Holonic Robot System: A Flexible Assembly System with High Reconfigurability

MASAO SUGI  YUSUKE MAEDA  YASUMICHI AIYAMA†  and  TAMIO ARAI

DEPARTMENT OF PRECISION ENGINEERING,
THE UNIVERSITY OF TOKYO
7-3-1 HONGO, BUNKYO-KU, TOKYO 113-8656, JAPAN

†INSTITUTE OF ENGINEERING MECHANICS AND SYSTEMS,
UNIVERSITY OF TSUKUBA
1-1-1 TENNODAI, TSUKUBA CITY, IBARAKI PREF. 305-8577, JAPAN
{sugi,maeda,arai}@prince.pe.u-tokyo.ac.jp
aiyama@esys.tsukuba.ac.jp

Abstract

This paper proposes a flexible assembly system that manages production by negotiating with autonomous manufacturing devices. With this mechanism, physical layout of the system is independent of the type of products to be assembled, and the system can asynchronously process multiple parallel production tasks. This system also supports Plug & Produce; a system function that enables easy reconfiguration. Thus the system can be quickly accommodated to breakdowns and changes in production quantities. This paper also deals with an index for general evaluation of the manufacturing systems. With this index, we can plan appropriate improvement of the system layouts. Then we reconfigure the system using Plug & Produce function, in order to accommodate it to the changes in the manufacturing environments.

1 Introduction

In recent years, the demands of the consumers have been diversifying, and the manufacturing environments have been getting no more static but dynamic. The manufacturers are expected to keep varieties of their product lineups, and quickly produce appropriate types of products in adequate quantities timely. For this reason, not only productivity, but flexibility against the changes of the manufacturing environments is also required for manufacturing systems. Applying distributed autonomous systems to manufacturing has been become very important.

Manufacturing system with this architecture has been widely studied. A common feature of those studies is the hierarchy of agents performing task assignments by competitive bidding [3, 7, 8]. Such hierarchical architectures, where agents achieve both autonomy and cooperation in balance, agree with the concept of Holon and Holarchy (i.e. holonic hierarchy) proposed by A. Koestler [4]. Most of the existing researches have, however, focused on the distributed scheduling or the negotiation between agents, and little attention has been paid to the implementation onto the actual manufacturing systems.

It is very important to estimate the expected performance at the designing stage, and to evaluate the current status in manufacturing systems [5, 6, 9]. Evaluation of conventional systems have been attempted so far, but that of distributed autonomous systems still remains to be developed.

This paper proposes Holonic Robot System that aims at assembling activities of real manufacturing devices. This system assembles products by negotiations between production management agents and device agents. Furthermore, this paper proposes "Plug & Produce", a system function which enables easy addition/removal of devices. Then Plug & Produce function is introduced to Holonic Robot System. Evaluation of the proposed system is also dealt with in this paper, since it is essential for effective utilization of system’s high reconfigurability. We propose the evaluation index suitable for distributed autonomous manufacturing systems. We also describe how to calculate the index.
2 Holonic Robot System (HRS)

In this section, Holonic Robot System (HRS) is outlined. Holonic Robot System, based on the concept of Holon by A. Koestler [4], is an assembly system with high flexibility [1, 2].

2.1 Holarchy

Holonic Robot System consists of two different kinds of layers as shown in Figure 1. Execution layer is located at the lower level and represents the manufacturing devices. It controls devices. In this layer, holons exist statically since they correspond to actual devices. Management layer is the upper one to manage assembly processes. Inside of the management layer, there are three kinds of layers; task layer, process layer and operation layer.

Each component of the layers is a holon. An upper holon asks lower holons to execute an order and lower holons reply by bids. These three layers form a kind of hierarchy, called holarchy. We will show the details of the management holons and the execution holons.

2.2 Management holons

The management holons aim at:

- Managing assembly processes of a product: The management holons look after assembly processes according to a product database that describes how to assemble a product with parts.
- Assigning devices to assembly process: According to the sequence of assembly, the holons ask assembly devices and make contracts with them.

Here an assembly “device” means equipment such as manipulators, belt-conveyors, warehouses and so forth. These two characteristics are realized by introducing three layers. Holons in each layer have different functions as follows;

(I) Task holon: It manages the whole assembly process of a product. There exists only one task holon for the manufacturing of a product. It works as the top holon of the assemble holarchy.

(II) Process holon: It manages one assembly process that forms a part of the whole assembly task. A process holon is dynamically created by the task holon.

(III) Operation holon: One assembly process may include a series of motion of devices. According to the request from a process holon, an operation holon makes devices move.

2.3 Execution holons

Execution holons exist statically and originally in order to control existing assembling devices such as manipulators, belt-conveyors and warehouses. Note that one device is usually subject to its own execution holon. When a device is mounted onto a system, the existing “Execution holon” is also installed onto the holarchy.

It bids for a movement request from an operation holon and negotiates about the movement with its upper holons. When it receives an order, it controls motions of the device and reports results. It observes the status of devices, and it also detects and reports errors.

2.4 Experimental implementation

An assembly cell is installed according to the proposed architecture. We made an assembly cell that has three manipulators, one bi-directional belt-conveyor and two warehouses as shown in Figure 2. The task of a cell is to assemble parts into a product. The parts are initially set in warehouses and assembled in the cell, then the product is transferred back to one of the warehouses.

Each execution holon has a configuration file containing information on:

(a) relations between its local coordinates and the world coordinates,
(b) relay points to its neighboring devices that used for transferring parts from device to device, and
(c) data of objects stored in a buffer of each device.

In this system, every product is described as an assembly tree, i.e. precedence graph. Assembly trees are decomposed by the management holons into simpler operations (“move” or “insert”). An operation holon negotiates with execution holons, i.e. manufacturing devices, and assigns the operation to proper
devices. Neither specified cell layouts nor control programs for each product type are necessary. If only a new assembly tree is prepared, the system can start to assemble new products. It means that the proposed system is flexible against changes in product lineup.

Using this assembly cell, we carried out experiments to assemble some kinds of products. Detailed description of the experiments is shown in our previous paper [1]. The results show that the system can cope with multiple parallel tasks asynchronously.

3 Plug & Produce

3.1 Outline of the concept

The physical reconfiguration of a usual manufacturing system is very expensive, because the inevitable information renewal by the physical change is very troublesome. Thus the reconfiguration is preferred to be avoided as much as possible. If the reconfiguration can be carried out easily, the system can deal with the change of production quantity and the occurrence of machine breakdowns rapidly, by adding or removing some machines in the system.

The authors consider the assurance of easy reconfiguration necessary for the manufacturing systems, and call this function as “Plug & Produce,” on the analogy of the Plug & Play on personal computers.

Plug & Play enables easy addition/removal of peripherals by automating information renewals, i.e. installation of the device drivers or modification of the system configuration file, which has been manually managed. In Plug & Produce, information renewals associated with physical changes are automated in order to reduce costs at the reconfiguration. This method is equivalent to that in Plug & Play.

Applying Plug & Produce function along with distributed autonomous architecture, manufacturing systems will achieve much higher flexibility.

3.2 Requirements for Plug & Produce

In this subsection, let us consider what is necessary for the distributed autonomous manufacturing system in introducing Plug & Produce function.

Plug & Produce and Plug & Play have the same target and they use the same approach to the aim. However, system requirements are rather different. This is caused by the difference of system architectures and existence of cooperation.

In Plug & Play, there is no cooperation among the peripherals, and resource conflict occurs in the IRQ (interrupt request) number or the I/O port addresses, and is solved by the central management of the PC.

On the other hand, manufacturing devices cooperate with each other, and they conflict on their working spaces. Furthermore, this conflict should be solved by means of distributed coordination, since we presume the architecture of the system not to be centralized but to be distributed.

In this distributed solution of the problem, heterogeneity of the system should be considered. The abilities of the manufacturing devices are not uniform in the view of degrees of freedom (e.g. manipulators have three or more degrees of freedom, but belt-conveyors have only one).

Manufacturing systems can realize the Plug & Produce function by satisfying these requirements.

3.3 Implementation of Plug & Produce

In the previous subsection, we considered the requirements for introducing Plug & Produce onto manufacturing systems with distributed architecture. Here Plug & Produce function is implemented onto HRS.

As mentioned in the subsection 3.2, mutual conflict of devices occurs on their working spaces. There are some available methods to solve this problem. For example, if each device knows the position of the neighbors at any moment, it can evade collision with others. Such a method is, however, too much rigorous for assembly system, where movement of the manufacturing devices are rigidly limited to several patterns of the motion.
The working space of each device is divided into two kinds of domains, and the rules of device motion are introduced as follows:

- **Shared Domain**: A domain where the neighboring devices can enter. This domain is used only for transferring parts or products. Relay points of two neighboring devices are defined in this domain.

- **Exclusive Domain**: A domain where other devices cannot enter. This domain is used for assembly and keeping parts or products.

Each device memorizes the allocation of exclusive domain and shared domain in its own working space. This allocation will be modified if the number of neighboring devices changes, as shown in Figure 3. If a new device comes, a new shared domain will be defined for a new comer and the exclusive domain will be decreased. On the other hand, if a neighbor is removed, the shared domain with the removed device will be incorporated into the exclusive domain.

Here, let us settle the procedure for adding new devices. As preconditions for each device, the reachable area and the relation between its local coordinates and the world ones are given. Then the procedure is settled as follows:

1. A human operator sets the new device in the cell.
2. The operator launches an execution holon that corresponds to the new device.
3. The new holon asks all of the existing holons about their location and the shape of their working space. Then the new holon understands its neighbors.
4. The new holon secures all of the neighbors and makes them halt.
5. The new device negotiates with each neighbor to decide shared spaces and relay points.
6. If all the neighboring holons have finished the renewal of their information, the new holon releases them. Then the procedure finishes.

In the above procedures 4–6, the information stored in multiple devices are renewed simultaneously and guaranteed no contradiction to each other.

In procedure 5, the relay point negotiations between the new device and each of its neighboring devices are held. As mentioned in the subsection 3.2, not the new device but a device with lower degree of freedom seizes the initiative in this negotiation. For example, in a negotiation between a manipulator and a belt-conveyor, the conveyor decides the relay points and the manipulator obeys it. If the manipulator determines the relay points, these points might be inconvenient for the belt-conveyor, because the manipulator has three (or more) degrees of freedom but the conveyor does only one.

In the case of removal of a device, similar procedure is used. The device to be removed secures all of the neighboring devices, asks them to renew information, then releases them and eliminates itself.

### 3.4 Simulation about addition or removal of devices

The authors have constructed a simulator of the HRS with the Plug & Produce function by the above-mentioned procedures. Using this simulator, addition/removal of a device is examined.

An assembly cell composed of two manipulators, one belt-conveyor and two warehouses is assumed, as shown in Figure 4. In the midst of the assembly of "Product16" that consists of four parts, a new manipulator "Robot3" is added into the cell.

The result is shown in Figure 5. At 28[sec] from the task started, a new execution holon “Robot3” is launched. Robot3 collects information of the existing devices and concludes that the neighbors are Robot1 and Robot2. Robot3 tries to secure both of them. Robot1 and Robot2 are secured at 31[sec], then information renewal starts. At 33[sec], the renewal finishes. Robot3 releases Robot1 and Robot2 and joins
the bids. The time spent for the whole procedure was 6 [sec]. From 72 [sec] to the end, Robot3 contracts for three operations and executes them. This means that the added manipulator works regularly.

We have also examined the case of the removal of the belt-conveyor in the middle of the assembly of “Product16”. The assembly continues to function and the reconfiguration is completed.

In this way, easy and quick addition/removal of devices is realized by automated information renewal of devices.

4 Evaluation of the System

In HRS, physical layout of the system is totally independent of product types. Thus we can arbitrarily compose manufacturing devices. If some of the devices may break down, the system can maintain manufacturing activities using remaining devices. We can also add new devices or remove the existing ones freely, even if manufacturing tasks are ongoing. Nevertheless, these functions are only the necessary condition for system to be flexible, and not the sufficient condition. This may be due the facts that we don’t have any evaluation criterion of determining optimal layouts, or we cannot find out where to add a new device. An index is necessary for evaluating total performance of the system.

In addition, predictive evaluation is necessary; if the system is finally turned out to be inadequate after reconfiguration, such an “observe-only” evaluation is useless. Before the system reconfiguration, we must predict the performance of the modified system for every alternative layout plan, reject inadequate plans and choose the optimal configuration.

In a word, it is necessary to introduce an index to evaluate the system and a method to predict the future performance using that index. With both of the index and the prediction method, HRS will really obtain high flexibility.

4.1 System evaluation index

In this subsection, we consider what item should be included in the general index, and determine the index.

Productivity is regarded as the most important ability of manufacturing systems. This can be simply measured as production quantity per unit time (or as its reciprocal, necessary time per one product). Flexibility has been often neglected so far, but it is also an important ability. In this paper, flexibility is considered equal to fault-tolerance. Fault-tolerance can be calculated using theory of reliability.

Koren [5] proposed an index “Expected Productivity”, which is expressed by productivity and reliability. Concretely, expected productivity $E[P]$ is defined as follows:

$$E[P] = \sum_{\lambda \in \Lambda} Pr(\lambda)V(\lambda), \quad (1)$$

where $\Lambda$ is a set of possible fault patterns. A system composed of $n$-devices has $2^n$ patterns. $Pr(\lambda)$ is probability for occurrence of a fault pattern $\lambda$. $V(\lambda)$ is the productivity under the pattern $\lambda$.

We think this index is appropriate, since it contains both flexibility and productivity. In this research, expected productivity is used for the evaluation.

4.2 Calculation method of the index

Expected productivity assumes the system to be conventional manufacturing line. In order to use this index for evaluation of HRS, we must solve some problems.

Productivity of manufacturing lines are easily calculated using their tact times. On the other hand, HRS is asynchronous and dynamic system, and can execute multiple tasks in parallel, which is mentioned in section 2. Even if all the conditions of the two assembly cells are the same, their total processing times might be different, since their behavior is not always the same. Thus, it is difficult in HRS to predict exact productivity of the cell. In this paper, the following approximation is used for prediction.

1. Prepare an assembly cell in the simulator, where the condition of the imaginary cell reflects that of the real system.
2. Put in more than one assembly tasks in the simulator. Using the total processing time resulted, calculate processing time per one task. As the number of the tasks increases, processing time per task will be reduced through the system’s parallel processing ability. However this reduction of time will finally reach a critical point, since the number of tasks executed in parallel might be limited.

3. Increasing the number of tasks gradually, find out the minimum value of time per task. Calculate the productivity (i.e. quantity per unit time) as the reciprocal of this minimum value.

4. Repeat the same simulation and get the average of the productivity.

Another problem is how to take into account the heterogeneity of the system. In ref. [5], manufacturing devices are treated as versatile machines that is not so concrete an entity. In this model, whether the system can continue the production or not is depends only on the preservation of the connection; if the system separated, the production is no more feasible. We deal with heterogonious system, where manipulators (for assembling parts or moving parts), belt-conveyors (only for moving parts), and warehouses (for feeding parts or keeping products). We must consider whether these abilities (assembling, feeding, etc.) is preserved in the system, in order to continue the production.

4.3 Sample case: real calculation of the index

In this subsection, we calculate the expected productivity on a sample problem.

Figure 6 shows an assembly cell with two manipulators, one belt-conveyor and two warehouses. The third manipulator “Robot3” is added in the cell. Here let us name the original cell with two manipulators “Cell-1” and the enhanced cell “Cell-2”. We will evaluate these two cells.

Let us consider the expected productivity of “Cell-1”. As shown in Figure 6, “Cell-1” has a linear connection. If one of the devices breaks down, “Cell-1” cannot continue assembly. In breakdowns of a manipulator or a belt-conveyor, the cell is separated into two. In the case of a warehouse, the parts necessary for the assembly cannot be supplied any more. Therefore, the expected productivity of the “Cell-1” \( E[P](\text{Cell-1}) \) is expressed as the multiplication of productivity \( V(\text{Cell-1}) \) and reliability of each devices,

\[
E[P](\text{Cell-1}) = V(\text{Cell-1}) \cdot R_{\text{Storage1}} \cdot R_{\text{Robot1}} \cdot R_{\text{BeltConv}} \cdot R_{\text{Robot2}} \cdot R_{\text{Storage2}}. \tag{2}
\]

In order to calculate \( E[P](\text{Cell-1}) \), we have to know the reliability of each manufacturing devices. In this paper, each reliability is given by,

\[
R_{\text{Robot1}} = R_{\text{Robot2}} = R_{\text{Robot3}} = 0.9,
R_{\text{Storage1}} = R_{\text{Storage2}} = 0.99,
R_{\text{BeltConv}} = 0.95. \tag{3}
\]

Substituting these values in the equation (1), we get

\[
E[P](\text{Cell-1}) = V(\text{Cell-1}) \cdot 0.99 \cdot 0.9 \cdot 0.95 \cdot 0.9 \cdot 0.99 = V(\text{Cell-1}) \times 0.754, \tag{4}
\]

where \( V(\text{Cell-1}) \) will be obtained through the simulation.

Next let us consider the expected productivity of “Cell-2”. “Cell-2” has network connection, while “Cell-1” has linear one. Consequently, “Cell-2” can continue production even if some devices are out of order. Here we list all the fault patterns in which the cell can continue the assembly. Calculating probability of each usable pattern, we obtain the result shown in Table 1.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline
Pattern & S1 & S2 & R1 & R2 & R3 & B & \( Pr(\lambda) \) \\
\hline
\( \lambda_1 \) & & & & & & & \( 5.99 \times 10^{-1} \) \\
\hline
\( \lambda_2 \) & & & & & & \times & \( 3.57 \times 10^{-2} \) \\
\hline
\( \lambda_3 \) & & \times & & & & & \( 7.54 \times 10^{-2} \) \\
\hline
\( \lambda_4 \) & & \times & & \times & & & \( 7.54 \times 10^{-2} \) \\
\hline
\( \lambda_5 \) & & \times & \times & & & & \( 3.97 \times 10^{-3} \) \\
\hline
\( \lambda_6 \) & & \times & & \times & & & \( 7.54 \times 10^{-2} \) \\
\hline
\( \lambda_7 \) & & \times & \times & \times & & & \( 3.97 \times 10^{-3} \) \\
\hline
\( \lambda_8 \) & & \times & \times & \times & \times & & \( 8.38 \times 10^{-3} \) \\
\hline
\( \lambda_9 \) & & \times & \times & \times & \times & \times & \( 4.41 \times 10^{-4} \) \\
\hline
\end{tabular}
\caption{Usable Fault Patterns and Their Probabilities}
\end{table}
Using the method described in subsection 4.2, we obtain the result shown in the Table 2. Using Table 2, we obtain the expected productivity of “Cell-2”,

\[
E[P](\text{Cell-2}) = \sum_{i=1}^{9} Pr(\lambda_i)V(\lambda_i) = 1.09. \tag{5}
\]

The expected productivity of “Cell-2” is 1.09 and that of “Cell-1” is 0.804. This means that “Cell-2” is superior to “Cell-1”. Thus we have derived the difference between two cells quantitatively, supporting what is expected. Moreover, using this index and simulations, we can predict the increase of system performance at reconfiguration. Now the system can deal with changes of environment, by adequate reconfiguration.

### 5 Conclusion

An assembly system with high flexibility has been studied as an approach to the total view of assembly systems in the future.

- Holonic robot system as an assembling system with a holonic architecture is proposed.
- A holonic robot system is implemented as an assembly cell with three manipulators, one belt-conveyor, and two warehouses. The cell can deal with multiple parallel tasks asynchronously.
- A system function Plug & Produce is proposed that supports easy and quick reconfiguration.
- Plug & Produce function is realized on holonic robot system. System shows quick response to physical reconfiguration.
- Applying expected productivity as general index that evaluates the system, the method to predict the system performance is presented. By this index and prediction method, we can add a new device to proper location quickly.

### Acknowledgements

This research is supported by IMS international research program HMS (Holonic Manufacturing Systems). The authors thank the members of HMS, especially those of HANDS (Holonic Handling Systems) for their valuable comments.

### References


<table>
<thead>
<tr>
<th>Pattern</th>
<th>( V(\lambda_i) ) (1/min.)</th>
<th>( Pr(\lambda_i) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \lambda_1 )</td>
<td>1.418</td>
<td>5.99 \times 10^{-1}</td>
</tr>
<tr>
<td>( \lambda_2 )</td>
<td>1.418</td>
<td>3.57 \times 10^{-2}</td>
</tr>
<tr>
<td>( \lambda_3 )</td>
<td>1.066</td>
<td>7.54 \times 10^{-3}</td>
</tr>
<tr>
<td>( \lambda_4 )</td>
<td>1.463</td>
<td>7.54 \times 10^{-3}</td>
</tr>
<tr>
<td>( \lambda_5 )</td>
<td>1.463</td>
<td>3.97 \times 10^{-4}</td>
</tr>
<tr>
<td>( \lambda_6 )</td>
<td>1.281</td>
<td>7.54 \times 10^{-4}</td>
</tr>
<tr>
<td>( \lambda_7 )</td>
<td>1.281</td>
<td>3.97 \times 10^{-4}</td>
</tr>
<tr>
<td>( \lambda_8 )</td>
<td>1.289</td>
<td>8.38 \times 10^{-3}</td>
</tr>
<tr>
<td>( \lambda_9 )</td>
<td>1.289</td>
<td>4.41 \times 10^{-4}</td>
</tr>
</tbody>
</table>