A Multi Agent Scheduling Integrating Planning and Maintenance for Generalized Floor Shops

S. Bouzini-Hassini, F. Benbouzid-Sitayeb, S. Aknine

Abstract- **Multi-agent scheduling offers reactivity and distributed decision-making for floor shop control. Agents, which may represent any entities that act in production, negotiate to find best schedules. In this paper, we present a new multi-agent scheduling method that integrates both planning and maintenance activities. Actually, more than one plan can be generated for a job production. We suppose that plan selection must depend on information about machines maintenance and states to offer realistic schedules. Tests demonstrate that despite increasing time resolution with the agents' number, our system is scalable with reduced Cmax and machines failure risk.**

Index Terms: **Maintenance, Multi Agent Systems, Negotiation. Planning, Production Systems, Scheduling.**

I. INTRODUCTION

In manufacturing systems, scheduling is an essential activity where deciders must assign actions and deadlines to production resources with respect to time and cost constraints. Face to changes that may occur during production like arrival of unexpected tasks and machines breakdown, production control has to be more reactive and flexible to ensure robustness. A control strategy is said robust if it is insensitive to changes that may occur during production process. The scheduling problem is recognized to be NP Complete [1].

Multi Agent Systems (MAS) have been used to solve the scheduling problem [2] [3]. Autonomy, reactivity and reasoning properties allow agents to be an efficient support for decision making. Agents may represent any entities acting in production (jobs, products, machines, etc). They negotiate to get an agreement on tasks and deadlines allocation over the machines.

Planning is an important activity closed to scheduling. It generates, for a given job, tasks that must be executed, the needed machines and tools to its realization. For every job to be realized, at least one plan can be identified. Always, several plans can be generated for one job. In generalized shops, machines can be identical or with different options that characterize one machine over the others. This concept is known as Flexibility [6]. A job realization is said flexible if it has several plans that can be followed. In recent years, integrating planning and scheduling has attracted big interest because it allows deciders to choose the preferred solution among several alternatives. Job plan selection is based on scheduling results. The best plan corresponds to the best schedule.

Manuscript received November 30, 2011; revised December 22, 2011. S. Bouzidi-Hassini is with the National School of Computing; Algiers, Algeria. (phone: 213699679953; e-mail: s_hassini@ esi.dz). F.Benbouzid-Sitayeb is also with National school of computing; Algiers; Algeria. (e-mail: f_sitayeb@esi.dz). S. Aknine is with Claude Bernard University, Lyon, France. (e-mail: samir.aknine@ recherche.univ-lyon1.fr).

Actually, after a certain period of use, machines can break down and then provoke production disturbance. Traditionally, maintenance intervenes after a machine failure in order to restore its good state. This maintenance category is called corrective maintenance. However, preventive maintenance has been introduced to carry out maintenance operations in machines and equipments before the failure takes place, and at fixed time intervals previously established. The objective of this latter is to prevent failures before they happen and therefore to minimize the probability of failure. As a consequence, preventive maintenance activities have to be scheduled at their turn with respect of production schedules because they use the same machines.

In this paper, we propose a new scheduling method that integrates both planning and maintenance activities. Indeed, introducing machine-states and maintenance activities into scheduling and planning help deciders to choose realistic plans where risks of breakdown and machine unavailability are reduced. The proposed method called MASMPLAM (for Multi Agent Scheduling Method based on Planning and Maintenance data) is based on multi agent systems. Three types of agents are considered: Job Agent, Plan Agent and Machine Agent.

The rest of the paper is organized as follows. In the following section, we present a survey of related work. In section 3, we develop the suggested scheduling method. Section 4 presents experimental results. Finally, section 5 summarizes contributions of this paper.

II. RELATED WORK

Scheduling problem has been widely studied and many methods have been proposed. After our literature review, we find that research in this area can be classified according to three criteria:

- *Adopted resolution approaches*: exact [7] or approached methods [8]. Exact approaches like branch-and-bound algorithms are efficient for problems with small number of machines. They give exact and optimal solutions. In case of problems with large number of machines, approached methods like genetic algorithms are preferred because exact methods fail to find solutions within reasonable computational times. Approached methods don't generate optimal schedules but acceptable ones.

- *Deterministic or stochastic scheduling*: a scheduling is deterministic if its initial data (job and task number, machine-state…. etc) are supposed unchangeable. Such models assume that all parameters are well-known and precisely defined [9]. However, a scheduling is stochastic [10] if it takes into account events, which may disturb production, as well as uncertainties in time processing and machine availability. In stochastic scheduling, resolution process is composed of two phases: *predictive phase* and *reactive one*. Thus, proposed schedules can be classified into three categories according to the importance given to each phase (*proactive schedules* (none reactive phase), *Proactive-reactive schedules* (the two phases are considered) and *dynamic schedules* (none predictive phase)).
- *Integration of other activities or not into scheduling:* we can find methods for *scheduling only*, *integrating planning into scheduling* or *integrating maintenance into scheduling*. In the first category, proposed methods try to find optimal or acceptable schedules with respect to only temporal and cost constraints. In the second category, as many plans may be generated for a job, planning and scheduling activities are unified to select the best solution. Finally, in order to reduce machine breakdown, in the last category, maintenance activities are considered as tasks with the same importance as the production ones. Both maintenance and production tasks are scheduled.

The last identified categories can be combined to provide more robust solutions. In our work, we are interested in proactive scheduling that inserts other activities like planning and maintenance.

Authors in [6] suppose job flexibility realization and propose an integrated planning scheduling based on multi agent systems. Agents that represent machines negotiate with a *Job Agent* to construct all possible plans and then, an *Optimization Agent* selects the optimal schedule. The *Optimization Agent* uses a genetic algorithm for solution calculation.

Integrating maintenance into scheduling is another research axe followed by searchers to find more robust schedules. Integrating maintenance planning and production scheduling took great interest [4] [5] and presented good results. Integration can be *sequential* or *total*. A *sequential integration* generates a schedule for production then maintenance by taking production tasks as constraints; or generates a schedule for maintenance then production by taking maintenance tasks as constraints. On the other hand, a *total integration* simultaneously generate schedule for both production and maintenance tasks. In [11], authors propose a cooperative method between two systems each one dedicated to one of the activities: production and maintenance. Production tasks are first scheduled by following a multi-agent negotiation model. Then, the same negotiation model is applied to schedule maintenance tasks. Finally a *Negotiation Agent* tries to find a consensus among the two responsible of the two systems.

III. INTEGRATING PLANNING AND MAINTENANCE INTO SCHEDULING: THE MASMPLAM METHOD

In MASMPLAM, floor shops are generalized, where machines may be identical or different. Three types of agent are identified: *Job Agent*, *Plan Agent* and *Machine Agent*.

- *Job Agent (JA):* represents a job. Its role is to initiate resolution process while specifying all information related to job realization (product to realize, deadlines, etc.).
- *Plan Agent (PlA):* is responsible of proposing one possible schedule.
- *Machine Agent (MA):* represents a production resource.

A. PROBLEM ILLUSTRATION

To clearly explain the proposed method, we consider an example of a job to be scheduled. The shop floor is composed of six machines that can be identical or different. "Tab.1"describes precedence constraints and machines able to execute job tasks. This example will be applied throughout the explanation of this method.

B. DESCRIPTION OF THE METHOD'S PHASES

After a job arrival, *Job Agent* initializes resolution process. Stages described below ("Fig.1") are followed by system agents.

Stage 1: generating possible plans

Job Agent (JA) uses job database to construct possible plans. Job database contains tasks that must be executed for each job to be realized. The result of this stage is an AND/OR graph ("Fig.2") composed of several branches. Each node represents a task and can be *initial* (first task in a plan), *final* (last task in a plan) or *intermediary task* (situated between initial tasks and final ones).

TABLE 1. PRECEDENCE CONSTRAINTS AND MACHINES ABLE TO EXECUTE TASKS' JOB

Task	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}
Anterior tasks	-	-	-	t_1, t_2	t_3	t_3	t_4	t_4	t_4	t_5, t_6	t_7,t_8,t	t_{10}	t_{11}	t_{12}
Able Machines to its execution	m_1, m_2, m_3	m_2, m_3	m_1, m_2	m_4, m_6	m_4, m_5, m_6	m_5, m_6	m_4, m_5	m_2, m_5	m_1, m_2, m_3	m_1, m_2, m_3	m_1,m_2,m_6	m_3, m_5	m_3,m_4,m_5,m_6	m_4, m_5, m_6

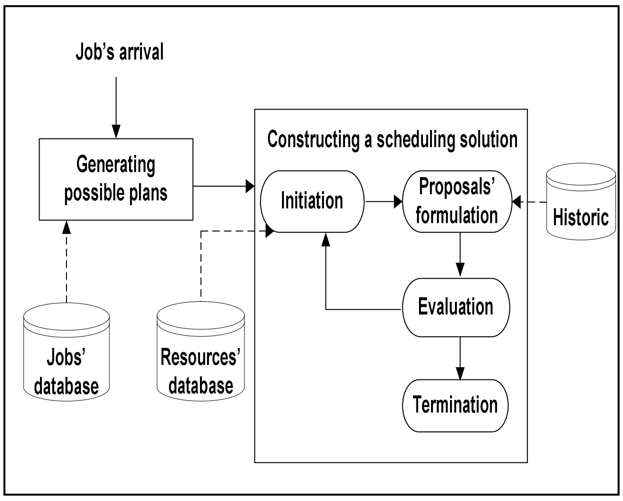

Fig. 1. MASMPLAM stages

The graph generated for the job described in section 3.1 is given on "Fig.2". A plan P is a path between an IT and a FT. In "Fig.2", Plan P_1= {t_1, t_2, t_4, t_7, t_{11}, t_{13}}.

From AND/OR graph, many relations among tasks can be identified: *Independence*, *Or-relation* and *And-relation.* Two tasks *i* and *j* are *independent* if there is no relation between them (*t_1*and*t_2*are *independent*). Independent tasks can be executed in parallel. A task *i* is *Or-related* with tasks *j* and *k* if *i* can be executed after the achievement of one of the two tasks *j* or *k* (*t_{11}* is *Or-related* with *t_7, $t_8$and t_9*). It is *And-related* with tasks *j* and *k* if it cannot be executed unless *j* and *k* are both achieved (*t_4* is *and-related* with *t_2*and *t_1*).

In this stage, *Job Agent* creates as many *Plan Agents* as Graph parts in the AND/OR graph. A Graph Part is constituted of tasks that compose one plan. In "Fig.2", four Graph parts are considered. (GP_1 = {(t_1||t_2), t_4, t_7, t_{11}, t_{13}}, GP_2 = {(t_1||t_2), t_4, t_8, t_{11}, t_{13}}, GP_3 = {(t_1||t_2), t_4, t_9, t_{11}, t_{13}} and GP_4 = {t_3, (t_5||t_6), t_{10}, t_{12}, t_{14}}). Note that "||" symbol indicates that tasks can be executed in parallel.

Stage 2: constructing a scheduling solution

Agents construct a solution by following a negotiation model. They are considered rational and egoistic because *Machine Agents* want to maintain their machines in good state as long as possible, by optimally managing their use and programming maintenance tasks. Each *Plan Agent* (PlA) negotiates with the MAs that can execute the tasks of its Graph Part. Proposed negotiation model is cyclical ("Fig.1") and is composed of four steps. Negotiations are parallel among PlAs. In each iteration, a PlA negotiates temporal window of one task or many independent tasks if they exist. A temporal window is composed of *start* and *end-date* of a task execution.

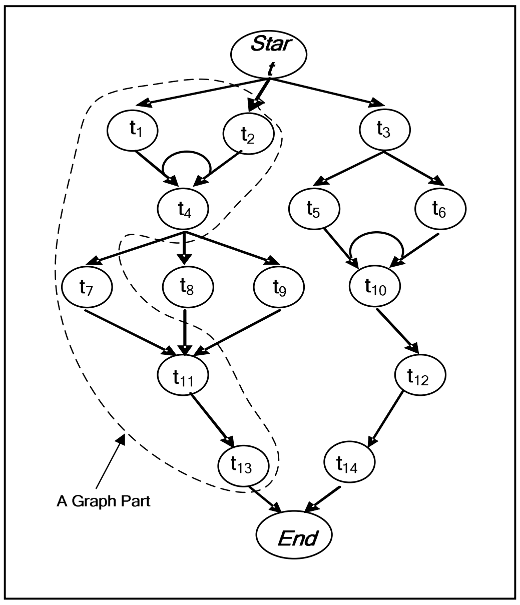

Fig. 2. AND/OR graph generated for the example's job

- .*Initiation step:* each PlA initiates a negotiation by sending a call for proposals to MAs able to execute initial independent tasks of its graph part. For this end, every PlA uses resource database that contains existing production resources and tasks that it can execute. According to "tab.1", in our job example, PlA_1 sends proposals to MA_1, MA_2 and MA_3 for executing t_1. Another proposal is sent to MA_2 and AM_3 for executing t_2. Messages include *task identifier T_{id}* and suggested *start-date SD^-*.
- *Proposal formulation:* As one machine may occur in many graph parts, an MA can receive many proposals from different PlAs. After receiving a proposal, an MA specifies maintenance activities that are programmed locally.

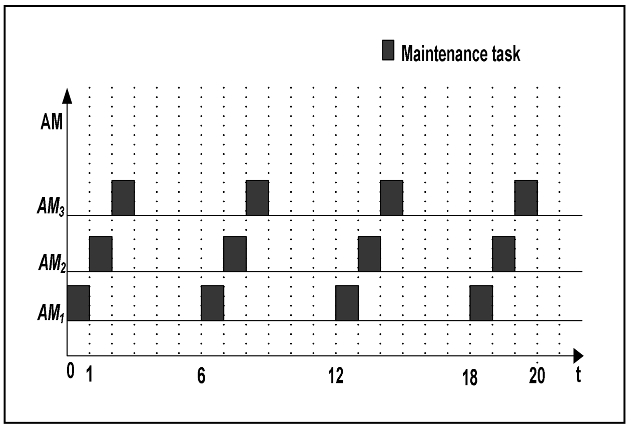

Fig. 3. Gantt chart for maintenance tasks' planning.

TABLE 2.MAS PROPOSALS AND PLA'S EVALUATION

PlA CFP (T_{id} , SD^-)	AM_1 prop. $<T_{id},D^-,d,DM>$	AM_2 prop. $<T_{id},D^-,d, DM>$	AM_3 prop. $<T_{id},D^-,d,DM>$	1st choice $<T_{id}, EEt, Dy>$	2ndchoice $<T_{id}, EEt,Dy>$
CFP (t_1, 0)	Propose $< t_1, 1, 4, 6 >$	Propose $< t_1, 0, 3, 1 >$	Propose $< t_1, 0, 5, 2 >$	M1 <1, 5, 0 >	M_2 <0, 3, 2 >
CFP (t_2, 0)		Propose $< t_2, 0, 7, 1 >$	Propose $< t_2, 0, 6, 2 >$	M_3<0, 6, 4 >	M_2 <0, 7, 6 >

Preventive or corrective maintenance can be considered. Each agent aims to maintain the good state of the machine it represents. In our method, we adopt a sequential strategy of integration of maintenance and scheduling. If we go back to our example, we suppose that the maintenance task schedules of M_1, M_2 and M_3 are shown on "Fig.3".AM_1 programmed a maintenance task that takes 1 time unit at 0t and a second one at 6t, etc. In this stage, contacted MAs formulate their proposals. A proposal is composed of a task identifier T_{id}, a start date D^-, an execution duration d and a date of the next maintenance task *DM*. Start date depends on machines availability. Machine tasks are ranked by priority into a queue. To propose execution time, each MA uses a historic module where previous experiences are saved. This module includes operation types and their execution duration, used materials and piece dimensions (ex: duration needed for cutting wood is different from the one needed for cutting aluminum. Also, cutting a piece that is 4cm depth more time than a one that is 1cm). By using this module, MAs propose more realistic dates so that negotiations may converge quickly. To formulate proposals, three cases may appear when inserting production tasks. In case 1, maintenance task is programmed before that of production. So, production task must wait until maintenance task is completed. In case 2, the two tasks are programmed at the same time. In this case, maintenance task has a higher priority than production. So it will be programmed first. In the last case, when maintenance is programmed after a task production, the MA may decide to start with production if it considers that production time does not damage the state of the machine.

- *Evaluation step:* contacted MAs send their proposals to the Plan Agent. It chooses the first and second best proposals. The evaluation criteria are *Execution End time (EEt)* and *Delay of maintenance tasks (Dy)*. Best proposal minimizes the average between execution end time and maintenance delay in order to insure machines good state. Execution end time is calculated by applying (1) whereas maintenance delay is calculated by applying (2). In case of equivalent proposals, PlA chooses the one with low maintenance delay. "Tab.2" shows AM_1, AM_2 and AM_3 proposals and final choice of PlA. For t_1 execution, PlA chooses AM_1 proposal in spite of its equivalence to AM_2 proposal (For M_1, execution end time= 5 and maintenance delay = 0 $\Rightarrow$ the average = 2.50 and for M_2, execution end time = 3 and maintenance delay = 2 $\Rightarrow$ the average = 2.50) because maintenance delay of M_1 is lower. The three proposals are schematized in a Gantt diagram on "Fig.4".

$$EEt = D^- + d \quad (1)$$

$$Dy = EEt - DM \quad (2)$$

After evaluation, Plan Agent sends a confirmation message to the selected MA. Then, it prepares the following negotiation iteration by calculating start time of the next task. Start time of the next task depends on its relation with previous tasks. If the considered task is related by an *And-relation* with its previous tasks, PlA calculates start time by applying (3). Otherwise, it will be the end execution time of the previous task.

$$\text{Start time } t_n = \text{Max}_j\,[EEt_{n-1}(M_j)] \quad (3)$$

In our job example, as t_4 is *and-related* with t_1 and t_2, start time of t_4 = Max (5 *(execution end time of t_1)*, 6 *(execution end time of t_2)*) = 6.

- *Termination step:* when every PlA terminates negotiations, it generates one schedule and calculates the resulting C_{max}. For this end, each PlA applies (4). All PAs schedules are sent to Job Agent for selecting the best one. Best schedule is the one that minimizes C_{max} .

$$C_{max} = \text{Max}_{j=1,n}\,(EEt_j) \quad (4)$$

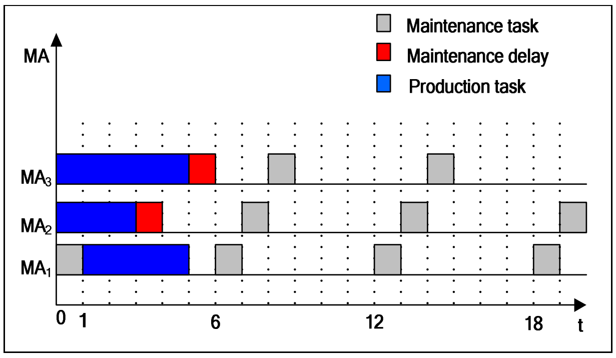

Fig. 4. Gantt diagram of agents' proposals for task t_1

IV. EXPERIMENTAL RESULTS

In order to evaluate MASMAPLAM feasibility, we have proceeded by simulation using JADE 3.0 [12]. The method has been evaluated for several numbers of agents, tasks, machines and plans. Consequently, three series of tests have been made.

- *Series 1*: in this series of tests, we measure the system scalability. We vary agent numbers per plan and plan numbers per job. Results are shown on “Fig 5”. We observe that negotiation time increases reasonably (12 seconds for 33 agents) with rise of agent numbers. Agent number varies from 7 to 33 (11 per plan) which makes our system scalable. Indeed, when agent number per plan increases, negotiations will be slower because Plan agents have to contact all MAs able to execute plan tasks.
- *Series 2*: In this series of tests, we evaluate the impact of integrating planning into scheduling. Therefore, we vary the plan number of a given job. Resulting C_{max} and negotiation time are shown on “Fig.6”. We observe that as negotiation time increases with plan number, C_{max} decreases. Indeed, the existence of several plans increases negotiation time (series 1 result) but allows agent Job to select the schedule that offers minimal C_{max}.

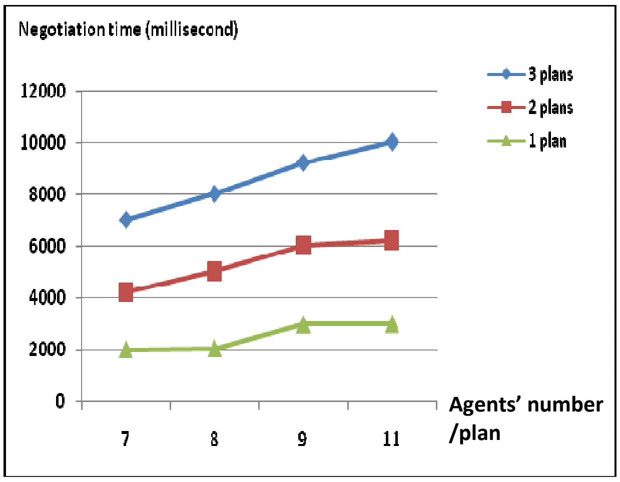

Fig. 5. Agents’ number per plan vs negotiation time

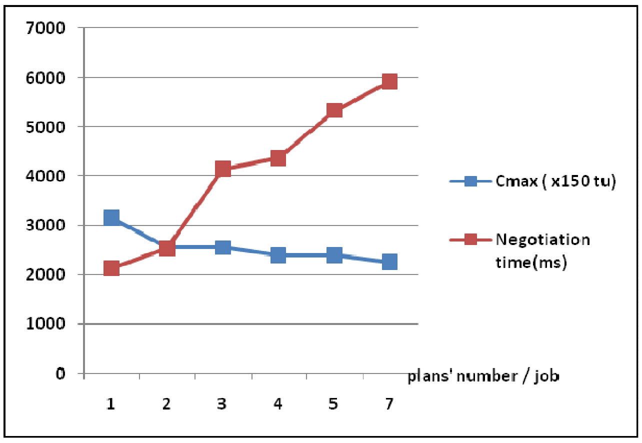

Fig. 6. Plans’ number per job vs C_{max} and negotiation time

- *Series 3*: In this series of test, we evaluate the impact of maintenance task insertion into plans. For this, we vary the machine number for a given Job and we calculate both global failure risk (5) and negotiation time. We observe on “Fig.7” that global failure risk of machines decreases when the machine number that can execute the same tasks increases. Because *Plan Agent* selects machine proposals with lower risk. Consequently, the global failure risk of the best schedule decreases.

$$FRisk = (\sum^{n}_{k} (r_j / d_k) * 100)/n \quad (5)$$

Where: n: task number in the plan, d_k: duration of production task *k* that provokes delay, r_j : maintenance delay on machine *j*.

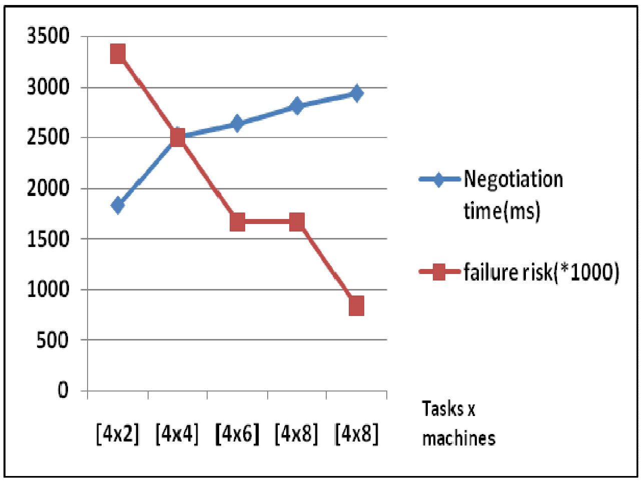

Fig. 7. Tasks x machines vs negotiation time and failure risk

IV. CONCLUSION

In this paper, we presented a new method for multi agent scheduling called MASMPLAM. It is based on integrating both planning and maintenance activities. MASMPLAM presents several advantages comparing it to existing methods. Its essential properties are its proposition of realistic and robust schedules by considering maintenance activities. The use of multi agent systems offers more reactivity to the floor shop control. Machines are represented by rational agents that aim to maintain the good state of machines. The use of historic, job and resource database allows the system to have realistic data and estimation. MASMPLAM can be also applied to *generalized job shops* or *hybrid flowshops*. Simulation results confirmed the advantages of the method by insuring minimal C_{max} and risk failure. In future work, we will improve the method by proposing a predictive-reactive solution that integrates both planning and maintenance activities.

REFERENCES

[1] J. Carlier and P. Chretienne, “Problèmes d’ordonnancement: Modélisation / complexité /Algorithme », Editions Masson, 1988.

[2] M. Owliya, M. Saadat, M. Goharian and R. Anane, “Agents-based Interaction Protocols and Topologies in Manufacturing Task Allocation”, Proceeding of the 5th International Conference on System of Systems Engineering, 2010.

[3] K. Kouiss, H. Pierreval and N. Mebarki, “Using multi-agent architecture in FMS for dynamic scheduling”, Journal of Intelligent Manufacturing 8, 1997, pp. 41- 47.

[4] F. Benbouzid. “Contribution à l’étude de la performance et la robustesse des ordonnancements conjoints production/maintenance - cas du flowshop”, Thèse de doctorat, Université de Franche-Comté, 2005.

[5] Y. Harrath, “Contribution à l’ordonnancement conjoint de la production et de la maintenance : application au cas d’un job shop”, Thèse de doctorat, Université de Franche-Comté, 2007.

[6] X. Li , C. Zhang, L. Gao, W. Li and X. Shao, “An agent-based approach for integrated process planning and scheduling”, Journal of Expert Systems with Applications 37, 2010, pp. 1256 -1264.

[7] T.C.E., Cheng and M.Y., Kovalyov, “An exact algorithm for batching and scheduling two part types in a mixed shop: Algorithms and Complexity”. International Journal of Production Economics 55, 1998, 53–56.

[8] H., Paul, C., Bierwirth and H. Kopfer, “A heuristic scheduling procedure for multi-item hoist production lines”, International Journal of Production Economics 105, 2007, 54–69.

[9] R.G. Parker, “Deterministic scheduling theory”, Chapman & Hall, 1995.

[10] H.S., Min, Y., Yih, and C.O., Kim, “Development of a real-time multi-objectives scheduler for a semiconductor fabrication system”, International Journal of Production Research, 41(10), 2003, 2345-2364.

[11] T. Coudert, B. Grabot and B. Archimède, “Système multi agents et logique floue pour un ordonnancement coopératif production/maintenance”. Journal of decision systems, volume 13- No. 1, 2004, pp. 27-62.

[12] Java Agent DEvelopment Framework, http://jade.tilab.com