Due to the ability of self-organization, the system possesses high potential to cope with unknown breakdown of the transfer module and changes of the configuration, such as a location of the machining sites and extensions. Generated path is not necessarily optimal in terms of the conventional meaning of optimality, but the simulation results show the basic performance is satisfactory for the moderate conditions, which we call quasi-optimal. We confirmed that simulations of different task configurations exhibit the similar results. On the other hand, the experimental system is developed and ready for implementation. However, a fundamental problem remains to be solved, which is re-organization of the strategic vector field. We are currently working on the issue to enhance further flexibility of the FTS.

This research is a part of IMS Project.

References