Using multi-agent architecture in FMS for dynamic scheduling

KHALID KOUISS,[1*] HENRI PIERREVAL[1] and NASSER MEBARKI[2]

[1]*Laboratoire de Recherche en Systèmes de Production, Institut Français de Mécanique Avancée, Campus des Cézeaux, BP 265, 63175 Aubière Cedex, France*
[2]*Laboratoire PRISMa, Université Claude Bernard, Bât 710, 43 Boulevard du 11 Novembre, 69622 Villeurbanne Cedex, France*

Received July 1995 and accepted June 1996

The proposed scheduling strategy is based on a multi-agent architecture. Each agent of this architecture is dedicated to a work centre (i.e. a set of resources of the manufacturing system); it selects locally and dynamically the most suitable dispatching rules. Depending on local and global considerations, a new selection is carried out each time a predefined event occurs (for example, a machine becomes available, or a machine breaks down). The selection depends on: (1) primary and secondary performance objectives, (2) the operating conditions, and (3) an analysis of the system state, which aims to detect particular symptoms from the values of certain system variables. We explain how the scheduling strategy is shared out between agents, how each agent performs a local dynamic scheduling by selecting an adequate dispatching rule, and how agents can coordinate their actions to perform a global dynamic scheduling of the manufacturing system. Each agent can be implemented through object-oriented formalisms. The selection method is improved through the optimization of the numerical thresholds used in the detection of symptoms. This approach is compared with the use of SPT, SI^X, MOD, CEXSPT and CR/SPT on a jobshop problem, already used in other research works. The results indicate significant improvements.

Keywords: Dynamic scheduling, dispatching rules, flexible manufacturing systems, multi-agent system, simulation-optimization, object-oriented models

1. Introduction

The dynamic scheduling of manufacturing systems is concerned with the allocation of jobs to the resources in real time. This allocation is made according to the state of the shopfloor (e.g. breakdown of a machine, availability of a resource, or existence of bottlenecks) and the production objectives (e.g. reduce the number of jobs in progress, or reduce the tardiness). One of the most common approaches to dynamic scheduling of the jobs to process is to use dispatching rules (DRs). Dispatching rules can be very simple or extremely complex. Examples of simple dispatching rules are: 'select a job at random' or 'select the job with the longest waiting time'. A more complex example might be 'select the job with the shortest due date whose customer's inventory is less than a specific amount'.

Numerous DRs exist, but research in recent decades has demonstrated that there is no one DR that is globally better than the others (Blackstone *et al.*, 1982; Kiran and Smith, 1982; Montazeri and van Wassenhove, 1990). Their efficiency depends on the performance criteria considered, and on the operating conditions (e.g. shop load, tightening of due dates, or existence of bottlenecks).

We propose an approach based on a multi-agent architecture. Each agent selects locally and dynamically the DR that seems the most suited to the operating conditions, to the production objectives, and to the current shop status. Because the shop status changes over time, each agent analyses the system state each time an event occurs (e.g. a machine becomes available, or an urgent job arrives).

In this paper, we first present the general principles of multi-agent systems, and we focus on the benefits of this approach in the production management area. Next, we present the way in which the dynamic scheduling decisions can be shared between agents, and the role of these

*Author to whom all correspondence should be addressed.

0956-5515

agents is highlighted. Finally, the benefits of this approach are demonstrated through the example of a job-shop system.

2. Agents and multi-agent systems

2.1. *Definitions*

The use of a multi-agent architecture allows decisions to be taken in a decentralized way. In artificial intelligence, this approach appears to be well suited to complex problems, especially those with a great number of interactions between components, and for which classical incremental methods cannot provide good results. The multi-agent method allows one to solve subproblems locally with an agent, and to propose a global solution as a result of interactions between the different agents.

Several researchers have proposed formal definitions for agents and multi-agent systems. We retain those proposed by Ferber (1993):

(1) An *agent* is a real or a virtual entity able to act on itself and on the surrounding world, generally populated by other agents. To perform its actions, this entity contains a partial representation of its environment, and can communicate with other agents of this environment. Its behaviour is a result of its observations, its knowledge and its interactions with the world and other agents. An agent has several interesting features:

(a) it has capabilities of perception and a partial representation of the environment;
(b) it can communicate with other agents;
(c) it can reproduce son agents;
(d) it has its own objectives and an autonomous behaviour;

(2) A *multi-agent system (MAS)* is an artificial system composed of a population of autonomous agents, which cooperate with each other to reach common objectives, while simultaneously each agent pursues individual objectives.

2.2. *Agent structure*

We can split an agent into three layers, as depicted in Fig. 1:

(1) *The static knowledge layer*: contains knowledge on itself and on the other agents. This is an agent's specific memory, used to memorize its observations and its knowledge concerning its environment (social knowledge);

(2) *The expertise layer*: contains knowledge that represents treatments and actions that an agent is able to carry out and which can be described in various forms (e.g. algorithms, production rules, frames, or logical expressions). This layer constitutes the agent know-how;

(3) *The communication layer*: includes the communication tools. They describe the communication protocols

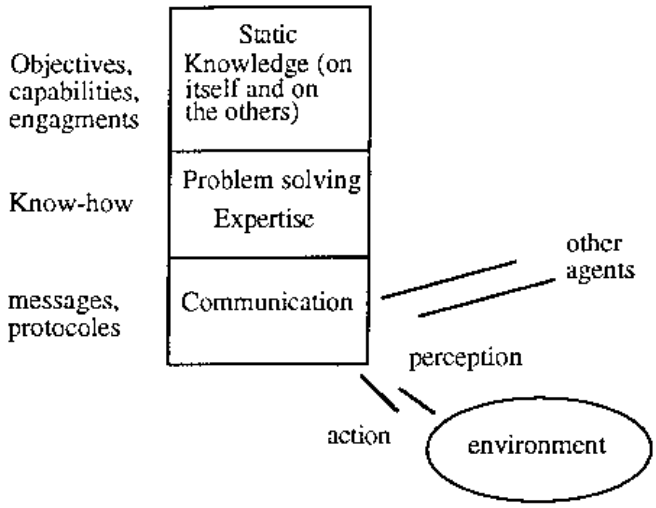

Fig. 1. Agent structure.

between the agent on one side and some other agents and resources of the environment on the other side. This layer characterizes the way that the agent takes into account the messages it receives. This layer also allows the agent to act and to apprehend the environment changes.

2.3. *Multi-agent structure and production management*

For the production management of a manufacturing system, many decisions have to be taken to reach the production objectives (e.g. planning decisions, scheduling decisions, and control decisions). Indeed, these decisions must be periodically updated in order to take into account changes in the production system (e.g. a machine breakdown, worker absences, or the arrival of an urgent job). The use of a multi-agent architecture allows one to share out all these decisions between several agents in a hierarchical manner. Each agent is in charge of specific decisions (Chandra and Talavage, 1991; Iffnecker *et al.*, 1991; Baptiste and Manier, 1993; Kwok and Norrie, 1993; Parunak, 1993; Barbuceanu and Fox, 1994; Lefrançois and Montreuil, 1994; Tacquard *et al.*, 1994; Trentesaux and Tahon, 1995; Ouzrout, 1996). Unfortunately this structure presents some disadvantages, due mainly to possible contradictory decisions of agents that can lead to a global lock of the system (Ayel, 1994; Attoui *et al.*, 1995).

The allocation of decisions to agents can be made according to several criteria, listed below:

(1) Technological criterion: for example, an agent may be dedicated to resources using the same communication protocol;

(2) Topological criterion: for example, an agent may be dedicated to resources close (in distance) to each other;

(3) Functional criterion: for example, an agent may be dedicated to a particular function (e.g. quality function,

monitoring function, or scheduling function). This paper emphasizes the dynamic scheduling function;

(4) Organizational criterion: the manner in which the works are allocated to each agent.

3. A dynamic scheduling approach based on a multi-agent structure

3.1. *Principles of the proposed approach*

The dynamic scheduling is supported by a multi-agent architecture. Each agent of the system is in charge of a work centre of the manufacturing system. An agent solves the scheduling problem by selecting dynamically the most adequate DR to apply locally. Examples of DRs that can be applied are: shortest processing time (SPT), smallest critical ratio (SCR), earliest due date (EDD), conditional expeditive shortest processing time (CEXSPT), and critical ratio shortest processing time (CR/SPT) (Mebarki, 1995). To select a DR, the agents take into account primary and secondary objectives (e.g. reduce the mean flow time and reduce the percentage of tardy jobs), the state of the work centre (e.g. length of the waiting job queues, or availability of resources), and information received from other agents.

In order to take into account the changes of the system state, the application of new DRs is envisaged by agents each time a triggering event occurs; that is, each time a resource becomes available, a new job arrives, or a job leaves the system.

It has already been shown that combinations of different DRs could perform better than applying the same DR to all the work centres (Barrett and Barman, 1986). In a multi-agent architecture, each agent takes its decisions in an independent way, so in a given time different DRs may be applied to different work centres.

The selection of the DR applied by each agent is carried out through two steps using the following strategy (Pierreval and Mebarki, 1997).

3.1.1. *Step 1: Detection of an active symptom*

The system status is analysed to try to detect predefined symptoms. This is done using knowledge of the following form:

If [condition about state variables] then [active symptom]

An example of such a rule is:

If [job due date − current time −
remaining processing time < α] then [active 'job tardy']

where α is called a *threshold*. A symptom becomes active when an observed variable (e.g. utilization rate of resources, waiting time of jobs, or length of queues), has exceeded a predefined threshold. This means that the system might be deviating from its production objectives. Symptoms may concern the behaviour of the whole system [global symptoms detected by the supervisory agent (see section 3.2), e.g. 'Too many tardy jobs'], or the behaviour of a particular work centre [local symptoms detected by a simple agent dedicated to a work centre (see section 3.2) e.g. 'Station S becomes bottleneck'].

Thresholds depend on the particular scheduling problem, and cannot be generally predefined. Pierreval (1992) has shown that these thresholds can have a great impact on the performance of this scheduling method. Thus thresholds need to be tuned for each agent using an optimization procedure (Pierreval and Mebarki, 1997).

3.1.2. *Step 2: Choice of DRs*

The DRs to apply are chosen from a set of pre-selected DRs, using rules of the following forms:

If [conditions about the objectives
and/or conditions about information received
from other agents
and/or conditions about the state of the
local work centre
and/or conditions about the active symptoms]
then [apply {selected DR} to the considered queue]

An example of such a rule is:

If [the primary objective = 'reduce the mean flow time'
and no 'tardy jobs'
and no 'urgent jobs']
then [apply SPT]

3.2. *Organization of the agents for dynamic scheduling*

The manufacturing system is composed of several work centres, each one made up of one or several resources. The multi-agent architecture is shown in Fig. 2.

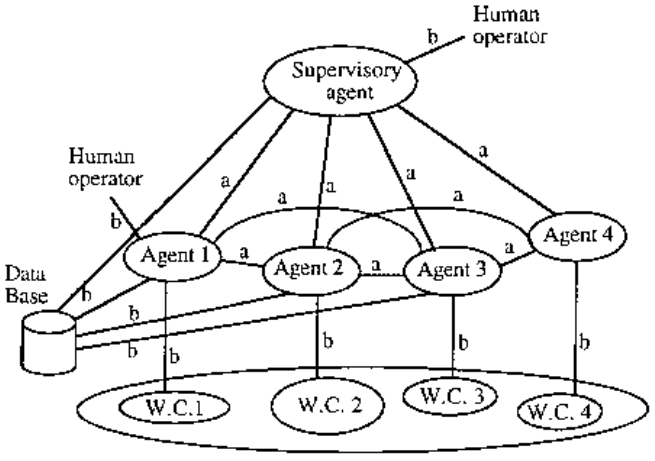

Fig. 2. Multi-agent architecture. WC: work centre.

In this architecture we distinguish a supervisory agent and several simple agents each dedicated to a specific work centre. Those agents perform two types of communication:

(1) Communication with other agents (type a in Fig. 2). Examples of this type of communication are: a request to apply a specific DR, information about active symptoms, and a request of the state of another agent;
(2) Communication with the environment (type b in Fig. 2). Examples of this type of communication are: execution of a program, reading of the state of a variable, orders from a human operator, and exchange with the database system.

The role of the supervisory agent is to monitor the global state of the manufacturing system. This agent has only an external vision of the state of other work centres; dedicated agents keep it informed using messages. It can detect global symptoms, and can impose particular DRs to agents controlling the work centres if it considers this necessary to satisfy the global objectives. For example, if it notices the global symptom. 'The number of jobs that become tardy is too high', and if the objective is to reduce the mean flowtime of jobs, then it imposes SPT to all agents.

The agent allocated to a work centre is in charge of the scheduling of jobs inside the centre. This agent manages its own waiting job queues. The selection of DRs is made according to the two steps described above, and depends on the system state, the orders received from the supervisory agent, and the global objectives of the manufacturing system (e.g. reduce the mean tardiness). This selection can be very simple when the supervisory agent imposes a particular DR on the agent.

3.3. *Implementation of the dynamic scheduling approach in an agent*

Each agent has the structure presented in Fig. 1, and can be implemented using object-oriented formalisms (Kwok and Norrie, 1993; Lefrançois and Montreuil, 1994). This structure is based on the three layers described below.

3.3.1. *Static and social knowledge layer*

This layer contains such pieces of knowledge as:

(1) Threshold values used in the rules to activate the predefined symptoms. These values may change during the life of the agent according to a learning procedure;
(2) Data about DRs that the agent can select;
(3) Data about capabilities of other agents (for example, the supervisory agent has data about capabilities of all the agents dedicated to the work centres of the manufacturing system).

3.3.2. *Expertise layer*

This is the intelligent part of the agent. It is based on object methods representing production rules (as previously described), and uses data taken from the static and social knowledge layer and information received by the communication layer. The expertise concerns the application of the two steps described in Section 3.2. It checks the conditions to detect the predefined symptoms and, if necessary, selects a new, adequate DR. Then it applies it, using (from the static and social knowledge layer) the relevant data necessary for its application, and sends orders to the concerned entities, using the communication layer functionalities. In the case of the supervisory agent, an order can be a choice of a specific DR for another agent. In the case of an agent dedicated to a work centre, an order can be the start of a machining operation or the notification of the name of a new active symptom to the supervisory agent.

3.3.3. *Communication layer*

Agents need to communicate with the physical environment. We include in the term 'physical environment' each entity that is not an agent. This comprises all the resources (e.g. programmable controllers, robots, machining centres). Communication between an agent and the environment uses the machine's communication protocols (e.g. programmable controller protocols such as MODBUS, UNI-TELWAY or SINEC L2, or numerical control protocols). The communication layer of an agent must contain tools (e.g. drivers) to carry out the communication with all the resources of the work centre. This communication allows an agent, for example, to monitor machines (e.g. to start, stop, or download a program), to extract information in the database system, or to have information about the state of machines (e.g. alarm messages, and the state of a sensor that indicates the number of parts in a waiting job queue).

Agents also need to communicate with each other. This communication is performed by exchange of messages, and is supported by a communication network installed between the agents' host computers. The protocol can be a speech–act type (Trouilhet, 1993) such as KQML (Finin *et al.*, 1992), which is an agent communication language (ACL). This communication allows, for example, the supervisory agent to know the state of a waiting queue in a work centre, or to request the application of a given DR in another agent.

4. Simulation of the distributed dynamic scheduling

At present, this approach has not been implemented on a real FMS. In order to evaluate its performance, specific object-oriented simulation software was designed. Although this software is implemented in a simple program, it is based on the distributed dynamic scheduling that we have presented. To make the comparison as relevant as possible, we have chosen a jobshop model that has been already used by several researchers to compare DRs. Eilon

and Cotteril (1968) have used this model to test the effects of the SI^X rule, Baker and Kanet (1983) to demonstrate the benefits of the MOD rule, Baker (1984) to examine the interaction between dispatching rules and due-dates assignment methods, Russel *et al.* (1987) to analyse the effects of the CoverT rule comparatively with several other DRs, and Schultz (1989) to demonstrate the benefits of the CEXSPT rule.

The system is a four-machines jobshop. Each machine can perform only one operation at a time. The number of operations of the jobs processed in the system is uniformly distributed between two and six. The routeing of each job is random. More precisely, when a job leaves a machine and needs another operation, each machine has the same probability of being the next, except the one just released, which cannot be chosen. The processing times on machines are exponentially distributed, with a mean of 1. The arrival of jobs in the system is modelled as a Poisson process. The mean arrival rate of this process is equal to the shop utilization, which is defined as follows:

$$\text{Shop utilization} = \frac{1}{m}\sum_{k=1}^{m} \partial_k \qquad (1)$$

where ∂_k is the steady-state utilization rate of the kth resource, and m is the number of workstations in the shop. Due dates of jobs are determined using the TWK method (Baker, 1984).

The dispatching rules compared are: SPT, CEXSPT, CR/SPT, plus MOD and SI^X. These DRs seem to be accepted as being among the most efficient (see for example Baker, 1984; Russel *et al.*, 1987; Engell and Moser, 1992).

A simulation model of the system previously described was build using our simulation–optimization software. It was first run to tune the thresholds, and then to compare the dynamic change of DRs, managed by the multi-agent system (called SFSR), with regard to the two following pairs of objectives:

(1) Reduce the mean tardiness as a primary objective and reduce the mean flow time as a secondary objective (noted as SFSR1);
(2) Reduce the conditional mean tardiness as a primary objective and reduce the mean tardiness as a secondary objective (noted as SFSR2).

The experiments were conducted with a mean arrival rate of jobs of 0.9, which corresponds to a utilization rate of the resources of 90% (i.e. a high level of utilization). For this case, an average flow allowance of 30 time units represents tight due dates (i.e. an allowance factor k of 7.5), whereas an average flow allowance of 60 time units represents loose due dates (i.e. an allowance factor k of 15).

The simulation experiments have been designed in the same way as those of Schultz (1989): that is, the transient phase is estimated at 500 jobs, ten replications are carried out, and each run yields estimates of the performance measures collected on 5000 jobs. The performance measures collected are the mean flow time (MFT), the mean tardiness (MT), the conditional mean tardiness (CMT), and the proportion of tardy jobs (PT). These measures are averaged over the ten replications. The results are given in Tables 1 and 2.

Table 1. Comparison of the dynamic selection with various dispatching rules, with tight due dates

Rule	*MFT*	*MT*	*CMT*	*PT*
SPT	17.4	3.38	45.45	0.07
SI^x	22.6	1.97	9.16	0.2
MOD	23.0	1.87	12.11	0.14
CEXSPT	21.9	1.78	5.37	0.31
CR/SPT	22.2	1.42	8.73	0.15
SFSR1	21.1	1.5	9.18	0.15
SFSR2	23.4	1.86	4.5	0.39

Table 2. Comparison of the dynamic selection with various dispatching rules, with loose due dates

Rule	*MFT*	*MT*	*CMT*	*PT*
SPT	17.4	1.75	85.17	0.02
SI^x	19.9	0.04	4.55	0.007
MOD	25.9	0.13	6.48	0.01
CEXSPT	20.3	0.1	1.89	0.04
CR/SPT	23.8	0.03	2.81	0.007
SFSR1	21.1	0.03	3.33	0.007
SFSR2	26.3	0.21	1.29	0.16

In order to find out the significant differences between the strategies, we used a statistical test, based on 0.95 confidence intervals of the means of the differences between the results of each couple of strategies. This test is known as the paired-t confidence interval method (Law and Kelton, 1982).

Table 3 lists the best three strategies for each criterion, with loose or tight due dates (i.e. mean flow allowances of 60 and 30 time units), and the average of the percentage of the difference between the best and second-best strategies and the second and the third-best strategies. Note that all mean percentage differences are significant at $\alpha = 0.05$ except those indicated with an asterisk.

From these tables we can see that SFSR can overcome the dispatching rules on primary objectives. In the jobshop example, the SFSR2 strategy was found to perform the best on the conditional mean tardiness as the primary objective. The results of SFSR1 were the best on the mean tardiness, except in the case of tight due dates, where CR/SPT gives slightly better results.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.