Using Multi-Agent System for Dynamic Job Shop Scheduling

Min-Jung Yoo*, Jean-Pierre Müller**

*Hautes écoles commerciales (HEC) - Université de Lausanne
CH-1015 Lausanne - Switzerland
Min-Jung.Yoo@hec.unil.ch
**CIRAD 73, rue Jean-François Breton
34398 Montpellier cedex 5 - France
Jean-Pierre.Muller@cirad.fr

Abstract.

Today's industries need more flexible scheduling systems able to produce new valid schedule in response to the modifications concerning orders, production processes and deliveries of materials. This paper introduces a multi-agent system applied to a job shop dynamic scheduling problem in which new production orders or deliveries arrive continuously and affect the already scheduled plan. We have solved the problem by: i) coupling reactive and pro-active agent behavior; and ii) implementing a stochastic method - simulated annealing - into agent's behavior. The job shop scheduling system is implemented using various types of agents whose interactions make the global state of the system move from a solution to another by continuously adapting to the changes from the environment. In this perspective, the interactions between the agents representing the client job orders, the production centers and the material stocks result in the assignment of operations and the plan for stock movements. Our experimental results show that, by modifying the classical agent-based message scheme, the integration of stochastic approach and multi-agent based technology could improve dynamic scheduling problems for a small to medium size problem space.

Keywords
Multi-agent system, dynamic scheduling, job shop, simulated annealing, reactive agent.

Introduction

Most current planning and scheduling systems in manufacturing industries rely on stationary properties of the production process but this assumption has been unrealistic, given the possibilities of machine faults, multiple delays and urgent incoming orders. After receiving the modifications concerning orders, production processes, material deliveries, the provisional plan rapidly becomes invalid. The responsible persons of production lines want to know the impacts of these changes on scheduling in order to take an immediate decision. In that context, today's industries need more flexible scheduling systems which react dynamically.

This paper introduces a multi-agent system applied to a job shop scheduling problem in a dynamic environment in which the above-mentioned changes arrive continuously and affect the existing production schedule. The problem has been solved by:

- Coupling reactive and pro-active agent behaviors to solve the dynamic issues;
- Emerging the schedule through the agent interactions;
- applying a stochastic method - *simulated annealing* - as an optimization technique.

This work has been carried out in collaboration with an industrial partner (TI-Informatique in Valais, Switzerland) which wanted to replace the scheduling module integrated in their software product [Visual Prod], and an academic institute [ICARE-IsNet] for integrating the two systems – VisualProd and the multi-agent system.

This paper is organized as follows: In section 2, we outline a general definition of job shop scheduling and some problems which must be taken into consideration in a *dynamic* environment. And then we introduce our approach. Section 3 is devoted to detail the formulation of the optimization function and the technique of simulated annealing. The description of each agent is given in section 4. In section 5, we discuss our experimental results and general conclusions are given in section 6.

Problems of Job Shop scheduling in dynamic environment

General definition

A job shop is a manufacturing environment where a set of m jobs (or tasks) J={J1, … Jm} have to be performed on a set of n centers R={R1, …, Rn}([Liu Sycara 96][Daouas 96][Laahoven et al. 92]). Each job Ji is composed of a set of partially ordered operations op(ij), i= 1, …, m, j=1, …, m(i), where i is the index of the job and j is the index of the step (or the sequential order), m(i) is the total number of elementary operations in a production process. Each operation op(ij) has a deterministic processing time l(ij) and pre-assigned materials M={M1, M2, .., Mk}. The job shop scheduling problem involves synchronization of the completion of m jobs J on n resources R in consuming k materials M, and is one of the most difficult NP complete combinatorial optimization problems.

The problem constraint of job shop scheduling includes:

1. operation temporal precedence, i.e., an operation must be started on after the previous operation in the job sequence (set of operations) should be finished.
2. center capacity constraint, i.e., centers have only a single operation processing capacity.
3. material quantity constraint, that is, corresponding materials must be supplied in stocks before the consumer operation starts.

In principle, the job order has a deadline condition. This condition is considered as a preference rather than a constraint because it is possible to have non optimal solution which satisfies the deadline condition with the other constraints. A solution of a job shop scheduling problem is the set of operations to which is assigned a *start time*(st) and an *end time*(et) which satisfies all problem constraints and satisfies as well the possible the deadline preferences.

Dynamic characteristics and using agents

One of the important dynamic characteristics of the manufacturing is its environmental changes. The system should respond to a sudden change like a new job order, a canceling of a scheduled order or materials' arrival not respecting the scheduled supply time. Another kind of dynamic issue concerns the fact that operations' real finishing time sometimes differ from the scheduled time. In the case of a job shop floor, these dynamic characteristics are not so easy to take into account because of tightly coupled problem constraints (see section 2.1). As was noted in [Sauter Parunak 99], early scheduling software products were too limited to handle such dynamics or now the software becomes too complex to handle it successfully.

By the characteristics of constructing decentralized complex systems, multi-agent based approaches are often applied to solve the planning and production scheduling problems in industries ([Muller Parunak 98][SauterParunak 99][Liu Sycara 96]. Actually there exist many approaches to tackle the dynamic issues in scheduling systems ([Daouas 96][Bussman 98]) or to solve efficient scheduling problems in job shop ([Liu Sycara 96][Laahoven et al.93]). To follow some existing technologies for our application, we have noticed certain limitations.

First of all, the existing researches were carried out with a limited size which is very often not the case of a real industrial application. (less than or equal to 30 job orders In [Daouas 96], [Laarhoven et als. 92]).

The implementation of a system was based on a simplified context. In [Daouas, 96][Liu Sycara 96], the authors tackled some dynamic issues in scheduling, or in [Laarhoven et als 92] an efficient job shop scheduling problem was addressed, but the supply chain problem was often ignored. Even though some researchers have underlined the importance of the integration of entire dynamic issues in production lines ([Sauter Parunak 99]), few applications have realized it in a real context.

The integration of legacy systems is hardly considered. It is certain that some technical constraints due to the integration may affect the application of a theoretical knowledge and the real implementation. But other experiments rarely mentioned the integration aspect.

Thus we attempted, in our approach,:

- To realize a combined agent behavior, that is, goal-directed and reactive agent behavior for solving a dynamic scheduling problem for production centers and stocks;
- To use a stochastic optimization technique like simulated annealing studied by other researchers ([Laahoven et al. 92]) but in the context of different types of agents;
- To solve the integration problem between a legacy software and the multi-agent system;
- To test the prototype in an integrated environment with a significant example size.

Dynamic system and optimization problem

In the context of a job shop scheduling problem, a problem state is represented by an allocation of operations associated to production orders in respecting the constraints related to the operations (temporal precedence, center capacity and material constraints). Dynamic scheduling means finding, starting from the initial state, a new next state (i.e., allocation of operations) which satisfies the newly given constraints. For the purpose of comparing a new state with the current state, a cost function is used which represents the value of a given state. In this section, we explain the formalization of the cost function and the algorithm of simulated annealing. How these functions are integrated within the agents' model is detailed in section 5.

Optimization problem and cost function

The cost function is defined over the problem space in which each state represents a complete schedule. The optimization criteria *to minimize* considered in our system are listed below.

- the number of lately placed roders with a weight parameter (client priority): tardy cost,
- the total tardiness with a weight parameter,
- the mean value of each command's tardiness.

Given:

- nc: the total number of job orders entered into the system,
- fj: end time of production of a job order j in schedule,
- dj: given deadline for the order j,
- Lj: lateness of the job order j (Lj = fj – dj),
- Tj: effective tardiness of the job order j (Tj = max (0, Lj)),
- Uj: indicator of lateness of the job order j (0 or 1),
- Wj: weight parameter for a job order j which depends on the client priority,

four optimization criteria have been formulated as follows:

Number of job order which are scheduled lately : $\Sigma_{j(1,nc)}\, Wj * Uj$ *(f1)*

Tardiness : $\Sigma_{j(1,nc)}\, Wj * Tj,$ *(f2)*

Mean tardiness : $1/nc * \Sigma_{j(1,nc)}\, Tj$ *(f3)*

Number of non-placed job orders: np (np <= nc in a certain state) *(f4)*

Being based on these formulas, the cost function to minimize is given as follows :

$$F = c1 * (f1) + c2 * (f2) + c3 * (f3) + c4 * (f4) \qquad (f5)$$

where c1, c2, c3, c4 are the associated coefficients. In our system, there is a global view point which supervises the cost function.

Simulated annealing

Being based on statistical physics, simulated annealing technique allows to avoid local optima traps which often occur with greedy algorithms. Let F be a cost function to minimize, simulated annealing consists in:

- allowing local cost function deterioration, hoping to improve the global cost function,
- controlling this possibility with a stochastic process which is guided by a parameter, called the temperature or tolerance. Initially, there is a high tolerance to local cost function deterioration, and progressively this tolerance decreases until the system reaches the equilibrium state ([Daouas et al. 95]).

The algorithm of simulated annealing is relatively simple to implement but the efficiency of processing time and the quality of solution are high [Laahoven et al. 92]. More formally, the probability to accept the state g once generated from the current state c (Acg) is defined by the following formula.

$$Acg = \begin{cases} 1 \;\; if \;\; F(g) \le F(c) \\ 1 \;\; if \;\; random[0,1] < e^{\frac{F(g)-F(c)}{T}} \\ 0 \;\; otherwise \end{cases}$$

where T is the temperature at the current state c, F(c) is the cost function value of current state, F(g) is that of a generated state. This formula means that if the newly generated state is better than the old one, the system accepts the new state. If it is not the case, the system accepts the new state with a certain probability. Once the system decides to accept the new state g, the system decreases the temperature to adjust it for the new state g. Having drawn on the result of [Daouas 96], the following formula was used to decrease the temperature.

$$T(n+1) = Tn - (0.07 * T(init)),$$

where T(init) is the initial temperature, T(n) is the current temperature and T(n+1) is the modified temperature.

Modeling with agents

The following list shows our model matching between the real world job shop model and the model based on agent.

- Production order from client → command agent
- Production center → center agent
- Stock for materials → stock agent
- Operation → interaction element between command agents and center agents, managed by command agents
- Stock movement → interaction element between command agents and stock agents, managed by stock agents

In this model, the assignment of operations results from the interactions between agents which represent the client job orders and production centers, and the plan for stock movements arises from the interactions between agents representing the client job orders and stocks (see figure 1). The center agents and the stock agents are created at the moment of the platform initialization and exist until the platform receives the deletion perturbation of a center or of a stock.

In subsection 5.1 and 5.2, we introduce "Operation" and "Stock movement" as interaction elements and after that we detail our agents with their proactive and reactive behavior.

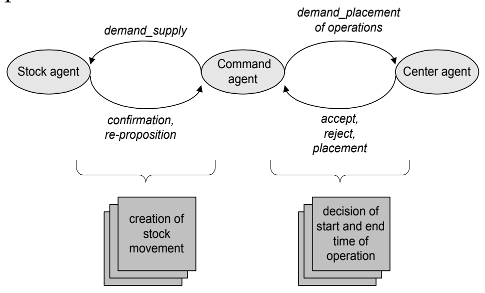

demand_supply
demand_placement of operations
Stock agent
Command agent
Center agent
confirmation, re-proposition
accept, reject, placement
creation of stock movement
decision of start and end time of operation

Fig. 1. The schedule emerges from the agent interaction.

Operation

The "Operation" element represents a elementary operation. The following information is associated with each operation.

- Center id : identification of the center on which the operation is executed,
- materials : composed of {stock identifier, quantity, duration for new supply},
- order relation between operations : previous and next operation,
- Processing time : duration of execution dependent on the quantity to produce,
- Temporal information : the earliest, latest and scheduled start time (see below).

Using processing time l(ij) and the deadline Dl , the earliest start time (es) and latest start time (ls) are calculated as:

es(ij) = i) equal to the current time unit, if op(ij) is the first operation in production,
ii) es(i, j-1) + l(i, j-1) otherwise, where l(i, j-1) is the precedent operation of op(ij).
ls(ij) = i) equal to the deadline Dl, if op(ij) is the last operation in production,
ii) ls(i, j+1) + l(i, j+1) otherwise, where l(i, j+1) is the next operation of op(ij).

The interval marked by the earliest and the latest start time mean that if an operation can be started between this time interval, the system guarantees the production deadline. And finally, the real start time is defined by the center agent, in our case, and this means that an operation is scheduled.

Stock movement

In industrial production, the supply chain problem concerns the stock management. Materials are stocked in stocks and consumed at the start time of operations. We call it stock movement. There are two kinds of stock movement: material entry into the stock (due to the delivery from suppliers) and material exit from the stock (to supply production processes). We used "Movement" element to model these aspects with the information: movement identifier, date of stock movement, quantity, stock quantity after movement.

There are also specialized information for each entry and exit type stock movement:

- Identifier of supplier for material entry: external supplier if the material must be purchased, or another production process if the material is an intermediate product.
- Identifier of consumer for material exit: the customer if it is a final product, or another production process, i.e., a command agent identifier, which manages the consumer operation.

Command Agent

A command agent represents a production order entered into the scheduling system. It is created when a new job order is entered and works for placing all of its operations on centers respecting the earliest & latest start time range. The necessary information for a command agent is: the production deadline, the priority of the client, a set of "Operations" (see section 5.1composing the production process.

- Proactive behavior: A command agent is responsible for calculating temporal information (es, ls) of its operations. Before launching the production planning, a command agent verifies operation by operation the availability of their materials, that is, if the necessary materials can be ready at the "earliest start time". Afterward, it constantly asks to place its operations on centers.
- Reactive behavior: Various perturbations (modification of the production deadline, insertion/deletion of an operation into/from the production process, receiving a reject notice from a center, or modification of materials) sent from the environment are transmitted to a command agent. These invoke the associated reactive behavior.

Center agent

The center agents represent the machines or production centers and work to allocate execution start time of all requested operations without overload. Each center agent keeps: (i) the planned operations in the center, (ii) the available free places to be occupied by operations, (iii) the current simulated annealing temperature.

The experimental results of [Daouas 96] shows that, by having a separated temperature for each center – distribution of the simulated annealing mechanism – we can speed up significantly the processing time. By the nature of job shop problem, all centers don't have equal occupancy rate for the production of the articles. So it is possible that there are some centers frequently requested for production while other centers are rarely asked for. This characteristic naturally leads us to conceive distributed simulated annealing value for each center.

- Proactive behaviour: none.
- Reactive behavior: When a center agent receives a 'demand place-operation' message with an operation from a command agent, it searches a free place within his schedule to satisfy the execution of the operation. If there is no more free place, the center agent is obliged to decide whether : i) it rejects another operation to make a place for the new one, or ii) it places the new operation regardless of the temporal values. To take a decision, the center agent uses its simulated annealing temperature to get the acceptance probability presented in section 3.2.

Stock agent

The stock agents are managing materials by maintaining the minimum stock level. The behavior of stock agents generates the stock movement plan including: purchase order, stock movement plan for operation supply:

- Proactive behavior: During the planning process, it is possible that the remaining stock quantity after movement becomes too short (below the minimum). If a stock agent encounters such a case, it generates a supplying order. If the material must be supplied by an external supplier, the stock agent creates a purchase order and then considers it will arrive at the planed time. If the material is produced by internal production, the stock agent creates an internal production order, which must be included in the schedule as a new job order.

- Reactive behavior: responding to the demand of command agents, the stock agents verify whether the stock will be sufficient to supply the necessary quantity of materials. It returns a positive confirmation if it is the case otherwise it re-activates itself for its pro-active behavior.

Experimentation

We have experimented the multi-agent system with randomly generated examples. The purpose of the experimentation was:

- To justify the application of simulated annealing method in complex job shop environment,
- To verify whether the processing time is acceptable in real world execution.

The original database that we used has been supplied by the industrial participant (TI Informatique). It contains the necessary information concerning the production processes and materials of each unit operation. We have used only one type of final product which is composed of 10 elementary operations and which needs an intermediary article composed of 6 elementary operations. The number of production centers is fixed to 6.

We have considered only the production job orders. Other dynamic aspects of the system, for example, the comparison of processing time of different types of perturbations, is excluded from this experimentation.

Parameter used in random generation

Being based on [Daouas 96], we have varied: (i) the number of production job orders, (ii) the difficulty of the generated problem.

To decide the number of production order, we have referred the statistical analysis results carried out among the users of VisualProd system. The result shows that there are 44 production orders entered into the system and 10 productions finished daily. Approximately, we have tested the system with maximum number of 100 commands. This already overpasses the number of job orders tested by other researchers.

We randomly generated the production deadline using the following formula to vary the problem complexity by controlling the deadline.

$$Dl(j) = Du(j) + Du(mean) * g(\alpha) * nd,$$

where Dl(j) is the production deadline of the job order j, Du(j) is an approximate duration of the job order j, Du(mean) is an approximate duration to produce 50 final products (see 5.2), *nd* is the number of job orders, g(α) is a randomly generated number between (α-0.1, α+0.1).

We have randomly generated the quantity of product, for each production order between, 1 and 100 (according to a uniform distribution) and measured the scheduling time in millisecond. We varied certain parameters: (i) the number of production order varies between {10, 20, 30, 40, 50, 100}, (i) α: {0.5, 1}. We have executed 5 times with the same example - that is, same number of command, same degree of difficulty, randomly generated quantity of product - to get an average value.

Results and Discussion

We have tested the first implementation version of our system with the number of production orders 10, 30, and 50, and the initial stock level set to zero. Unfortunately, the result has been away from our expectation (see table1) in measuring the processing time.

One of the benefits of applying simulated annealing method is its linear increase concerning the processing time ([Daouas 96][Laahoven et al.92]). But in our case, the processing time has increased exponentially in proportion to the number of job order. We have analyzed the reason like follows:

- The problem context becomes much more complex by integrating the stock management which was not the case in [Daouas 96][Laahoven et al.92]. By the result, the interaction between command agents and stock agents becomes complex.
- In our case, the problem size is more important that the others. Most of the experimentation has been executed over smaller problem space composed in more or less than 30 job orders. If we reduce our problem size like those, we show no more serious growth of processing time but the exponential increase is remarkable after processing of 30 job orders.

After our first trial, we have modified certain system properties, in reserving the important principals, so as to reduce the exponential growth curve. The modifications are:

- Limitation of the maximum number of rejection per operation in centers;
- Modification of agent communication scheme. We have limited the pure agent message send/ receive. Some party of agent communication were replaced by a blocked transaction using direct message passing between agent. For example, the interaction between command agents and center agents for "demand place-operation".
- For concentrating on the processing time evaluation between center agents and command agents, we have taken a sufficiently high initial stock level.

The table2 show the result after the modification:

- the growth of processing time is in linear increase, and
- the processing time of a hard problem ($\alpha = 0.5$) is superior to the less difficult one ($\alpha = 1$).

Table 1. First experimentation

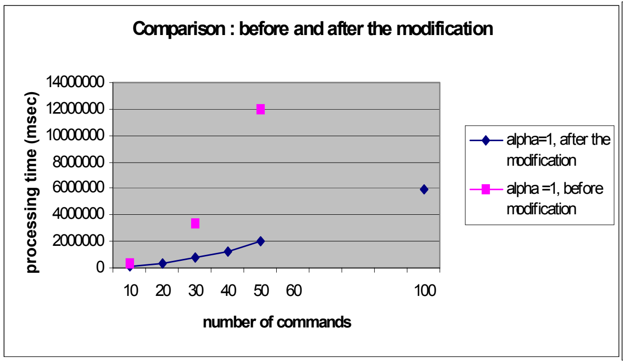
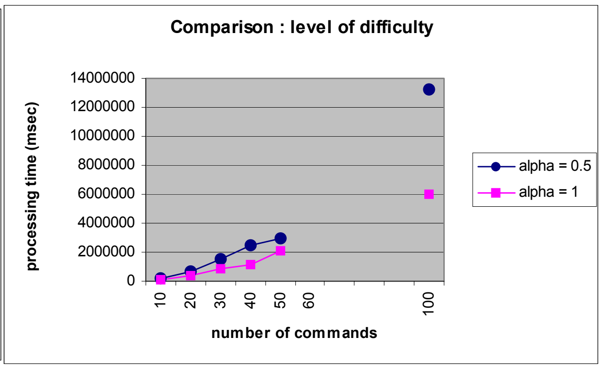


Table 2. After some modifications



Conclusions

As was mentioned in [Parunak 99], an industrial system is a mean to an end and will be accepted only if the firm can justify the project cost against expected benefits. Although the characteristics of multi-agent and distributed agent systems promise their applicability for modular, decentralized, complex and changeable enterprise systems, there are always the problems of legacy system integration and of performance.

We have succeeded to replace the existing scheduling module using agents' interaction to integrate dynamic scheduling. Now, the users can consult the impact of their input concerning new job orders, modifications of certain conditions, information for stock movements. The solution shows the new production schedule on the centers, what materials must be purchased, what production orders are retained, etc., which was exactly what the industries wanted. With our case example in the previous section, the processing time for re-scheduling after receiving a perturbation is taken about 5 –10 minutes. Meanwhile, if the system is used in a complex problem space, that is, scheduling of operations with the stock movement planing like in our system, it is possible that the processing time increases exponentially in proportion to the number of job orders and the stock state.

Now TI-Informatique continues to integrate the prototype into the other parties of VisualProd so as to get a new upgraded version. One of major duties during this integration phase is to adjust simulated annealing parameters which will pertinent with different size and characteristics of database because simulated annealing is a parameter-sensitive algorithm.

This project has been financed by the CTI-HES project no 4544.1.

References

[Bussman 98] Stefan Bussmann, An agent-oriented architecture for holonic manufacturing control, In First International Workshop on Intelligent Manufacturing Systems, 1998.
[Daouas 96] Thouraya Daouas, Ordonnancement distribué réactif pour un atelier d'assemblage de type flow shop, Thèse d'université de Neuchâtel, Suisse 1996
[Daouas et al. 95] Daouas, T., K. Ghedira, J.-P. Muller, A distributed approach for the flow shop scheduling problem. Dans: Third International Conference on Artificial Intelligence Applications, 1995.
[Ferber 99] Jacques Ferber, Multi-Agent Systems, Addison-Wesley, Readings, MA, 1999.
[ICARE-IsNet] http://www.icare.ch/
[Laarhoven et al. 92] Peter J.M.Van Laarhoven, Emile H. L. Aarts, Jan K. Lenstra, Job Shop Scheduling by Simulated Annealing, Operations Research, volume 40, pp 113-125, 1992.
[Liu Sycara 96] Jyi-Shane Liu, Katia P. Sycara, Multiagent Coordination in Tightly coupled Task Scheduling, In Proceedings of Second International Conference on Multi-Agent Systems, 1996.
[Monostori et al.98] L. Monostori, J. Hornyak, B. Kadar, Novel approaches to production planning and control, First International Workshop on Intelligent Manufacturing Systems, Switzerland, 1998.

[Muller 98] Jean-Pierre Müller, Vers une méthodologie de conception de systèmes multi-agents de résolution de problèmes par émergence, JFIADSMA'98, Hermès.
[Muller Parunak 98] Jean-Pierre Muller. H. Van Dyke Parunak, Multi-agent systems and manufacturing, 9th Symposium on Information Control in Manufacturing INCOM98, France, 1998.
[Parunak 99] H. Van Dyke Parunak, Industrial and practical applications of DAI, In [Weiss 99]
[Sauter Parunak 99] John A. Sauter, H. Van Dyke Parunak, ANTS in the Supply Chain, Workshop on Agent Based Decision Support for Managing the Internet-Enabled Supply Chain, Agents99, 1999.
[Visual Prod] http://www.ti-informatique.com/
[Weiss 99] G. Weiss (editor), Multi-agent Systems, MIT Press Cambridge MA, 1999.