Basic study on production scheduling method for highly-distributed manufacturing systems

Eiji MORINAGA*, Akira TAKAGI*,**, Yuki SAKAGUCHI*,
Hidefumi WAKAMATSU* and Eiji ARAI*

* Osaka University, Division of Materials and Manufacturing Science

2-1 Yamadaoka, Suita, Osaka 565-0871, Japan

E-mail: morinaga@mapse.eng.osaka-u.ac.jp

** Currently, Sojitz Corporation

2-1-1 Uchisaiwaicho, Chiyoda, Tokyo 100-8691, Japan

Received 23 February 2014

Abstract

Recent development of computer network technologies is realizing highly-distributed manufacturing systems from the hardware point of view, where each facility is computerized and manages itself autonomously by communicating with other facilities. For this new type of manufacturing systems, a new discrete event simulation paradigm has been discussed, in which sorting of events is performed by exchanging messages about their occurrence times among the facilities. To make this paradigm beneficial enough, it is desirable that re-scheduling process performed after the simulation is carried out in a similar paradigm. For this reason, a highly-distributed scheduling method was proposed, where priority of each facility for job assignment is dynamically updated with indirect decision making and information control performed by a communication protocol. However, this method was developed only for a very simple scenario, and should be enhanced so that it can be utilized in real manufacturing. This paper presents an improved method which can handle the more complicated scenario where materials are transported to a production area one-by-one at non-constant time periods and some of them cannot be assigned to any machine at its arrival time.

Key words : Production scheduling, Dispatching rules, Distributed manufacturing systems, Agile manufacturing

1. Introduction

In order to convert data of artifacts generated in product design into real objects and provide them to customers as products with high cost-performance, it is necessary to optimally design, properly manage and efficiently operate manufacturing systems. Production scheduling is one of the most important issues for achieving the sophisticated design, management and operation of manufacturing systems. Numerous works have been conducted for a long time (Muth and Thompson, 1963, Graves, 1981) and various approaches have been discussed such as mathematical programming (Cheng and Gupta, 1989), heuristic dispatching methods (Panwalker and Iskander, 1977), artificial intelligence (Akyol and Bayhan, 2007, Metaxiotis, Askounis and Psarras, 2002, Noronha and Sarma, 1991), Petri nets (Tuncel and Bayhan, 2007), etc. For evaluating in detail a production schedule generated by using methods proposed in these researches, simulation needs to be performed. Simulation techniques for manufacturing has been also discussed and many methods has been proposed (Carrie, 1988, Jahangirian, *et al.*, 2010). Most of those researches have been conducted with the centralized concept in which the whole manufacturing system is modeled and then the optimal schedule is generated or simulation is performed based on it.

Diversification of customers' needs in the last quarter century has driven transition from the low-mix high-volume manufacturing to the high-mix low-volume manufacturing. Nowadays, it is desired to realize agile manufacturing, which can adapt to change of situation flexibly and quickly. The centralized concept should not meet this requirement, since it would be hard to re-model the whole system and find the optimal way of management and operation for each new

situation. From this point of view, distributed manufacturing systems have been discussed, where each production area is managed by a computer and the whole system is controlled by communications among the computers. In the research field of production scheduling, distributed methods using auction mechanisms (Kaihara, *et al.*, 2010), active databases (ADB) (Matsumoto, *et al.*, 1999), etc. have been actively discussed. Also in the simulation area, high-level architecture (Kuhl, *et al.*, 1999) was proposed and distributed methods based on it have been discussed (Inukai, *et al.*, 2007, Pedrielli, *et al.*, 2012).

Recent development of computer and network technologies is achieving highly-distributed manufacturing systems (HDMSs) from the hardware point of view, where each facility is computerized and manages itself autonomously by communicating with other facilities. This point has been discussed in the field of manufacturing simulation, and a new discrete event simulation paradigm was proposed (Fujii, *et al.*, 2012). In this paradigm, events, each of which is simulated in one of the facilities, are sorted by their occurrence time and then the first event is simulated by the facility which the event belongs to. This sorting is performed by exchanging messages about the occurrence times among the facilities, and the message exchanges are realized just by each facility broadcasting its own messages. All decision makings and information control for sorting are not directly carried out by any facilities but indirectly by the communication protocol.

To make this paradigm beneficial enough, it is desirable that re-scheduling process performed based on the simulation result is also carried out with indirect decision makings and information control in a similar paradigm. Although there are some distributed scheduling methods which can be applied to HDMSs (Hatono, *et al.*, 1995, Hino, *et al.*, 2000), they require that some facilities perform direct decision making or information control for scheduling. From this point of view, a highly-distributed scheduling method was proposed (Morinaga, Arai, and Wakamatsu, 2013). In this method, priority of each facility is dynamically updated with indirect decision making and information control achieved by message exchanges. This method was applied to an example and its fundamental feasibility was proved. However, this method was developed only for a very simple scenario, and should be enhanced so that the method can be utilized in real manufacturing. In that scenario, materials of the same type are transported to a production area one-by-one at a constant time period, and assigned by the shortest processing time (SPT) rule to one of the facilities in the area, each of which has each performance. But, it is unrealistic to assume that the materials are transported at a constant time period. In addition, the method is infeasible in the case where a material can not be assigned to any facility at its arrival time due to relatively shorter arrival time periods than facilities' processing times. As the first step of the enhancement, these two problems should be resolved. We addressed this issue and reported an intermediate result at a conference (Morinaga, Takagi, *et al.*, 2013). However, the improved method was incomplete, since it (and also the original method) does not work when delays occur in communication among facilities due to including inadequate mechanism for arranging exchanged messages. Furthermore, it also does not work when multiple machines send some messages at the same time. This paper presents an enhanced method considering these issues.

This paper is organized as follows: Section 2 provides a brief overview of the conventional highly-distributed scheduling method and then describes its problems. In section 3, the method is enhanced for resolving the problems. In section 4, the enhanced method is applied to a production scenario where materials are transported at non-constant time periods that are relatively shorter than facilities' processing times. Feasibility of the enhanced method is shown through a case study. Section 5 presents conclusions.

2. Highly-Distributed Scheduling Method

The highly-distributed scheduling method (Morinaga, Arai, and Wakamatsu, 2013) is briefly reviewed in this section. Presupposed production scenario is given, and the base concept, an overview and problems of the method are briefly described.

2.1. Considered Production Scenario and Overview of Highly-Distributed Scheduling Method

Flexible flow shop of just one stage shown in Fig. 1 is considered. Materials of the same type are transported to a production area one-by-one at a constant time period a . The area is composed of a buffer (or automated storage/retrieval system (AS/RS)) and multiple machines. They are computerized and connected to each other by a network. The machines are of the same type but have different performance. A_i stands for the time required for processing a material by the i -th machine, and ID_i is the identification number of the machine. Materials arrived at the buffer are assigned to one of the machines by the SPT rule.

In order to execute the assignment, it is required to determine priority sequence of the machines according to the

Fig. 1 Considered production scenario.

SPT rule. In the highly-distributed scheduling method, this determination, called *sorting* in this paper, is carried out by message exchanges among the machines as shown in Fig. 2. As described in detail in the next paragraph, this sorting can be performed by making each machine broadcast a message including its processing time and identification number. Since the machine of the highest priority is autonomously determined, material assignments are also carried out by making the buffer broadcast a job request and making each machine return messages concerning to the requests.

Fig. 2 Priority sequence determination and material assignment by message exchanges.

Figure 3 shows the detailed processing flow of the i -th machine for initial sorting which is executed in advance before material assignments. The i -th machine has variables B_i and S_i which stand for the priority value of the machine. B_i is static and does not change its value once the initial sorting is completed. On the other hand, S_i changes its value during the material assignment phase according to priority sequence that dynamically changes depending on availability of machines as described later. In the initial sorting phase, both of the variables always take the same value. The values of the variables are initially set to 1 which means that the machine has the highest priority. The machine broadcasts a message M which includes its processing time A_i , identification number ID_i and message type number 0 which means that the message is being sent for the initial sorting. The other machines also send the same type of messages and then the i -th machine receives them. When the machine receives the same type of message including A_j and ID_j which has been broadcasted by the j -th machine, the i -th machine modifies its priority value by comparing A_i to A_j . If A_i is smaller than A_j , which means the i -th machine can process a material faster than the j -th machine, the i -th machine leaves the values of S_i and B_i . On the other hand, if A_i is larger than A_j , which means the i -th machine processes a material slower than the j -th machine, the i -th machine increases S_i by 1, which means its priority decreases, according to the SPT rule, and then copies the value to B_i . If A_i is equal to A_j , the second rule which prioritizes the machine with smaller identification number is introduced. The i -th machine compares ID_i to ID_j , and then the same process as performed concerning to A_i is executed. By repeating this set of processes $F - 1$ times, the priority variables of the i -th machine S_i and B_i are set to the correct value eventually. All of the machines perform this set of processes respectively, and then the priority sequence is correctly determined.

After the initial sorting phase, material assignments start. This assignment phase can be performed by making the buffer assign the arrived material to the machine which has the highest priority value ($S_i=1$). This can be basically ex-

Fig. 3 Processing flow for initial sorting.

cuted by broadcasting a message about job request from the buffer, broadcasting a message about accepting the job from the machine of the highest priority, and decrementing the priority values (i.e. increasing the priority) of the other machines by themselves. However, when a machine which accepted a previous job request completes the job, the machines which are available for processing the next-arriving material should rearrange their priority sequence. Figures 4, 5 and 6 show the detailed processing flow of the buffer and a machine, respectively. First of all, the buffer broadcasts a message of type 1, which means that the message is a request for processing a material, and then it increments Q , which stands for the number of assigned materials, by 1 as shown in Fig. 4. When a machine receives this message, it behaves as described in Fig. 5. The machine checks whether $Flag$ is equal to 0, which means it is not processing any previously-accepted material and therefore is available for the current material. If this is true, the machine performs some additional processes for accepting or rejecting the request according to its priority value S_i , and then the buffer increments WC by a to the time at which the next material arrives after receiving replies from all machines. If $Flag$ is not equal to 0, the machine replies to the buffer that it cannot accept the request by sending a message of type 6. In addition, the machine checks whether the world clock WC , which is included in the request message from the buffer, satisfies the inequality $T_{iE} > WC \geq T_{iE} - a$. If this is true, the machine requests other machines to re-arrange the priority sequence, which is called *re-sorting*, by broadcasting a message of type 3, since this machine will be available at the time $WC + a$ at which the next material will arrive at the buffer. This re-sorting is performed by the same mechanism as that for the initial sorting, but in the material assignment phase, not processing time A_i but static priority variable B_i is utilized for efficient re-sorting by avoiding the case of $A_i = A_j$.

Figure 7 shows an event trace diagram for an example to help readers to understand basic behavior of the facilities. In this example, there are a buffer and three machines. Required processing times for the machines are $A_1 = 2$, $A_2 = 3$ and $A_3 = 4$, respectively, and therefore $B_1 = 1$, $B_2 = 2$ and $B_3 = 3$. In the beginning of material assignment, S_i has the same value as B_i , i.e. $S_1 = 1$, $S_2 = 2$ and $S_3 = 3$. The material arrival period a is 1. Because there are no materials that have been accepted and processed in a machine, T_{iS} , T_{iE} and $Flag$ are set to 0. First of all, at time $WC = 0$, the buffer broadcasts a message $M(WC, ID_0, 1)$ for job request, and then increases Q . Because machines 2 and 3 do not have the highest priority ($S_2 \neq 1$, $S_3 \neq 1$), they send a reject message $M(ID_{2,3}, 6)$ back to the buffer. On the other hand, machine 1 has the highest priority ($S_1 = 1$) and is not processing any previously-arrived materials ($Flag = 0$). Therefore, this machine broadcasts an acceptance message $M(B_1, ID_1, 2)$ and then changes $T_{1S} = 0 (= WC)$, $T_{1E} = 2 (= WC + A_1)$ and $Flag = 1$. Then machines 2 and 3 increase their priority ($S_2 = 1$, $S_3 = 2$). Assignment of the first material can be completed by this set of processes. Next, the buffer increases the WC to the arrival time of the second material ($WC = 1$) and broadcasts the second job request message $M(WC, ID_0, 1)$. Machine 3 sends a reject message $M(ID_3, 6)$ back to the buffer. Machine 2 broadcasts an acceptance message $M(B_2, ID_2, 2)$ and then change $T_{2S} = 1 (= WC)$, $T_{2E} = 4 (= WC + A_2)$ and $Flag = 1$. Machine 3 increases its priority ($S_3 = 1$). Machine 1 also sends a reject message $M(ID_1, 6)$ to the buffer, and then broadcasts a message $M(B_1, ID_1, 3)$ to prepare for the next job request, since it will be available at the arrival time of the third material. Only machine 3 reacts to this message and its priority is decreased ($S_3 = 2$). Machine 3 sends a message $M(P (= 0), ID_3, 5)$ back to machine 1, and then, machine 1 change $S_1 = 1 (= S_1 + P)$ and $Flag = 0$. Now, all

Fig. 4 Processing flow of the buffer for material assignment phase.

Fig. 5 Processing flow of a machine for messages of type 1 and 2 in material assignment phase.

Fig. 6 Processing flow of a machine for messages of type 3 and 5 in material assignment phase.

the machines are ready for assignment of the third material, and then the buffer broadcasts the third job request message $M(WC, ID_0, 1)$ after increasing the world clock ($WC = 2$).

There are some distributed scheduling methods which can be applied to HDMSs. For example, Hatono, *et al.* proposed a distributed real-time scheduling method for job shop (Hatono, *et al.*, 1995). In this method, a job needs to determine the machine that process the next operation of the job based on the evaluation values for the operation sent from machines. In other words, jobs need to perform direct decision makings concerning to the scheduling. For another example, Hino, *et al.* proposed a decentralized job shop scheduling method (Hino, *et al.*, 2000). This method is for modification of a master schedule, and therefore it is premised on the idea that a machine knows which machine will process the next operation of its current job. Machines need to know information about other machines in the sense that they need to know who is in the area and who is in charge of each operation. Unlike these conventional methods, in this highly-distributed scheduling method, the buffer and machines do not need to know information about other machines. In addition, they do not need to make decisions concerning to scheduling, since the priority sequence is determined by the communication protocol.

2.2. Problems from the practical point of view

The conventional highly-distributed scheduling method overviewed in the previous subsection was developed for the very simple production scenario shown in Fig. 1, and should be improved thoroughly so that it can be applied to real manufacturing. As the first step, it is necessary to resolve the following two fundamental problems. In this method, it is assumed that a material can be always assigned to a machine at its arrival time. Therefore, this method can be applied only to the case where the arriving time period a is relatively longer than processing times A_i of the machines. This is inefficient and unreasonable, and this method should be enhanced so that it can be utilized even if the time period is shorter and some materials must be placed in a waiting queue at the buffer. In addition, it is also unreasonable to assume that all materials arrive at the buffer at a constant time period. This method should be also enhanced so that it can be applied to the case where materials are transported to the buffer at non-constant time periods.

Furthermore, the processing flows do not correctly work in the following two cases. The first one is the case where multiple machines broadcast messages of type 3 described in Fig. 5 at the same time. In this case, the increment of S_i , which should be performed only once for each combination of available machines, is carried out multiple times and therefore incorrect priority sequence is generated. For example, assume that two machines i and j respectively send messages of type 3 and machine i has higher performance than machine j . In this case, increment of the priority value S_j of machine j is performed twice, which should be once, due to receiving message of type 3 and message of type 5 both from machine i .

Fig. 7 Event trace diagram for the example of three machines. (Blue arrows stand for broadcasted messages.)

The second case is that where a message of type 1 reaches a machine which does not have the highest priority value after receiving the message of type 2 sent from the machine of the highest priority value due to delays in communication. In this case, the machine decrements its priority value before receiving the message of type 1, and wrongly accepts the material. As a result, both this machine and the machine with the highest priority accept one material.

Although an enhanced method which resolves the first and second problems was developed and reported in a conference proceeding (Morinaga, Takagi, Wakamatsu, and Arai, 2013) as an intermediate result, it was not developed considering the third and fourth problems. The next section describes a method enhanced further so that all of the four problems are resolved.

3. Enhanced Method for Highly-Distributed Production Scheduling

This section describes the enhanced highly-distributed scheduling method which can be applied to the production scenario where materials are transported to the buffer at non-constant time periods which are relatively shorter than processing times of materials on the machines and can work even if it faces multiple message exchanges performed at the same time and delays in communication.

Figure 8 shows the improved processing flow of the buffer. In this flow, the variables R , Q and C are introduced for coping with the case where an arrived material cannot be assigned to any machine at its arrival time and then is placed in a waiting queue at the buffer. In addition, the new variables $N1$ and $N15$ and the modified message type 3 and the new message type 15 are also introduced for avoiding the two problems caused by delays in communication or multiple message broadcasts at the same time.

In the conventional processing flow shown in Fig. 5, the machine broadcasts a message of type 3, which was defined for re-sorting in the conventional method, when the value of WC satisfies $T_{iE} > WC \geq T_{iE} - a$. This is because the arrival time period is constant and therefore a machine which cannot accept the current material due to processing of a previously-arrived material can know in advance whether it will be available when the next material arrives at the buffer. By carrying out the re-sorting process in advance before a material arrives, it is expected to avoid concentration of message exchanges

Fig. 8 New processing flow of the buffer for material assignment phase. (Each part has the following function: Part (I) is for checking that all machines are ready for a new job request and then broadcasting the request message; Part (II) is for memorizing that a machine has accepted the current job request; Part (III) is for checking all available machines have completed updating their priority value according to the job acceptance; Part (IV) is for counting the number of accepted materials and modifying the number of waiting materials if necessary; Part (V) is for properly storing the unaccepted material in the waiting queue; Part (VI) is for proceeding with the next assignment.)

in a very few moments. However, if the time period is not constant, this mechanism is no longer valid. Therefore, the re-sorting process should be executed at the same time when a material arrives at the buffer. To achieve this, the new message type 9 is introduced, as shown in the lower right part of Fig. 8. This type of message requests machines to prepare for job request, that is, to carry out the re-sorting process if required.

Figures 9, 10 and 11 show the improved processing flow of a machine. As described in Fig. 10, a machine performs the preparation process as follows when it receives a message of type 9 from the buffer (Part (III) of Fig. 10). At first, the machine checks whether it has been processing a previously-arrived material. If this is true (i.e., $Flag = 1$), the machine also checks whether processing of the latest previously-accepted material will be completed by the time when the next material arrives at the buffer, i.e., WC included in the received message is larger or equal to T_{iE} . If this is also true, the machine becomes available by the time when the next-coming material arrives at the buffer. Therefore, the machine changes $Flag$ to 0, $N9$ to 1 which means that the machine needs re-sorting, and then broadcasts a message of type 10 which requests the re-sorting process. On the other hand, if one of these two conditions is false, the machine does not need re-sorting because it is still unavailable or still available for the next-coming material, and then it broadcasts a message of type 14 which means that it is not necessary for this machine to perform re-sorting.

Fig. 9 New processing flow of a machine concerning to messages of types 1, 2 and 6 for material assignment phase. (Part (I) is for accepting or rejecting a job request; Part (II) is for updating the priority value of an available machine according to the job acceptance of another machine.)

If each of the other machines receives the message of type 10, the machine performs the following processes described in part (IV) of Fig. 10. It first checks whether processing of the latest previously-accepted material will be completed by the time when the next material arrives at the buffer ($WC \geq T_{iE}$). If this is false, this machine does not need to join the re-sorting, and sends a message of type 11 back to the re-sorting requester (i.e. the sender of the message of type 10). If true, the re-sorting process is executed in this machine. If the static priority of the machine is lower than the re-sorting requester (i.e. $B_i > B_j$), then the machine increases S_i by 1 and the number of this increasing process is recorded by increasing $N10$ by 1, which is required to obtain correct priority value when multiple machines broadcasts messages of type 10 at the same time as described later. If higher ($B_i < B_j$), S_i and $N10$ are left unchanged. Then a message of type 12 is sent back to the sender of the message of type 10.

The machine requesting re-sorting also updates its own priority value S_i if the static priority of the machine is lower than the sender of the message of type 12 (i.e. $B_i > B_j$). As shown in part (V) of Fig. 11, this machine counts the number of messages of types 11 and 12 by $N11$ and $N12$, respectively. When it receives those messages from all of the other machines ($N11 + N12 = F - 1$), the re-sorting process has been completed, and then the machine broadcasts a message of type 13 after resetting $N11$ and $N12$ to 0.

Fig. 10 New processing flow of a machine concerning to messages of types 8, 9 and 10 for material assignment phase. (Part (III) is for requesting re-sorting; Part (IV) is for responding to a re-sorting request broadcasted from another machine.)

Fig. 11 New processing flow of a machine concerning to messages of types 11, 12, 13 and 14 for material assignment phase. (Part (V) is for notifying all other machines that re-sorting requested by itself has been completed; Part (VI) is for modifying its wrong priority value caused by re-sorting requested by multiple machines.)

If only one machine requests re-sorting, this set of processes correctly works. However, if multiple machines request re-sorting, the priority value S_i of some re-sorting requesters are wrongly updated. Take the example considered in Fig. 7, and assume that machine 3 are available and machines 1 and 2 are currently unavailable but will be available for the next material. Therefore, the value of S_i is 1 for all machines and $Flag$ is 1 for machines 1 and 2 and 0 for machine 3, as shown in Fig. 12. In this case, machines 1 and 2 broadcast re-sorting request message (type 10), respectively. When machine 1 broadcasts the message of type 10, the priority values of machines 2 and 3 are increased by 1 ($S_2 = 2$ and $S_3 = 2$) and the value of machine 1 is left unchanged ($S_1 = 1$). Also machine 2 broadcasts the message of type 10, and the priority values are wrongly changed to $S_1 = 1$, $S_2 = 3$ and $S_3 = 3$ (S_2 should be 2). As can be seen in this example, the reason of this wrong re-sorting is that the priority value of a machine which needs re-sorting is updated not only by re-sorting requested by itself but also by re-sorting requested by another machine. Therefore, after performing re-sorting processes with all other machines, the machine which requested re-sorting must modify its priority value by subtracting the number of the unnecessary updatings caused by other requesters from the value. For this reason, the following processes are introduced. Each machine counts the number of messages of types 13 and 14 by $N13$ and $N14$, respectively. When it receives messages of these types from all of the other machines ($N13 + N14 = F - 1$), it modifies its current priority value S_i by subtracting $N10$ from itself when $N9 = 1$, since each increment S_i is executed twice in the machine which $N9$ is equal to 1. Finally, each machine sends a message of type 15 which means it is now ready for assignment of the next-arriving material (Part (VI) of Fig. 11).

Fig. 12 Example of wrong priority updating caused by multiple re-sorting requests. (Blue arrows stand for broadcasted messages.)

The buffer counts the number of messages of type 15 from all machines as shown in Fig. 8. When it receives this type of messages from all machines, now the preparation has been completed in all of them, and then the buffer broadcasts a message of type 1 for job request concerning to the next material (Part (I) of Fig. 8). As described in Fig. 9, when a machine receives this type of message, it checks whether it is not processing a previously-arrived material and it has the highest priority value. If these are true, the machine broadcasts a message of type 2 for accepting the request. If false, it refuses the request by broadcasting a message of type 6 (Part (I) of Fig. 9). When the buffer receives a message of type 2 from the machine with the highest priority, it increments R , whose initial value is 0, by 1 to memorize that the material is accepted (Part (II) of Fig. 8). When receiving messages of type 2 or 6 from all other machines, each machine which was available for the current material but could not accept it due to low priority increases its priority by subtracting 1 from its priority value. Then all machines send the buffer a message of type 3 which means that they have completed updating their priority values (Part (II) of Fig. 9).

When the buffer receives a message of type 3, the buffer performs a set of processes for proceeding to the next

material assignment. After making sure that all machines completed the update process by counting the number of received messages $N1$ and checking that $N1$ is equal to the total number of machines F (Part (III) of Fig. 8), it carries out one of the following two sets of processes for proceeding to the next assignment. If R is larger than 0, the material is accepted by a machine. Therefore, the value of Q introduced for counting the number of accepted materials is increased by 1, and then it is checked whether C , introduced for counting the number of materials in the waiting queue, is equal to 0. If this is false which implies the material was in the waiting queue, the value of C is decremented by 1, since the material in the queue has been successfully assigned to a machine (Part (IV) of Fig. 8).

If R is equal to 0, the material can not assigned to any machine because no machine are available. In this case, it is checked whether C is equal to 0, which means that there are no materials in the waiting queue. If this is true, C is increased by 1 because the current material must be newly placed in the waiting queue. If false, the current job request was broadcasted not for a newly-arrived material but for a material which had been placed in the waiting queue. Therefore, the value of C is left unchanged. Next, the time of the buffer WC is increased by 1, and then it is checked whether the updated value of WC is equal to AT_{Q+1} which is the arrival time of the next-arriving (the $(Q + 1)$ -th) material. If this is true, the $(Q + 1)$ -th material cannot be assigned to any machine and must be also placed in the waiting queue. Therefore, the value of C is increased by 1 again (Part (V) of Fig. 8).

After performing one of the above two sets of processes, R is reset to 0 and then the buffer broadcasts a message of type 8 or 9. If Q is equal to L , all materials have been successfully assigned to the machines and therefore the buffer broadcasts a message of type 8, which means all assignments has been completed. If Q is smaller than L , there are some materials to be assigned. Then the buffer broadcasts a message of type 9 (Part (VI) of Fig. 8).

These flows require that the buffer and all machines have obtained the value of F in advance. Strictly speaking, this is not desirable since this means that each facility must keep information about another facility, which may not meet the concept of the proposed method. However, this is not so serious since the information can easily obtained in the initial sorting phase shown in Fig. 3 just by adding a sub-flow for counting received messages of type 0.

These sets of processing flows can work even if all machines are processing other materials when a material arrives at the buffer and therefore it has to be placed in a waiting queue, since a mechanism for considering the waiting queue is implemented (parts (IV), (V) and (VI) of Fig. 8). In addition, the flows also work even if materials arrive at the buffer at non-constant time periods, because a mechanism for performing re-sorting at the arrival time of the next material is also implemented (part (VI) of Fig. 8 and parts (III), (IV), (V) and (VI) of Figs. 10 and 11). Furthermore, the flows work correctly even if they face multiple message exchanges performed at the same time and delays in communication, because of the mechanisms for modifying wrong priority values (part (VI) of Fig. 11) and for checking whether all other machines have responded (parts (I) and (III) of Fig. 8 and parts (II), (V) and (VI) of Figs. 9 and 11).

4. Case Study

The proposed method was applied to the production scenario shown in Fig. 1. In this scenario, 30 materials are transported to the buffer at non-constant time periods. The production area is composed of 4 machines, which are of the same type but have different performance. The times required for processing a material are 5 minutes for machine 1, 6 minutes for machine 2, 7 minutes for machine 3 and 8 minutes for machine 4, respectively. Materials arrive at the buffer at the times 0, 2, 4, 6, 8, 11, 11, 16, 19, 21, 24, 25, 26, 31, 31, 31, 32, 34, 35, 41, 42, 44, 46, 48, 49, 50, 56, 56, 56 and 56. Figure 13 shows the schedule generated by the SPT rule for this scenario. The materials arrived at the buffer at time 34 and 35 can not be assigned to any machine at the arrival times and therefore must be placed in a waiting queue for a while. At the times 31 and 56, three machines become available and multiple messages of the same type for re-sorting are exchanged each other at the same time.

The proposed processing flows were coded in the C language. Due to equipment constraints, the method was implemented on one computer and performed by using interprocess communication. The code was executed for this scenario, and the schedule in Fig. 13 was correctly generated by only message exchanges in accordance with the proposed flows.

5. Conclusion

Recent development of computer and network technologies is achieving highly-distributed manufacturing systems (HDMSs) from the hardware point of view, where each facility is computerized and manages itself autonomously by communicating with other facilities. From this point of view, a new discrete event simulation paradigm has been discussed.

Fig. 13 Gantt chart of the schedule generated by the SPT rule for the production scenario of the example.

Considering re-scheduling after the simulation, a highly-distributed scheduling method was also proposed for HDMSs, in which material assignment to machines based on the SPT rule is performed by message exchanges among the facilities. This method was developed for a very simple manufacturing scenario and has a lot of problems to be solved for using it in real manufacturing. In this paper, the conventional method has been enhanced such that the method can be applied to the case where materials arrive at the buffer at non-constant time periods which are relatively shorter than processing times of the machines. In addition, the method has been also improved so that it works even if it faces multiple message exchanges performed at the same time and delays in communication. Feasibility of the proposed method was shown by an example.

Future works include the following issues. One is to improve the proposed method further so that it can be applied to flexible flow shop for multiple kinds of products requiring multiple process steps and job-shop. Another issue is to take transportation problems into account. Development of algorithms for other dispatching rules such as the earliest-due-date rule and the minimum slack time rule is also important to make this scheduling concept practical.

Acknowledgements

We would like to thank the members of Technical Committee on Manufacturing and Management Knowledge, the Japan Society for Precision Engineering, for their advice.

References

- Akyol, D.E. and Bayhan, G.M., A Review on Evolution of Production Scheduling with Neural Networks, Computers and Industrial Engineering, Vol.53 (2007), pp.95-122.
- Carrie, A., Simulation of Manufacturing Systems (1988), John Wiley & Sons.
- Cheng, T.C.E. and Gupta, M.C., Survey of Scheduling Research Involving Due Date Determination Decisions, European Journal of Operational Research, Vol.38 (1989), pp.156-166.
- Fujii, S., Fujii, N., Iwamura, K., Morinaga, E., Tsumaya, A., Inoue, T. and Mariyama, T., A Basic Study on A Highly Distributed Simulation of Manufacturing Systems Under The Ubiquitous Environment, Proceedings of the ASME/ISCIE 2012 International Symposium on Flexible Automation (2012), Paper No. ISFA2012-7208.
- Graves, S.C., A Review of Production Scheduling, Operations Research, Vol.29 (1981), pp.646-675.
- Hatono, I., Tachibana, K., Umamo, M. and Tamura, H., Distributed Real-time Scheduling for Flexible Manufacturing, Transactions of the Society of Instrument and Control Engineers, Vol.31 (1995), pp.108-115 (in Japanese).
- Hino, R. and Moriwaki, T., Decentralized Job Shop Scheduling by Recursive Propagation Method, Transactions of the Japan Society of Mechanical Engineers, Series C, Vol.66 (2000), pp.2449-2455 (in Japanese).

- Inukai, T., Hibino, H., and Fukuda, Y., Enhanced distributed-simulation using orin and hla, *Mechatronics for Safety, Security and Dependability in a New Era* (2007), Elsevier, pp.261-264.
- Jahangirian, M., Eldabi, T., Naseer, A., Stergioulas, L.K., and Young, T., Simulation in manufacturing and business: A review, *European Journal of Operational Research*, Vol.203 (2010), pp.1-13.
- Kaihara, T., Fujii, N., Toide, S., Ishibashi, H. and Nakano, T., Optimization Method using Combinatorial Auction for Production Scheduling with Batch Processing, *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, Vol.4 (2010), pp.588-596.
- Kuhl, F., Weatherly, R., and Dahmann, J., *Creating Computer Simulation Systems: An Introduction to the High Level Architecture* (1999), Prentice Hall.
- Matsumoto, T., Kato, Y., Nagafune, N., Wakamatsu, H., Shirase, K. and Arai, E., Advanced Autonomous Distributed Manufacturing System Using Active Database, *Transactions of the Japan Society of Mechanical Engineers, Series C*, Vol.65 (1999), pp.837-843 (in Japanese).
- Metaxiotis, K.S., Askounis, D. and Psarras, J., Expert Systems in Production Planning and Scheduling: A State-of-the-Art Survey, *Journal of Intelligent Manufacturing*, Vol.13 (2002), pp.253-260.
- Morinaga, E., Arai, E. and Wakamatsu, H., A Basic Study on Highly Distributed Production Scheduling, *Advances in Production Management Systems. Competitive Manufacturing for Innovative Products and Services* (2013), pp.638-645, Springer Berlin Heidelberg.
- Morinaga, E., Takagi, A., Wakamatsu, H. and Arai, E., A Basic Study on Production Scheduling for Highly-Distributed Manufacturing Systems, *Proceedings of the 22nd International Conference on Production Research* (2013), Paper No. 417.
- Muth, J. and Thompson, G.L. (eds.), *Industrial Scheduling* (1963) Prentice-Hall.
- Noronha, S.J. and Sarma, V.V.S., Knowledge-Based Approaches for Scheduling Problems: A Survey, *IEEE Transactions on Knowledge and Data Engineering*, Vol.3 (1991), pp.160-171.
- Panwalker, S.S. and Iskander, W., A Survey of Scheduling Rules, *Operations Research*, Vol.25 (1977), pp.45-61.
- Pedrielli, G., Sacco, M., Terkaj, W., and Tolio, T., An HLA-based distributed simulation for networked manufacturing systems analysis, *Journal of Simulation*, Vol.6 (2012), pp.237-252.
- Tuncel, G. and Bayhan, M., Applications of Petri nets in production scheduling: a review, *International Journal of Manufacturing Technology*, Vol.34 (2007), pp.762-773.