
Agent-based agile manufacturing system scheduling

David He – Astghik Babayan

*Department of Mechanical and Industrial Engineering
The University of Illinois at Chicago
Chicago, IL 60607*

*davidhe@uic.edu
ababay1@uic.edu*

ABSTRACT. In this paper an attempt is taken to formalize the design and implementation of agent-based approach for agile manufacturing system scheduling. The general framework of an agent-based approach for agile manufacturing systems scheduling is presented. Some steps have been taken to construct a methodology for the development of negotiation mechanism in agent-based manufacturing systems scheduling to improve scheduling flexibility and robustness. A lower and upper bound for measuring the effectiveness of the scheduling system are also developed. The developed model integrates data and decisions associated with several entities within a scheduling system. This approach can be used to model and solve large-scale scheduling problems in an agile manufacturing environment.

KEY WORDS: manufacturing scheduling, agent-based heuristic, agile manufacturing systems.

1. Introduction

Producing customized products in a short time at low cost is one of the goals of agile manufacturing. To achieve this goal in a manufacturing system, products are differentiated either by a machining-driven or an assembly-driven differentiation strategy [HE 96]. The successful implementation of these two strategies lies in efficient scheduling of the system. To react fast to changes in the market, agile manufacturing demands its manufacturing operations be distributed, its manufacturing system be modular, interactive, and robust, and the scheduling problems be solved fast. Yet, most existing scheduling systems are developed based on centralized structures, which makes manufacturing systems scheduling complicated. The purpose of applying agent-based approach to solve scheduling problems is to make the scheduling system easier to design and implement, more robust and less prone to errors, easier to use, faster, cheaper, and so on.

In this paper, the agent-based approaches are applied to solve a complex scheduling problem in an agile manufacturing system. The structure of the manufacturing system that implements the product differentiation strategy in agile manufacturing is shown in Figure 1. It consists of two stages: machining and assembly, where there are m number of identical machines at the machining stage and q number of identical assembly machines at the assembly stage. The manufacturing system with this configuration is referred as $\{m, q\}$ manufacturing system.

To solve the scheduling problem, the representation of assembly sequence of a product produced in the system follows the digraph representation in [KUS 89]. In a digraph G , each node represents a part or a subassembly/assembly, and an arc represents a precedence relationship between two nodes. The level of assembly in a digraph is assigned as follows: value of 1 is assigned to the root node (assembly) and working backward from the root node, values of increment 1 are assigned to each subassembly node. Part nodes have the same assembly level as the corresponding subassembly or assembly nodes. Two products and the corresponding digraph representations of their assembly sequences are illustrated in Figure 2. The digraphs representing the assembly sequences of products can be classified into two types: simple digraph and complex digraph. A simple digraph is a digraph in which at most one subassembly node can be found at every assembly level (see Figure 2(a)). A simple digraph represents a linear assembly sequence of a product design. In a complex digraph, more than one subassembly node can be found on at least one assembly level (see Figure 2(b)). Throughout the paper, an assembly sequence of a product represented by a simple digraph is referred to as a simple assembly sequence and an assembly sequence of a

product represented by a complex digraph is referred to as a complex assembly sequence.

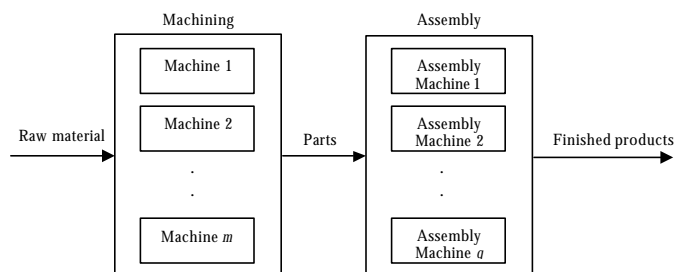


Figure 1. General structure of $\{m, q\}$ manufacturing system

The scheduling problem to be solved in this paper is defined in the following: there are m number of identical machines at the machining stage and q number of identical assembly equipment at the assembly stage (see Figure 1). The objective of the scheduling problem is to assign parts and subassemblies/assemblies to the machines at the machining and assembly stages and determine the processing sequences on the machines so that the makespan (C_{\max}), i.e., the maximum completion time, is minimized.

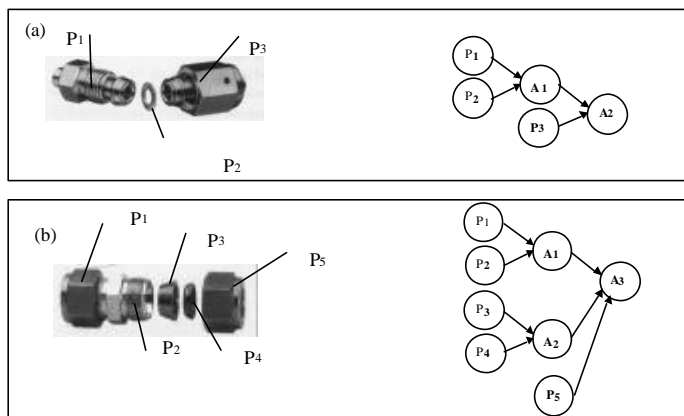


Figure 2. Example of (a) a product and its simple digraph and (b) a product and its complex digraph

2. Literature on agent-based scheduling

The scheduling problem to be solved in this paper is considered as a combination of two classical scheduling problems with complex assembly sequence: two-machine flow shop scheduling problem and parallel machine scheduling problem. The two-machine flow shop scheduling problem can be solved by Johnson's algorithm [JOH 54]. Parallel machine scheduling problem ($P \parallel C_{\max}$) has been proved by [GAR 78] as *NP*-hard in the strong sense when the number of machines is unlimited. However, the problem is solvable in pseudo-polynomial time when the number of machines is fixed and thus *NP*-hard only in the ordinary sense. Most of the algorithms developed for solving $P \parallel C_{\max}$ are heuristics (e.g., [GRA 69], [COF 78], [FRI 86]). There are some problem characteristics that complicate solving scheduling problem: (i) precedence constraints between operations (machining/assembly); (ii) assignment of operations to machines; and (iii) sequence of unrelated (independent) operations (machining/assembly).

Scheduling problems have attracted various efforts in multi agent approaches (see for example [SOU 97], [RAB 94], etc.). The notion of agent was found in the wide range of research in Computer Science (CS), Distributed Artificial Intelligence (DAI), etc. The multi agent systems paradigm represents one of the most promising approaches to the development of agile scheduling systems in manufacturing [RAB 99]. In fact, distributed systems have the following advantages [DEC 87]: (i) can simplify problem solving by splitting the problem into simple tasks; (ii) can tolerate uncertain data and knowledge; (iii) offer conceptual clarity and simplicity of design; (iv) present graceful degradation in computational complexity; (v) allow incremental modification of the system boundary; and (vi) suit well to distributed problems.

The application of agent-based approach to manufacturing scheduling is based on the idea that scheduling agility can be improved because of distributed autonomous systems and negotiation based decision-making in a multi-agent environment. [MIN 86] suggests the use of multi-agent system where there is a wide-range of reasonably self-contained pieces of functionality that are independently distributed in terms of timing and resources.

A survey by [SEN 99] reports 30 projects using agent technology for manufacturing planning, scheduling and execution control where agents represent physical entities, processes, operations, parts, etc. Real world examples of multi-agent application include: power systems management (see for [JEN 95]; [VAR 94]), particle accelerator control [JEN 93], telecommunications network management [WEI 94], spacecraft control [SCH 93], computer integrated manufacturing [PAR 95], job shop scheduling [MOR 93], etc. Assembly line design problem with agent application was studied by [SPR 95]. The design problem was solved by distributing overall system among agents

responsible for tasks, workstations, resource cells, etc. The coordination among distributed subsystems was achieved by negotiation and self organization of agents. The objective of the design was minimization of technological cost of the equipment that was achieved by applying simulated annealing.

A common definition for agent or “processor” was not found in the literature. In spite of this, [SIC 92] consider a “processor” as an agent when it possesses at least the following three properties: (i) has a certain degree of autonomy to reason about and to make decisions by itself; (ii) has the capability to interact with other agents; and (iii) has the knowledge to independently solve a part of the global problem.

An agent can possess various properties depending on the nature of the problem under consideration and the environment. The agents involved in the decision making process are not necessarily with same functionality. The level of functionality is defined in the problem decomposition process. Some of the agent properties found in the literature (*i.e.* [GEN 94], [GAL 88], [ROS 85]) are appropriate for the type of the agents used in our system. Here are the properties we believe our agents possess: *autonomy, sociability, reactivity, pro-activity, veracity, benevolence, and rationality.*

3. Agent-based scheduling framework

In this section, the general framework of the agent based approach for scheduling in an agile manufacturing environment is presented. The successful implementation of agent based approach strongly depends on how the system is decomposed into interrelated subsystems so that the integrity of the system is not violated. The digraph representation of product assembly sequence allows successful decomposition of large size scheduling problems into sub-problems, so that the integrity of original problem is not violated.

The digraph decomposition procedure described by [KUS 89] is used for decomposing the overall scheduling problem into sub-problems. The decomposition procedure consists in the following; a complex digraph is decomposed into simple sub-digraphs by removing root node A_i (A_3 in example presented by Figure 2(b)) from it. Then, moving forward we remove all the assembly nodes until the digraph is decomposed to the level where all resulting sub-digraphs have simple structure. Products with simple digraph representation have linear assembly structure and their optimal scheduling can be achieved by polynomial algorithms (see [HE 02]). As a result of decomposition, a set of sub-digraphs and single nodes (machining and assembly operations) is generated. Each decomposed sub-digraph, generated during the decomposition process, is an individual job that needs to be scheduled according to its precedence relationships with other jobs. For each sub-digraph an agent is assigned as

an entity of the system that is responsible and authorized only for scheduling his job in the system. If a single assembly node generated during the decomposition process has immediate preceding part nodes all the part nodes immediately preceding to that assembly node are combined with that assembly node and assigned to an agent. As a result each agent will have at least one assembly operation to schedule. If designers make some changes in the product assembly structure, the sub-digraph corresponding to the redesigned component will be changed, hence the agent responsible for that component will take care of rescheduling. In this way the decomposition method, and the schedule obtained using agent-based approach is robust.

The autonomous nature of system agents also offers flexibility of decision making under different circumstances. However, like many other agent-based systems our system requires a significant monitoring overhead which is described next.

Besides the agents owning individual jobs, another type of agent, or so called *monitoring overhead* or *scheduling manager (SM)* is added to the decision making process in the system. Unlike the other agents in the system, scheduling manager is not responsible for scheduling activities. He/she is the decision-maker at a higher level and is responsible for choosing the winner among other agents during bidding process.

Before presenting agent-based scheduling algorithm, the following notations and definitions are introduced:

Define:

$J=(J_1, J_2, J_3, \dots, J_k)$ the set of the jobs to be scheduled;

SAJ =set of schedulable jobs, i.e., set of the jobs that can be scheduled because all their preceding jobs are scheduled, or they have no preceding job;

NSJ =set of non-schedulable jobs, i.e., set of the jobs that cannot be scheduled because some of their preceding jobs have not been scheduled yet;

SJ =set of scheduled jobs.

Let *stage* be a step of the scheduling process, when a change in the set of schedulable/scheduled jobs occurs.

If we define t as the index of a stage, accordingly, the sets SAJ , NSJ , SJ will be defined for stage t as: SAJ_t , NSJ_t , and SJ_t .

At any stage t the following equation holds: $J=SAJ_t \cup NSJ_t \cup SJ_t$. Next, the agent-based scheduling algorithm is presented.

3.1. Agent-based scheduling algorithm

- Step 1. Decompose the digraph into a set of simple digraphs and assign an agent to each subdigraph according to the decomposition order.
- Step 2. Initialize the system by setting $t=0$ and defining sets SAJ_t , NSJ_t , and SJ_t .
- Step 3. Schedule Manager gives a signal to all scheduling agents, having job in the set of schedulable jobs (SAJ_t), to schedule their jobs in the system.
- Step 4. The agent with the minimum makespan is the winner, and schedules his job.
- Step 5. Set $t=t+1$ and update the sets SAJ_t , NSJ_t , and SJ_t .
- Step 6. *If $SJ_t=J$ then GoTo Step 7 else GoTo Step 3.*
- Step 7. End.

The individual jobs are scheduled by corresponding agents using appropriate scheduling algorithms developed in the literature. [HE 02] showed that products with simple/linear assembly structure can be scheduled with the same makespan on $\{m, q=1\}$ and $\{m, q>1\}$ systems. Since each agent possesses a job with simple assembly structure, mixed integer programming can be used to achieve optimal scheduling of individual jobs (see [HE 01] for formulation).

4. The negotiation model for agent-based scheduling

In this section, the model of negotiation among scheduling agents is presented. The purpose of negotiation is to reach an agreement about the assignment of operations to machining and assembly equipment and to improve the overall schedule. The model defines a range of strategies and tactics that an agent can deploy to generate new scheduling offers. The model is based on computationally tractable algorithms.

The following assumptions are valid for the model: (i) there is no deadline for the scheduling decisions to be reported to schedule manager; (ii) each agent starts decision-making process, as soon as he receives the necessary information from the schedule manager; (iii) agents report about their decisions to the schedule manager as soon as their decision is made; (iv) schedule manager announces his decision to the agents as soon as his decision is made.

Before describing the negotiation mechanism the following definitions and notations are introduced:

a_j agent $j, j=1, \dots, k$;

t_{ij}^t completion time of the last machining operation on machine i ($i=1, \dots, m$)
obtained by agent a_j ($j=1, \dots, k$) at stage t ;

T_{ij}^t completion time of the last assembly operation on assembly machine i , ($i=1, \dots, q$)
obtained by agent a_j ($j=1, \dots, k$) at stage t ;

$Sa_j^t = (t_{1j}^t, t_{2j}^t, \dots, t_{mj}^t, T_{1j}^t, T_{2j}^t, \dots, T_{qj}^t)$ completion time-vector of schedule Sa_j^t obtained by
agent a_j ($j=1, \dots, k$) at stage t ;

$C_{\max}(j, t) = \max\{t_{1j}^t, t_{2j}^t, \dots, t_{mj}^t, T_{1j}^t, T_{2j}^t, \dots, T_{qj}^t\}$ maximum completion time of the
schedule obtained by agent a_j ($j=1, \dots, k$) at stage t ;

At stage t , each agent schedules his job in the $\{m, q\}$ system and obtains $(m+q)$ completion time-vector Sa_j^t . This $(m+q)$ vector is being reported by each agent in SAJ_t to SM . Based on this information, SM makes decision and identifies who schedules his job on the system at the current stage. The value of the following expression is evaluated:

$$C_{\max}(t) = \min_j \{C_{\max}(j, t)\}$$

Agent a_j is the winner at stage t if the following is true:

$$C_{\max}(j, t) = C_{\max}(t)$$

Once an agent becomes the winner at one stage, he enters the negotiation process with certain input. The players in the negotiation process are all the agents already having scheduled their jobs in the system. Basically the input brought by an agent to the negotiation process is the values of the decision variables that uniquely define the total schedule obtained that far. Once the winner is identified (or new player entering the negotiation process is identified) the set of *schedulable jobs* (SAJ), the set of *scheduled jobs* (SJ), and the set of non-schedulable jobs (NSJ) are updated. The current stage index is incremented by one. The winner, the agent who is scheduling his job at the current stage, observes his schedule, and decides if he is going to request for negotiation from the agents in the negotiation process. Listed below are some conditions that indicate the need of negotiation.

- The makespan obtained exceeds the estimated upper bound on makespan. In this case the agent last entered the negotiation process initiates a negotiation with others.
- Even if the makespan does not exceed the estimated upper bound it is close enough to that.

– If for any assembly operation scheduled by an agent starting time is greater than the completion time of its preceding operations (machining and assembly). Basically this fact indicates that the agent is delaying his job because of other agents. Hence, he may consider a negotiation with others.

Suppose agent a_1 was the winner at stage $t=1$, and agent a_2 becomes the winner at stage $t=2$. The overall schedule obtained so far is the combination schedule of the jobs owned by agent a_1 and agent a_2 . Let's assume that for some reason agent a_2 initiates a negotiation process in order to improve the schedule. As a result of negotiation, a series of feasible schedules is generated by combined efforts of agent a_1 and agent a_2 . The result is represented in the matrix presented below:

$$\begin{bmatrix} C_{\max}(\hat{1},1) & \times & \times & \times & \times & \times \\ C_{\max}(2,\hat{1}) & C_{\max}(\hat{2},2) & \times & \times & \times & \times \\ \times & C_{\max}(3,\hat{2}) & C_{\max}(\hat{3},3) & \times & \times & \times \\ \times & \times & C_{\max}(4,\hat{3}) & C_{\max}(\hat{4},4) & \times & \times \\ \times & \times & \times & C_{\max}(5,\hat{4}) & C_{\max}(\hat{5},5) & \times \\ \times & \times & \times & \times & \dots & \dots \end{bmatrix}$$

The elements of the matrix are maximum completion time of the combination schedule (job 1 and job 2). Agent a_2 enters the negotiation process, where agent a_1 is a player with some schedule (defined as schedule 1 of agent a_1), with some initial schedule (defined as schedule 1 of agent a_2).

When agent a_2 enters the negotiation process, the schedule offered by him is the best that he is able to find with consideration of decision variables defined at stage 1 by agent a_1 .

The first element $C_{\max}(\hat{1},1)$ of the matrix is the maximum completion time that agent a_2 obtained when he entered the negotiation process. The *cup* sign above number 1 (which indicates the schedule 1 by agent a_1) indicates that agent a_2 obtained schedule 1 without changing the values of decision variables defined by agent 1 in schedule 1. From this point on, a series of feasible schedules is obtained as described. Consider the element $C_{\max}(2,\hat{1})$. This is the maximum completion time that was obtained by agent a_1 (at this point it is considered as schedule 2 by agent a_2), where the cup above 1 indicates that agent a_1 obtained schedule 2 without changing the values of decision variables defined by agent a_2 in schedule 1. The rest of the matrix is constructed similarly.

At some point, the negotiation process should be terminated. The following conditions are recommended to be checked for termination:

- the *current maximum completion time* is equal to the estimated lower bound on makespan, which means the solution obtained is optimal;
- agents involved in the negotiation process are satisfied by the results obtained;

Once the negotiation process is terminated, the schedule obtained is passed to *SM* and *SM* announces about it to the agents in the *SAJ* list.

The best schedule obtained is the one corresponding to the element of the matrix that satisfies to the following condition:

$$\min_i \left\{ \min_j \{C_{\max}(i, j)\} \right\}$$

This schedule is reported to the Schedule Manager. The output schedule at stage t represents the input for stage $t+1$. This means that agent a_1 and agent a_2 will participate in the scheduling process at stage $t=3$ with the corresponding schedules, obtained as a result of negotiation process carried out at stage $t=2$, numbered as 1.

The scenario will change once agent a_3 enters the negotiation process. Now agent a_3 negotiates with agent a_1 and agent a_2 together, since they both have separate units to control in the schedule. And each one is responsible and authorized for changes concerning to the job's schedule owned by him. However, starting from stage $t=3$ the negotiation process keeps same tactics as at stage $t=3$ which is described below.

As the process continues, agent a_1 becomes the winner at stage $t=1$, agent a_2 becomes the winner at stage $t=2$, and agent a_3 becomes the winner at stage $t=3$. The overall schedule obtained so far is the combination schedule of the jobs owned by agent a_1 , agent a_2 , and agent a_3 . In order to improve the schedule after entering the scheduling process, agent a_3 initiates a negotiation process with agent a_1 and agent a_2 . Each of them tries to improve the schedule by rescheduling the corresponding schedules they obtained as an output of stage $t=2$. The matrixes corresponding to negotiation processes are:

$$\begin{array}{cc} \text{Agent } a_3 & \text{Agent } a_3 \\ \text{Agent } a_1 \begin{bmatrix} C_{\max}(\hat{1}, \hat{1}, \hat{1}) \times \\ C_{\max}(2, \hat{1}, \hat{1}) \times \end{bmatrix} & \text{Agent } a_2 \begin{bmatrix} C_{\max}(\hat{1}, \hat{1}, \hat{1}) \times \\ C_{\max}(\hat{1}, 2, \hat{1}) \times \end{bmatrix} \end{array}$$

By comparing $C_{\max}(2, \hat{1}, \hat{1})$ and $C_{\max}(\hat{1}, 2, \hat{1})$ one can find out who generated better schedule. The following three cases are possible:

- $C_{\max}(2, \hat{1}, \hat{1}) \geq C_{\max}(\hat{1}, 2, \hat{1})$ and $C_{\max}(2, \hat{1}, \hat{1}) > C_{\max}(\hat{1}, \hat{1}, \hat{1})$. In this case agent a_2 becomes the winner and he gets the right to initiate the next negotiation process.
- $C_{\max}(\hat{1}, 2, \hat{1}) \geq C_{\max}(2, \hat{1}, \hat{1})$ and $C_{\max}(\hat{1}, 2, \hat{1}) > C_{\max}(\hat{1}, \hat{1}, \hat{1})$. In this case agent a_2 becomes the winner and he gets the right to initiate the next negotiation process.
- $C_{\max}(2, \hat{1}, \hat{1}) \leq C_{\max}(\hat{1}, \hat{1}, \hat{1})$ and $C_{\max}(\hat{1}, 2, \hat{1}) \leq C_{\max}(\hat{1}, \hat{1}, \hat{1})$. None of them is a winner and the negotiation process is over at the current stage. The stage index is incremented by one.

Suppose agent a_1 is the winner. He initiates a negotiation with agent a_2 and agent a_3 . Each of them tries to improve the current overall schedule. At the current moment the overall makespan obtained is $C_{\max}(1, 1, 1)$.

$$\begin{array}{cc}
 \text{Agent } a_1 & \text{Agent } a_1 \\
 \text{Agent } a_2 \begin{bmatrix} C_{\max}(\hat{1}, \hat{1}, 1) & \times \\ C_{\max}(\hat{1}, 2, \hat{1}) & \times \end{bmatrix} & \text{Agent } a_3 \begin{bmatrix} C_{\max}(\hat{1}, \hat{1}, 1) & \times \\ C_{\max}(\hat{1}, \hat{1}, 2) & \times \end{bmatrix}
 \end{array}$$

Considering three corresponding possible cases discussed at the previous iteration the winner is identified and the next iteration is started, or the stage counter t is incremented by one. Eventually the last agent will enter the scheduling process and after necessary negotiations the final schedule will be obtained.

Next, the formal agent-based scheduling algorithm with negotiation mechanism is presented. In addition to the notations and definitions prior introduced the counter of iteration is defined by c .

4.1. Agent-based scheduling algorithm with negotiation

- Step 1. Decompose digraph into a set of simple digraphs and assign an agent to each job according to the decomposition order.
- Step 2. Initialize the system by setting $t=0$ and defining sets SAJ_t , NSJ_t , and SJ_t .
- Step 3. Schedule Manager gives a signal to all scheduling agents, having job in the set of schedulable jobs (SAJ_t), to schedule their jobs in the system.
- Step 4. The agent who generates the shortest combined makespan with the scheduled jobs is the winner and schedules his job in the system.
- Step 5. Set $t=t+1$, $c=0$ and update the sets SAJ_t , NSJ_t , and SJ_t .

Step 6. The last agent entering the scheduling process checks if negotiation is necessary. If so, he initiates a negotiation with the other agents in the scheduling process. The counter of iteration is set $c=c+1$. A series of feasible schedules is generated with each agent in the scheduling process, and makespan matrixes are constructed.

If no negotiation is conducted then GoTo Step 3.

Step 7. The agent generating the minimum makespan with the agent last entering the scheduling process is the winner in that iteration.

Set last agent entering the scheduling process = winner at iteration c.

Step 8. Check for termination of negotiation.

If negotiation is terminated GoTo Step 9 else GoTo Step 6

Step 9. *If $SJ_i=J$ then GoTo Step 10 else GoTo Step 3.*

Step 10. End.

Note that the decomposition in Step 1 is performed by the digraph decomposition algorithm described in [KUS 89].

An illustrative example of agent-based scheduling algorithm *with* negotiation is presented in Appendix I.

5. Objective measures of effectiveness

In assessing the benefits of agent-based scheduling in terms of solution quality one should notice that solution quality strongly depends on the scheduling techniques used by individual agents for scheduling their tasks. Also note that each subsequent sub-stage in a negotiation process gives a better output than its previous sub-stage. Thus, the longer the negotiation process lasts the better solutions will be obtained. Next, lower and upper bound estimates on makespan are developed.

Suppose a product with corresponding digraph is going to be scheduled on a $\{m, q\}$ manufacturing system.

Let P_1, P_2, \dots, P_n be all part nodes in the digraph, and A_1, A_2, \dots, A_N be all assembly nodes in the digraph (see example in Figure 2). The following notations are introduced:

k = number of agents or decomposed sub-digraphs;

n_i = number of part nodes in the sub-digraph corresponding to the sub-problem handled by agent a_i .

N_i = number of assembly nodes in the sub-digraph corresponding to the sub-problem handled by agent a_i .

$P_1^i, P_2^i, \dots, P_{n_i}^i$ are part nodes in sub-digraph corresponding to the sub-problem handled by agent a_i .

$A_1^i, A_2^i, \dots, A_{N_i}^i$ are assembly nodes in sub-digraph corresponding to the sub-problem handled by agent a_i .

$t_{(1)}^i \leq t_{(2)}^i \leq \dots \leq t_{(n_i)}^i$ are ordered processing times corresponding to part nodes $P_1^i, P_2^i, \dots, P_{n_i}^i$.

$T_{(1)}^i \leq T_{(2)}^i \leq \dots \leq T_{(N_i)}^i$ are ordered processing times corresponding to assembly nodes $A_1^i, A_2^i, \dots, A_{N_i}^i$.

Then lower bound on makespan is calculated using the above defined terminology by equation [1]. Basically, what it represents is the best-case performance of the scheduling, when assembly starts right after the shortest machining time in the digraph and after that there is no idle time on assembly line.

$$LB = \min_{i=1}^k \{t_{(1)}^i\} + \frac{\sum_{i=1}^{N-1} t(A_i)}{q} + t(A_N) \quad [1]$$

For developing upper bound on makespan the following notations are introduced. Let $\lceil (\bullet) \rceil$ be the smallest greater or equal integer of evaluated expression (\bullet) , then define $k_i = \left\lceil \left(\frac{n_i}{m} \right) \right\rceil$ as average number of parts per machine for agent a_i and $r = \left\lceil \left(\frac{k}{q} \right) \right\rceil$ average number of agents/individual jobs per assembly line.

The maximum completion time for the schedule by agent a_i can be computed using the following formula $T_i = \sum_{j=0}^{k_i-1} t_{(n_i-j)}^i + \sum_{j=1}^{N_i} T_j^i$. The expression $T^i = \sum_{j=1}^{N_i} T_j^i$ represents total

assembly time corresponding to the job possessed by agent a_i . The expression $\sum_{j=0}^{k_i-1} t_{(n_i-j)}^i$ represents the machining time, when first k_i largest machining operations in the job possessed by agent a_i are assigned to the same machine, which corresponds to the worst possible performance. Then, T_i is the completion time, when assembly operations start right after all machining is completed.

The ordered total assembly time corresponding to each agent is $T^{(1)} \leq T^{(2)} \leq \dots \leq T^{(k)}$.

Using these notations and formulations the upper limit of total completion time is calculated by equation [2].

$$UB = \sum_{i=1}^k \sum_{j=0}^{k_i-1} t_{(n_i-j)}^i + \sum_{j=0}^{r-1} T^{(k-j)}, \text{ where } t_{(n_i-j)}^i = \begin{cases} t_{(n_i-j)}^i, & \text{if } n_i - j \geq m; \\ t_{(n_i)}^i, & \text{if } n_i - j < m \end{cases} \quad [2]$$

5.1. Computational experiments

To show the effectiveness of the proposed agent-based scheduling methodology in terms of solution quality computational experiments were conducted. We chose the system of consisting of one machine and one assembly station because optimal algorithms for scheduling on these systems were developed by [KUS 89] and we were able to compare the results obtained by applying agent-based approach with the optimal scheduling solutions.

The developed agent-based approach was coded in C++ and used for testing problems. A total of 4 types of problem with different assembly sequences were tested. For each testing problem, 5 instances were generated, and for each instance, assembly sequence, number of part nodes, number of subassembly nodes, maximum level of assembly, machining times, and assembly times were randomly generated. The data was generated based on the real assembly application information from industrial assembly handbooks ([NOF 96]; [BOO 92]; and [LOT 89]). The average size of the problems corresponds to 37 part nodes, 26 assembly nodes, and 7 assembly levels. The machining and assembly times were randomly generated that follow uniform distribution $U(7, 25)$ and $U(10, 30)$ respectively for the first type of the problems. For the second 5 instances of the problems the generated machining and assembly times follow uniform distribution $U(6, 14)$ and $U(6, 19)$ respectively. The third sample followed $U(3, 12)$ and $U(4, 12)$. And finally, for the fourth sample we had $U(5, 16)$ and $U(6, 20)$ correspondingly. The results of the computations are provided in Table 1.

Problem instance	No.	C_{\max}	C_{AB}	C_{AB-N}	C_{AB}/C_{\max}	C_{AB-N}/C_{\max}
Machining times: U(7, 25)	1	683	704	683	1.031	1.000
	2	638	641	638	1.005	1.000
	3	631	714	631	1.132	1.000
Assembly Times: U(10, 30)	4	582	640	598	1.100	1.027
	5	585	619	605	1.058	1.034
Machining times: U(6, 14)	1	404	434	422	1.074	1.045
	2	399	440	399	1.103	1.000
	3	392	417	407	1.064	1.038
Assembly Times: U(6, 19)	4	391	437	391	1.118	1.000
	5	405	443	408	1.094	1.007
Machining times: U(3, 12)	1	320	335	321	1.047	1.003
	2	335	345	337	1.030	1.006
	3	329	342	330	1.040	1.003
Assembly Times: U(4, 12)	4	301	317	310	1.053	1.030
	5	315	326	323	1.035	1.025
Machining times: U(5, 16)	1	487	504	502	1.035	1.031
	2	435	442	434	1.016	1.000
	3	443	452	448	1.020	1.011
Assembly Times: U(6, 20)	4	432	453	443	1.049	1.025
	5	464	492	478	1.060	1.030

Table 1. The results of the computational experiment

To evaluate the effect of negotiation in agent-based scheduling, for each problem we tested, the completion time of product was obtained without applying negotiation. The results are presented in Table 1. The notations in Table 1 are as follows:

C_{\max} – completion time corresponding to optimal schedule;

C_{AB} – completion time obtained by applying agent-based approach *without* negotiation;

C_{AB-N} – completion time obtained by applying agent based approach *with* negotiation;

The ratios in Table 1, i.e., C_{AB}/C_{\max} , C_{AB-N}/C_{\max} , represent the effectiveness of the tested agent-based approaches.

Comparing the results in ratio columns corresponding to agent based approach with and without application of negotiation, one can see that application of negotiation dramatically improves the solution quality. Agent-based scheduling *without* applying negotiation provided 0.5%-13.2% deviation in completion time from the optimal schedule when agent-based scheduling *with* application of negotiation provided 0.0%-4.5% deviation in completion time from the optimal schedule.

Concluding the experimental results, we can say that agent-based scheduling approach presented in this paper provides optimal and near optimal solutions.

6. Conclusions

In this paper, an agent-based approach was designed for manufacturing system scheduling in an agile manufacturing environment. The general framework of the agent-based approach was presented. The successful implementation of agent-based approach strongly depends on how the system is decomposed into interrelated subsystems and the negotiation among the agents so that the integrity of the system is not violated. A decomposition method was proposed for successful decomposition of large size scheduling problems into sub-problems. Then negotiation mechanism was developed and incorporated into decision-making process in a multi agent environment. A lower and upper bound on maximum completion time were.

The decomposition method and agent-based tool were designed in a way that each agent is assigned a separate individual job to be scheduled in the system according to the precedence relationships of other agents' jobs. If any changes are made by users in component designs, or assembly structures the product designs, rescheduling can be done easily by corresponding agents in the scheduling system. In this sense, the schedule obtained by agent-based approach is robust. The developed approach integrates data and decisions associated with several entities within a scheduling system. This approach can be used to model and solve large-scale scheduling problems in an agile manufacturing environment. To demonstrate the effectiveness of proposed methodology, computational experiments were conducted. The results of tested problems show that the scheduling method obtains optimal or near optimal solutions.

7. References

- [BOO 92] BOOTHROYD, G, *Assembly Automation and Product Design*, Marcel Dekker, New York, 1992.
- [COF 78] COFFMAN E. G., GAREY M. R. AND JOHNSON D. S. "An application of bin-packing to multiprocessor scheduling", *SIAM Journal of Computing*, vol. 7, 1978, p. 1–17.
- [DEC 87] DECKER K. S., "Distributed problem solving techniques: a survey", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC–17, no. 5, 1987, p. 729–740.
- [FRI 86] FRIESEN D. K., LANGSTON M. A., "Evaluation of a multifit-based scheduling algorithm", *Journal of Algorithms*, vol. 7, 1986, p. 35–59.

- [GAL 88] GALLIERS J. R., (1988). "A theoretical framework for computer models of cooperative dialogue. Acknowledging multi-agent conflict", *PhD thesis*, Open University, UK.
- [GAR 78] GAREY M. R., JOHNSON D. S., "Strong NP-completeness results: motivations, examples and implications", *Journal of Association of Computing and Mathematics*, vol. 25, 1978, p. 499–508.
- [GEN 94] GENESERETH M. R., KETCHPEL S. P., "Software agents", *Communications of the ACM*, vol. 37, no. 7, 1994, p. 48–53.
- [GRA 69] GRAHAM R. L., "Bounds on multiprocessing timing anomalies", *SIAM Journal of Applied Mathematics*, vol. 17, 1969, p. 416–429.
- [HE 02] HE, D., BABAYAN A., "Scheduling manufacturing systems for implementation of delayed product differentiation strategy for agile manufacturing", *International Journal of Production Research*, 2002 (to appear).
- [HE 01] HE D., BABAYAN A., KUSIAK A., "Scheduling manufacturing systems in an agile manufacturing environment", *Robotics and Computer Integrated Manufacturing*, vol. 17, 2001, p. 87–97.
- [HE 96] HE D., KUSIAK A., "Performance analysis of modular products", *International Journal of Production Research*, vol. 34, no. 1, 1996, p. 253–272.
- [JEN 95] JENNINGS N. R., CORERA J., LARESGOITI I., MAMDANI E. H., PERRIOLAT F., SKAREK P., VARGA L. Z., "Using ARCHON to develop real world DAI applications for electricity transportation management and particle accelerator control", *IEEE Expert-Special Issue on Real World Applications of DAI*, 1995.
- [JEN 93] JENNINGS N. R., VARGA L. Z., AARNTS R. P., FUCHS J, SKAREK P., "Transforming standalone expert systems into a community of cooperating agents. International", *Journal of Engineering Applications of Artificial Intelligence*, vol. 6, no. 4, 1993, pp. 317–331.
- [JOH 54] JOHNSON S. M., "Optimal two and three-stage production schedule with setup times included", *Naval Research Logistics Quarterly*, vol. 1, no. 1, 1954, p. 61–68.
- [KUS 89] KUSIAK A., "Aggregate scheduling of a flexible machining and assembly system", *IEEE Transactions on Robotics and Automation*, vol. 5, no. 4, 1989, p. 451–459.
- [LOT 89] LOTTER B., *Manufacturing Assembly Handbook*, Butterworth, London, 1989.
- [MIN 86] MINSKY M., *The Society of Mind*, Simon and Schuster, New York, NY, 1986.
- [NOF 96] NOF S., Y. WILHELM W. E., WARNECKE H.-J., *Industrial Assembly*, Chapman Hall, New York, NY, 1996.
- [PAR 95] PARUNAK H. V. D., "Applications of distributed artificial intelligence in industry", In *Foundations of Distributed Artificial Intelligence*, editors G. M. P. O'HARE and N. R. JENNINGS, Wiley, 1995.

- [RAB 94] RABELO R. J., CAMARINHA-MATOS L. M., "Negotiation in multi-agent based dynamic scheduling", *Journal on Robotics and Computer Integrated Manufacturing*, vol. 11, no. 4, 1994, p. 303-310.
- [RAB 99] RABELO R. J., CAMARINHA-MATOS L. M. AND AFSARMANESH H., "Multi-agent-based agile scheduling", *Robotics and Autonomous Systems*, vol. 27, 1999, p. 15-28.
- [ROS 85] ROSENSCHEIN J. S., GENESERETH M. R., "Deals among rational agents", *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)*, 1985, p. 91-99, Los Angeles, CA.
- [SCH 96] SCHWUTTKE U. M., QUAN A. G., "Enhancing performance of cooperating agents in real time diagnosis systems", *Proceedings of 13th International Joint Conference on Artificial Intelligence*, Chamberry, France, 1993, pp 332-337.
- [SEN 99] SHEN W., NORRIE D. H., "Agent-based systems for intelligent manufacturing: A state of the art survey", *Knowledge and Information Systems, an International Journal*. vol. 1, no. 2, 1999, pp. 129-156.
- [SIC 92] SICHTMAN J., DEMAZEAU Y., "When can knowledge-based systems be called agents?", *Proceedings IX Brazilian Symposium on Artificial Intelligence*. Rio de Janeiro, Brazil, 5-8 October, 1992.
- [SOU 97] SOUSA P., RAMOS A. "A dynamic scheduling Holo for manufacturing systems" *Proceedings of the Second World Congress on Intelligent Manufacturing Processes and Systems*. Budapest, Hungary, 10-13 June, 1997.
- [SPR 95] SPRUMONT F., GHEDIRA K, MÜLLER J.-P., "AMACOLA: a multi agent approach to the design of flexible lines: Preliminary report", *Proceedings of IJCAI workshop on Intelligent Manufacturing Systems*, Montreal, Canada, 1995.
- [WEI 94] WEIHMYER R., VELTHUIJSEN H., "Application of distributed AI and cooperative problem solving to telecommunications", In *AI Approaches to Telecommunications and Network Management*, editors J. Liebowitz and D. Prereau, IOS Press, 1994.

Appendix I: Illustrative example of agent based scheduling algorithm

Consider a product to be produced in $\{m=2, q=1\}$ manufacturing system. The assembly sequences of the product is shown in Figure 3. The machining and assembly times are provided in Table 1.

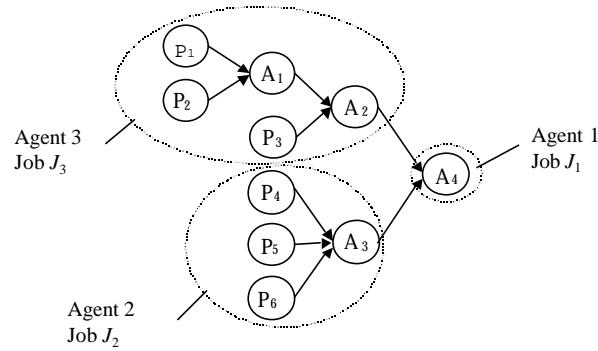


Figure 3. Assembly structure of a product

Part	P ₁	P ₂	P ₃	P ₄	P ₅	P ₆
Machining Time	6	4	6	1	6	9
Assembly	A ₁	A ₂	A ₃	A ₄	–	–
Assembly Time	3	4	3	5	–	–

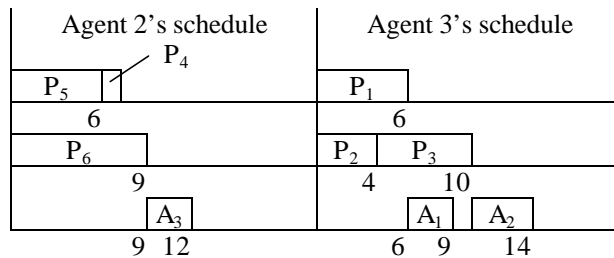
Table 1. Machining and assembly times

To decompose the digraph, we remove the root node A_4 . All the resulted sub-digraphs have simple assembly structure. After decomposition, an agent is assigned to each job. The set of following jobs is defined: $J = \{J_1, J_2, J_3\}$.

Initialize the system:

$$\begin{aligned}
 t=0 \quad SAJ_0 &= \{J_2, J_3\} \\
 NSJ_0 &= \{J_1\} \\
 SJ_0 &= \{\emptyset\}
 \end{aligned}$$

At this stage the schedule manager gives a signal to the agents of all the schedulable jobs to find the best possible schedule (shortest makespan) with the consideration of scheduled jobs (the set of scheduled jobs is empty at $t=0$). The Gantt charts of schedules obtained by individual agents having job in SAJ_0 are provided below.

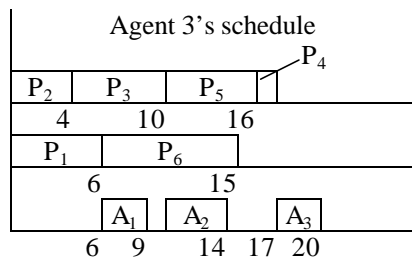


Agent a_2 is the winner, since his schedule has the shortest makespan.

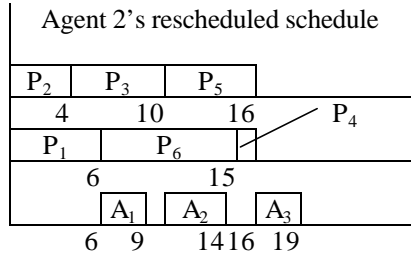
The system is updated as follows:

$$\begin{aligned}
 t=1 \quad SAJ_1 &= \{J_3\} \\
 NSJ_1 &= \{J_1\} \\
 SJ_1 &= \{J_2\}
 \end{aligned}$$

The Gantt chart of schedule obtained by the only agent having job in SAJ_1 is provided below.



Agent a_2 realizes that some changes happened to his schedule, the completion times of operations were changed. He reschedules his schedule without changing any assignment made by agent a_3 . The final schedule is shown below.

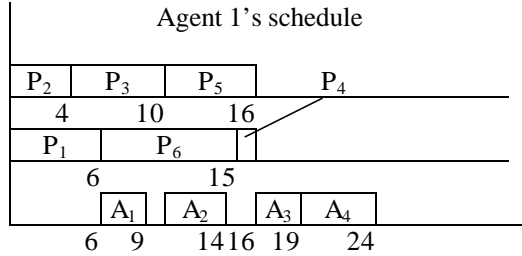


Here it is reasonable to terminate negotiation process, since the total machining time was broken evenly between 2 machines.

Update the system as follows:

$$\begin{aligned}
 t=2 \quad SAJ_2 &= \{J_1\} \\
 NSJ_2 &= \{\emptyset\} \\
 SJ_2 &= \{J_2, J_3\}
 \end{aligned}$$

The schedule manager gives a signal to all of the schedulable jobs' agents to find the best possible schedule with the consideration that the jobs in SJ_2 are scheduled. The last scheduling agent a_1 schedules his job and obtains the schedule presented below.



Scheduling agents a_2 and a_3 do not find any change in their scheduled operations completion times, hence there is no need for rescheduling.

Update the system as follows:

$$\begin{aligned}
 t=1 \quad SAJ_3 &= \{\emptyset\} \\
 NSJ_3 &= \{\emptyset\} \\
 SJ_3 &= \{J_1, J_2, J_3\}
 \end{aligned}$$

Since $SJ_3=J=\{J_1, J_2, J_3\}$, the scheduling process is over.