ELSEVIER

Robotics and Autonomous Systems 33 (2000) 179–190

Robotics and Autonomous Systems

www.elsevier.com/locate/robot

Genetic algorithms using multi-objectives in a multi-agent system☆

Alain Cardon [a], Thierry Galinho [b], Jean-Philippe Vacher [c,*]

[a] *LIP6 Paris VI, UPMC, Case 69, 4, Place Jussieu, 75252 Paris VI, France*
[b] *LIH, Institut Supérieur d'Etudes Logistiques, Quai Frissard, BP 1137, 76063 Le Havre Cedex, France*
[c] *LIH, Institut Universitaire de Technologie, Place Robert Schuman, BP 4006, 76610 Le Havre, France*

Received 1 January 1999; received in revised form 1 May 1999; accepted 1 December 1999

Abstract

We are interested in a job-shop scheduling problem corresponding to an industrial problem. Gantt diagram's optimization can be considered as an NP-difficult problem. Determining an optimal solution is almost impossible, but trying to improve the current solution is a way of leading to a better allocation. The goal is to reduce the delay in an existing solution and to obtain better scheduling at the end of the planning.

We propose an original solution based on genetic algorithms which allows to determine a set of good heuristics for a given benchmark. From these results, we propose a dynamic model based on the contract-net protocol. This model describes a way to obtain new schedulings with agent negotiations. We implement the agent paradigm on parallel machines.

After a description of the problem and the genetic method we used, we present the benchmark calculations that have been performed on an SGI Origin 2000. The interpretation of these is a way to refine heuristics given by our evolution process and a way to constrain our agents based on the contract-net protocol. This dynamic model using agents is a way to simulate the behavior of entities that are going to collaborate to improve the Gantt diagram. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Job-shop scheduling problem; Multi-objective genetic algorithm; Multi-agent system; Contract-net protocol

1. Introduction

In the job-shop scheduling problem (JSSP), Gantt diagram's optimization can be considered as an NP-difficult problem [10]. Determining an optimal solution is almost impossible, but trying to improve the current solution is a way of leading to a better allocation. Multi-agent systems [15] are often used for such problems, where a solution exists but is not easily calculable [52–55]. We expect some solution to emerge from such situations, that is the reason why we use them. We use them to simulate the behavior of entities that are going to collaborate to accomplish actions on the Gantt diagram so as to solve a given economic function. The ideal solution to such a problem is a point where each objective function corresponds to the best (minimum) possible value.

We present our JSSP, which is a simplified version of FISIAS [41]. We present some results based on genetic algorithms [50], then we present an agent model based on the contract-net protocol to improve a solution corresponding to a Gantt diagram.

☆ Expanded version of a talk presented at the Second International Symposium on Intelligent Manufacturing Systems, IMS'98, Sakarya University, Turkey, 6–7 August 1998.

* Corresponding author.
E-mail addresses: alain.cardon@lip6.fr (A. Cardon), thierry.galinho@univ-lehavre.fr (T. Galinho), jean-philippe.vacher@poleia.lip6.fr (J.-P. Vacher).

0921-8890/00/$ – see front matter © 2000 Elsevier Science B.V. All rights reserved.
PII: S0921-8890(00)00088-9

1.1. Job-shop scheduling problem

Scheduling is an essential function in production management. It is a difficult problem depending on the number of calculations required to obtain a scheduling that optimizes the chosen criterion [37]. In addition, there are many scheduling problems and various approach methods have been proposed to solve some parts of them. We are going to define our scheduling problem and describe some existing problems as well as the constraints that we considered.

Among various definitions of the scheduling problem, we can highlight a common denominator: it is the task allocation with a minimum cost and in a reasonable time. We are in the field of discontinuous production with the processing of small and average series.

A scheduling problem exists:

- when a set of tasks (jobs) is to be processed;
- when this problem can be broken up into tasks (operations);
- when the problem consists in the definition of the temporal task location and/or the manner to allocate them to the necessary resources.

Lamy [34] defines the scheduling problem of discontinuous production: "The scheduling problem of the production consists in manufacturing at the same time, with the same resources, a set of different products."

Scheduling determines what is going to be made, when, where and with what resources; given a set of tasks to accomplish, the scheduling problem consists in determining what operations have to be executed and in giving dates and resources for these operations.

1.2. Our job-shop scheduling problem

Scheduling and planning are difficult problems [34,37] with a long and varied history in the areas of operational research and artificial intelligence, and they continue to be active research areas. The scheduling problem, which is subject to precedence and resource constraints, is an NP-difficult problem [13]. It is thus impossible to obtain an optimal solution satisfying the real time constraint.

So, heuristic algorithms are usually implemented to obtain a "good" solution instead of an optimal one [34,50]. Due to the number of varieties of production processes and the increasing rate of change in operational parameters characterizing the data to be processed (capacities of the resources, demands, etc.), it is becoming more and more difficult for management boards to make decisions.

The reason that we have chosen the JSSP with M machines and N jobs is because it is the most complex and the most often considered [10]. To determine the quality of the solution, a graphical interface has been developed (Fig. 1). For our problem, the goal is the minimization of delays and advances for all jobs according to the "due dates" given by the manager

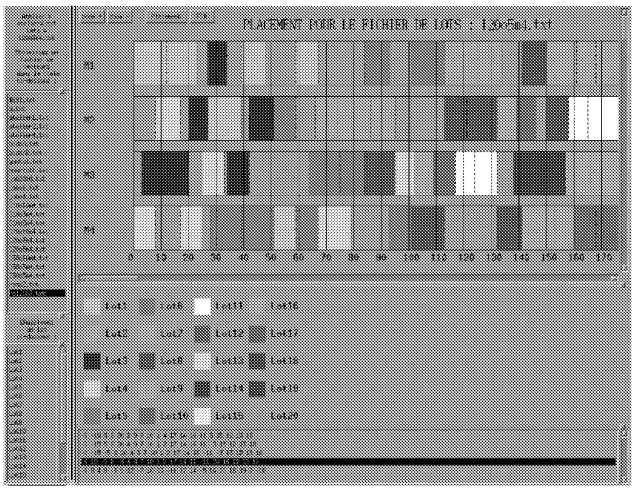

Fig. 1. Graphical interface used for our JSSP.

according to their objectives. This objective has been chosen to obtain solutions very rapidly because calculations are very numerous with genetic algorithms. Also because it is sufficient to compare these algorithms with other methods such as the gradient, the simulated annealing, etc. Genetic algorithms enable us to obtain a good quality solution quickly and easily compared to other research techniques [20,26].

2. The selection function

We cannot use the classic selection function like a method of the roulette wheel, for it is proportional. Indeed, we would take an agent as a specific entity but without taking into account some exterior pressures on the agent: its environment and the system's emergence phenomena. Therefore, the selection function should not only consider the actions of the agent, whether these actions are good or bad, but also the communications between agents. We talk about senses (semantic or link). By senses, we mean:

- the sense of communication with the other agents (network links);
- the semantics of the communication between two individuals.

Therefore, our crossover process has to take into account the semantics of communication. But an agent, seen as a structure is a compound entity made of the following elements: functions of communication, functions of action, functions of behavior along with a local genetic patrimony. Just as evolutionary algorithms simulate a Darwinian process, MAS can simulate the evolution of a nucleus or a group and by extension of an organization. A social organization may not diversify and evolve by cloning: in all social organizations (human or animal), we have a crossover process that tends to preserve the natural inheritance but also to make it more powerful. Therefore, our multi-agent system, by integrating this new concept of reproduction with crossing, will have to take into account these parameters. To achieve this, we can use a genetic algorithm switchboard, as defined by Holland [26] or an evolutionary strategy as defined by Bäck [2]. By doing that, each agent (individual) will be characterized by a chain of bits whose length will correspond to a multiple of the number of parameters. This chain will correspond to a chromosome [54] that will represent the structure of the agent. Each character (action, behavior and communication) composing the agent will correspond to numerical data referring to rules database. A_i, B_j and C_k will refer to an address database. Since knowledge is an "infinite dimension", due to the fact that an agent only has limited knowledge of its environment, the former only has, a priori, finite knowledge. By finite knowledge, we suppose that it has a finished number of actions or knowledge available. Especially, at the level of rules of action, if one takes the set of placement rules described by Pécuchet et al. [41], we have at most n rules, therefore by using assignment techniques commonly used in electronic and especially in the assignment memory, we can reserve a certain number of addresses corresponding to rules. Therefore, for a binary rule coding, we can use a coding on 10 bits; in this way, it is always possible to increase the knowledge to the level of our database. Nevertheless, the size of our chromosome is important in order to reduce the memory space. We use the coding of Gray. Thus, we can use genetic algorithms on MAS.

3. The mutation function

The mutation will correspond to the change of a bit, thus, we can use switchboard operators [17,22]. Our constraint at the mutation level, consists in having a correspondence between the bits string and the database. Thus, by changing the value of one bit, we can introduce a new character. This will have a repercussion on the environment, but especially on its membership to a group. The communications it has been able to have with other elements of the group will, incontestably, be changed. For example, consider that the mutation introduces a certain aggressiveness at the agent level, then communications with the group are going to change and the group consequently, will probably lose some of its social cohesion. Therefore in order to avoid the too abrupt upset of the social balance that can exist between individuals composing a group and the organization itself, the mutation interventions by genetic algorithms will need to be weak. Nevertheless, we can consider that at the beginning of the simulation of the organization, as at the beginning of a civilization, progress was rapid enough. Therefore, at the beginning, we can introduce an important number of mutations. We

will use as distribution for the number of mutation by generation, a curve of parameters (α, β)

$$f : x \to \frac{\beta}{x^{\alpha}} \quad \text{with } \beta \in \mathbb{R}^{+} \text{ and } \alpha \in \mathbb{R}^{+*}. \tag{1}$$

Thus, by using this type of distribution, we introduce a lot of mutations at the beginning of the simulation and few at the end in order to avoid disrupting the process of evolution by deeply modifying characteristics of chromosomes, and therefore of individuals. Too many mutations in the systems would inexorably sow the seeds of chaos. We have previously seen a possible distribution. Nevertheless, by using a Gaussian distribution to determine the probability of mutation, we keep the switchboard of the genetic algorithm [39].

4. The crossover operator

From an historical point of view, genetic algorithms [27] correspond to a random phenomenon, but the main difference compared to a classic random method is that here, we converge, step by step to an optimum (local or global) in the space of solutions [3]. Thus, we are not subject to chance as we are in the former, totally random method. A first crossing approach would be to consider an agent as a "pie chart" where each slice corresponds to a character. By randomly choosing two cut points in our agent compared to a referential, we would exchange two parts to form new individuals. However, a problem arises, as to how do we set our referential? We cannot set a permanent referential, because in this case, it supposes to consider an adjustable individual. So, an agent is an entity that has no facets. An agent is comparable to an individual part of an organization. Nevertheless, it is not possible to describe it as a physical individual (a man). Therefore this first approach is interesting but does not give satisfaction. Knowing that not all agents have the same genetic patrimony, that is to say that they have no equal chromosome lengths, and knowing that an agent has no facets, we can represent it as a toroïdal chain of bits:

- This representation does not suppose the intervention of the notion of facets of an agent.
- We can cross individuals of different lengths [20].

It is always necessary to define a starting point for our chromosome in order to correctly exchange phenotypes. Which one do we choose? In our system, an agent is composed of functions of action, knowledge and behavior, that make a certain number of possible referentials. Therefore, the choice of a referential would be a problem, except by randomly choosing it. Among possible functions, what distribution we must use? In theory, no distribution is ideal, nevertheless, to continue with this circle scheme, we will use a circle distribution or Gaussian method according to the probability. Thus, it is possible to set a referential for the crossing. However, the use of a simple crossing does not always give good results. Consequently, the use of multiple crossings allows us to make a bigger mix. We will use the uniform crossover to always have viable individuals for our representation. However, it is always possible to use the crossovers defined by Goldberg [20] such as the CX, OX and the PMX [38], that always give viable individuals.

5. The fitness function

In our case, it is necessary for us to optimize a Gantt diagram [43]. Therefore, the last operation to undertake will have to correspond to the due date minus completion time. It is necessary, therefore to minimize the delay and the advance of the set of jobs. The objective with an advance and a null delay is nearly impossible. In a general manner, we allow a certain delay or advance. When we calculate the fitness of an agent, we determine its impact on the Gantt [46]. Of course for the set of jobs, we can have a delay or a weak advance. Consequently, we no longer have a single fitness function but many. We have as many objectives as we have jobs. Consequently, we have a case of "multi-objective genetic algorithms" [44,45]. For this type of problem, we will use the basic concepts of the multi-objective optimization problem (MOP) [42].

5.1. Basic concepts and definitions

The fundamental difference between an optimization having simple or multiple objectives is in the idea of the definition of an optimal solution. The idea of optimality in the multi-objective case is a natural extension of what we have during an optimization for a unique objective.

An MOP can be defined as follows:

$$\text{MOP}: \quad \min_{x \in X} f(x), \quad \text{where } f(x) = (f_1(x), \ldots, f_n(x)) \tag{2}$$

is a vector of n real values coming from objective functions, x is a vector of n variables of decision and

$$X = \{x|\ x \in \mathbb{R}^m,\ \ g_k(x) \leqslant 0,\ \ k = 1, \ldots, m,\ \text{ and }\ x \in S\} \tag{3}$$

is a set of possible solutions. g_k is a real function value representing the kth constraint and S is a subset of $\mathbb{R}^m$ representing all the other forms of constraints. The ideal solution to such a problem is a point where each objective function corresponds to the best (minimum) possible value. The ideal solution in most cases, does not exist in view of contradictory objective functions and hence compromises have to be made. A different concept of optimality has to be introduced. Solving an MOP generally requires the identification of Pareto optimal solutions [33], a concept introduced by V. Pareto, a prominent Italian economist at the end of the last century. A solution is said to be Pareto optimal, or nondominated, if starting from that point in the design space, the value of any of the objective functions cannot be improved without deteriorating at least one of the others. All potential solutions to the MOP can thus be classified into dominated and nondominated (Pareto optimal) solutions, and the set of nondominated solutions to an MOP is called Pareto front. The first and most important step in solving an MOP is to find this set or a representative subset. Afterwards the decision maker's preference may be applied to choose the best compromise solution from the generated set. The natural ordering of vector valued quantities is basic for Pareto optimality. To define the notion of domination, let $f = (f_1, \ldots, f_n)$ and $g = (g_1, \ldots, g_m)$ be two real-valued vectors of n elements: f is partially smaller than g if: $\forall i \in 1, \ldots, n$ and $\forall k \in 1, \ldots, m,\ \ f_i \leq g_k$ and $\exists i \,|\, f_i \leq g_k$, we note $f <_{\mathrm{p}} g$. If $f <_{\mathrm{p}} g$, we say that f dominates g. Consequently, a feasible solution x^* is said to be a Pareto optimal of the problem if and only if another $x \in X$ does not exist such that $f(x) <_{\mathrm{p}} f(x^*)$.

6. Development of Pareto optimal solutions

Two different strategies are effective in generating Pareto optimal solutions [12,16]. In the first strategy, an appropriate scalar optimization problem (SOP) [42] is set-up in parametric form, so that the solution to the SOP with given values of the parameters, under certain conditions, belongs to the Pareto front; changing the parameters of the SOP leads the solution to move on the front. In the second one, the MOP is solved with a direct approach using the dominance criteria, so that a set of Pareto optimal solutions is developed simultaneously. The main advantage of the first strategy is that SOPs are generally, very well-studied problems and many efficient methods are available to solve them.

6.1. Equivalent SOP 1: The weighting approach

Following the weighting approach [16], the MOP [42] is made to correspond to the following parametrized SOP:

$$P(w)\colon \min_{x \in X} w^{\mathrm{T}} f(x) = \sum_{j=1}^{n} w_j f_j(x), \tag{4}$$

where

$$w \in W = \left\{ w | w \in \mathbb{R}^n,\ \ w_j(x) \geqslant 0,\ \ j = 1, \ldots, n \ \text{ and } \sum_{j=1}^{n} w_j = 1 \right\}, \tag{5}$$

the correspondence between the MOP and the SOP is subject to some rules. If x^0 is an optimal solution of $P(w^0)$, then it is also Pareto optimal if one of the two following conditions is verified:

- x^0 is the unique optimal solution to $P(w^0)$;
- w^0 is strictly positive.

This implies that at least some Pareto optimal solutions can be generated by solving $P(w)$ for some properly chosen w, without any hypothesis on the convexity of X and $f(X)$. Instead, some convexity hypotheses are a necessity condition. Therefore, if both X and $f(X)$ are convex, then for any given Pareto optimal solution, x^*, it is possible to find a weight vector w, not necessarily unique, such that x^* is a solution to $P(w)$. Therefore, when these convexity assumptions are verified, all Pareto optimal solutions can, in theory, be found by varying w and solving $P(w)$, while if they are not verified, some Pareto optimal solutions may never be discovered by this procedure.

6.2. Equivalent SOP 2: The constraint approach

The constraint approach [16,19] is based on the following parametrized minimum problem:

$$P_k(\epsilon) : \min_{x \in X} f_k(x), \tag{6}$$

subject to $f_j(x) \leq \epsilon_j$, $j = 1, \ldots, n$ and $j \neq k$, where $\epsilon = (\epsilon_1, \ldots, \epsilon_n)^{\mathrm{T}} \in \mathbb{R}^n$ is the vector of parameters. The main advantage of this approach is that convexity assumptions are not required. Therefore all Pareto optimal solutions can always be discovered by solving the constraint problem $P_k(\epsilon)$ for any k. The correspondence between the MOP and the SOP is subject to the following rules.

If x^0 is an optimal solution of $P_k(\epsilon^0)$ with ϵ^0 a vector for which $P_k(\epsilon^0)$ is feasible, then x^0 is a nondominated solution of the MOP, if one of the two following conditions occurs:

- x^0 is a unique solution of $P_k(\epsilon^0)$ for some given k between 1 and n.
- x^0 is not unique, but solves $P_k(\epsilon^0)$ for each and every $k = 1, \ldots, n$.

On the contrary, if x^* is a nondominated solution of the MOP, an ϵ^* can always be found such that x^* is the optimal solution of $P_k(\epsilon^*)$ for each and every $k = 1, \ldots, n$. In fact, this condition is verified when $\epsilon_i = f_i(x^*)$ for all $i = 1, \ldots, n$ with $i \neq k$.

6.3. Results from the genetic algorithm

From our object modelization, a genetic algorithm using the placing method has been developed [24,47]. This program uses the C++ language in order to use it on an SGI Origin 2000. Here are some benchmarks (see Table 1).

As the computational time depends on the computer loading, this real time does not correspond to the CPU time (Fig. 2). We can then determine heuristics [8,9,24], to use with our dynamic model based on the contract-net protocol [40,48]. Another result coming from the simulation process is a set of graphs giving the tardiness and the advance as a function of the number of generations. We can see that the delay decreases rapidly. But it is not necessary to increase the number of generations to obtain a better result (Fig. 3).

Table 1
Benchmark results

Number of jobs	Number of operations	Number of machines	Calculation time
10	10	4	50 s
50	5	4	9 min, 53 s
50	10	4	14 min, 27 s
100	10	10	40 min, 58 s
500	100	50	2 h, 24 min, 7 s

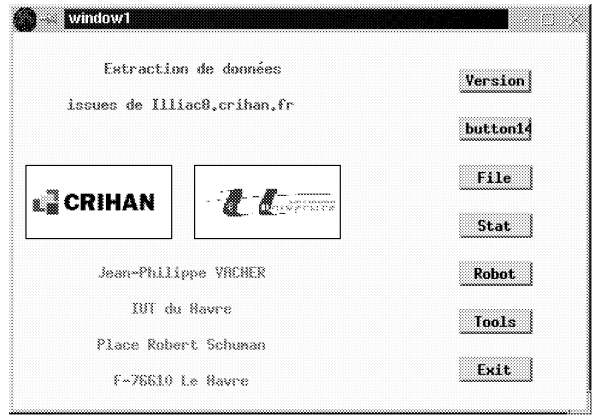

Fig. 2. Data extraction software.

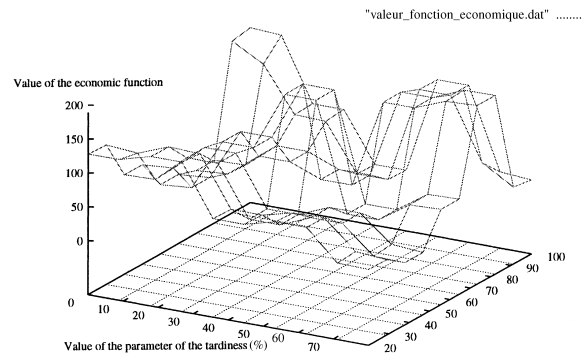

Fig. 3. Tardiness as a function of the number of generations.

6.4. Problems encountered

In our problem, the first level of modelization was a static model. It does not take into account the fact that we have interactions between machines and jobs. A job can be done only if the resource is free.

On the other hand, the static model does not include the possible interaction between the workshop and the environment such as a strike, a machine failure, etc.

In the static model, the placing obtained gives a solution corresponding to the schedule of all jobs. But, by negotiations, the schedule can be built step by step. For example, if a job arrives too soon, the delay corresponding to the tardiness increases. Therefore, if we can change the schedule during the calculation process, we can improve the tardiness. It is one of our economic functions.

7. Actions on the Gantt diagram

The JSSP does not admit a computable solution, so the use of multi-agent systems for the solving of such a problem seems reasonable [23]. Multi-agent system research is concerned with the behavior of a set of agents that cooperate in order to solve a problem [31]. In a multi-agent system (MAS) [11], the agents are seen as little problem solvers that cooperatively work in order to solve a problem [58] far beyond their individual abilities [13]. Here, we consider an agent as an entity with goals, actions to accomplish and areas of knowledge, which is situated in its environment [56]. Therefore, because of the knowledge of agents, rules of actions, etc., the MAS will have, for principal objective, to group agents having similar behavior [49] to elaborate strategies to the jobs level, jobs of jobs, machines, machines of machines, etc. Indeed, the problem of conflicts between agents is a major concern in MAS research [4,30–32]. The objective of the MAS is to improve the Gantt diagram [35,36], therefore it leads us to establish the notion of group corresponding to elementary entities having common grinds and physical sameness (same capacity of machine, etc.) or interdependence.

We will use the notion of zone for the roundup of entities on the Gantt diagram, while we will speak about the notion of group for the roundup of entities of similar or close nature. Since agents have to intervene on groups and elementary entities, the MAS will then be composed of micro- and meta-agents. It is therefore important for this evolution, to introduce agents having an evolving character: the meta-agents of evolution. These meta-agents' function will be to make this organization evolve by means of a genetic algorithm establishing a sexual reproduction of agents. It is interesting to note that, traditionally, agents only clone themselves. But here, we use a genetic algorithm for the physical evolution of the agents [28,29]. In the course of the evolution, different agent granularities appear. We have therefore micro and meta-agents that are going to intervene, according to their granularity, on an entity or a group, by passing through intermediate levels. Thus, agents having a meta-knowledge are going to be able to intervene on the macro-entities (groups) as well as on some zones of the Gantt diagram. Thus, there is a distributed agent system being able to mutate and hosting agents able to achieve a crossover based reproduction between them.

8. A contract net based approach

The contract net is a protocol for the resolution of distributed problem, defined by Smith [48]. The main objective of this protocol being the negotiation, it proceeds by allocation of tasks to a set of problem solvers and uses the concept of negotiation to grant contracts. The basic architecture contains nodes having a chief and carrying roles [7].

8.1. The contract-net procotol

Contract-net based systems represent a concept that can be used to establish mechanisms of cooperation between agents [6]. A contract net consists of a number of nodes that are represented by individual agents. By analogy to a sale session, suspended sub-tasks are openly proposed to auctions to which each node can reply according to the interest that it has for this sub-task. The stage of task attribution represents a process in which all the nodes (agents) are associated. The idea is to use the available resources and the existing knowledge of agents as efficiently as possible [51]; that is to say by allocation of one sub-task to the agent which is the most apt to operate at a given moment. The contract-net protocol forms the skeleton of our system. It is defined by a language between the nodes that can be understood by the set of agents. The communication between agents is always based on the notion of an "acceptance message". This specificity defines the exact role of an agent.

8.2. The different types of agents

The local database contains the basic knowledge of the associated nodes and also information on the progression state of the cooperative negotiation as well as the state of the resolution process [57]. The task of the communication manager is to establish communications with other agents. It is the unique component of a node that is in direct relationship with the system. Especially, it ensures the reception and sends messages.

The contract manager's job is to examine the "auctioned" task, the compliance with the contract and its ending. In other words, the contract manager ensures the coordination of all agents [14]. The task administrator is responsible for the progress made in a process and the results of a task assigned to a given agent. It receives the problem that needs to be solved from the contract manager. It uses the local database in order to find a solution and gives it back to the contract manager.

The job of a contract-net based system often begins with a problem division stage [15]. After that, the problem to be solved is divided into a set of sub-problems [25]. A special agent, the manager, assigns tasks to a sub-problem. The manager issues a public offer, called a contract for each sub-problem to be solved according to the scheme defined by Smith [48].

Because of the distributed nature of the problem, we chose an agent based modelization, a simplified version of FISIAS [41]. The different elements of the environment (the universe) can be the machines (and groups of machines), the jobs (and groups of jobs), the Gantt diagram (an attribute) and distributor agent of jobs (DAJ) (Fig. 4). Universe elements are the "objects" that can receive or send data to agents.

Machines can be seen as agents whose task is to perform the job operation at a given time. However, the machine can also be seen as a purely reactive agent, which, according to the job, can reply: "*I can do it or not*" (Fig. 5).

Then, we can use the contract net (Fig. 6) defined by Smith [48], whose manager, the DAJ, will propose the allocation of a job to a machine through the use of a negotiation agent delegated for this job (NA). According to the information given by a machine, the NA will establish a contract between a machine Mi and the DAJ [5]. But the DAJ can always propose a job to several machines to create some competition between machine agents. However, the DAJ can break this contract at any time if the agent is unable to comply with the contract.

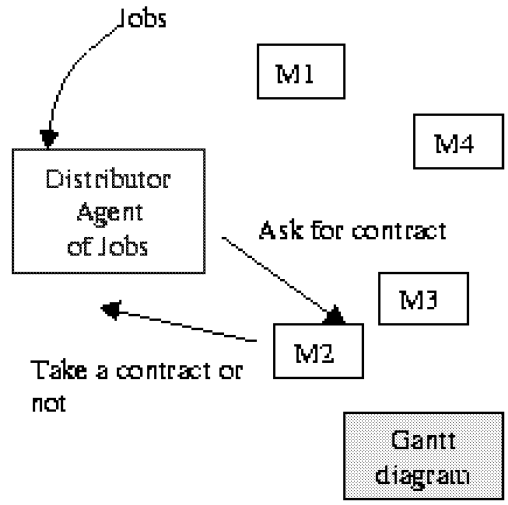

Fig. 4. Representation of the universe.

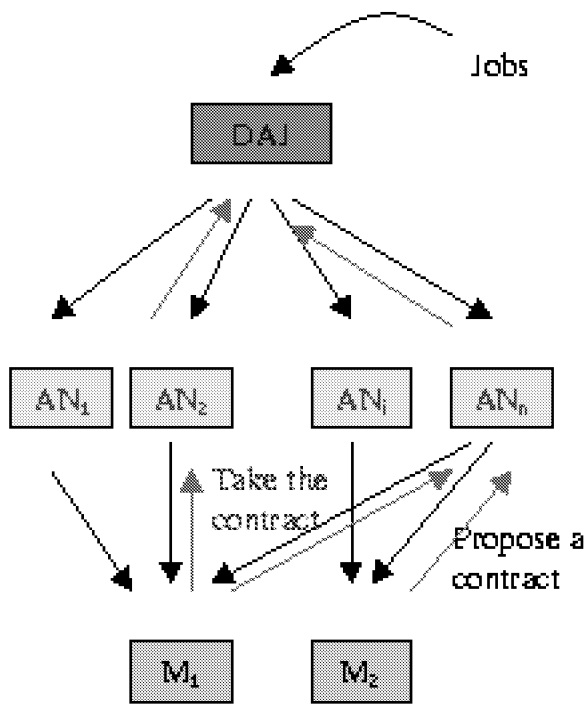

Fig. 5. First representation of the contract net.

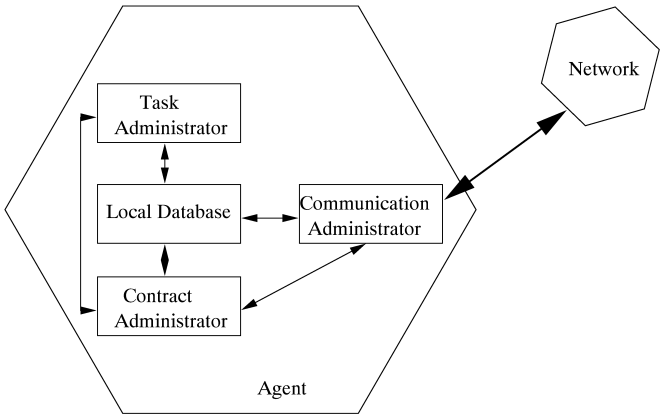

Fig. 6. Representation of the contract-net protocol in an agent.

A machine can accept many contracts corresponding to the same job, but negotiated by different NAs which have different strategies. Then, the contract ideology is "the strongest wins" since a contract can be broken by both partners [21]. However, we can propose two types of agents: a forward placing and a backward placing agent. Moreover, there are some workshop constraints since, we have generic machines, groups of machines and specialized machines. We can consider that we have "group of machines" agents which will propose the different jobs to the machines they represent. At this level, we can operate a parallel computation on the machines and so, on the Gantt diagram (Fig. 7). Nevertheless, for the moment, we consider that the DAJ can only process one job. An intermediate agent in charge of choosing the number of jobs can be proposed. Results corresponding to this approach are given in the following diagram (Fig. 8) which gives the value of the economic function (minimization of the tardiness and the advance) according to the number of agents and the number of genetic operations used by agents.

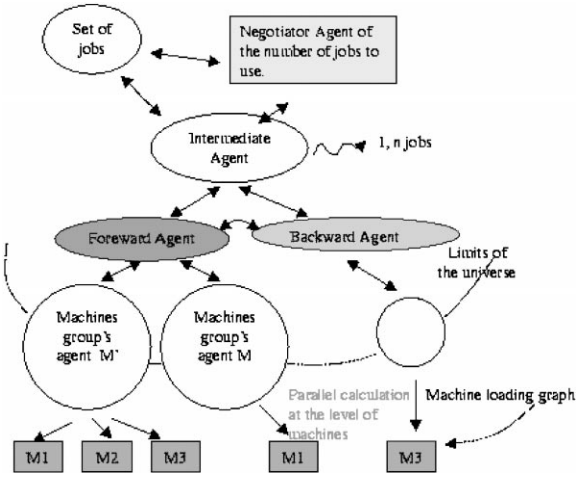

Fig. 7. Contract-net structure between the different level representations.

9. Going deeply into the relationships of GA and MAS

The use of GA in MAS is the beginning of what can be an interesting research area. There are clearly two kinds of approaches, the first is centralized, in other words, some of the genetic is outside the agent. The function of selection is a good example of such feature out-of-the-agent [18,55].

However, we believe that if one wants to completely merge the GA and MAS (Fig. 9), we must make the agent a completely autonomous genetic entity. By this, we mean that not only the genetic patrimony must be "onboard" but also the functions of selection and crossing. An agent must choose which other agent it wants to reproduce with [55]. The location of the function of mutation is not clearly known since it is

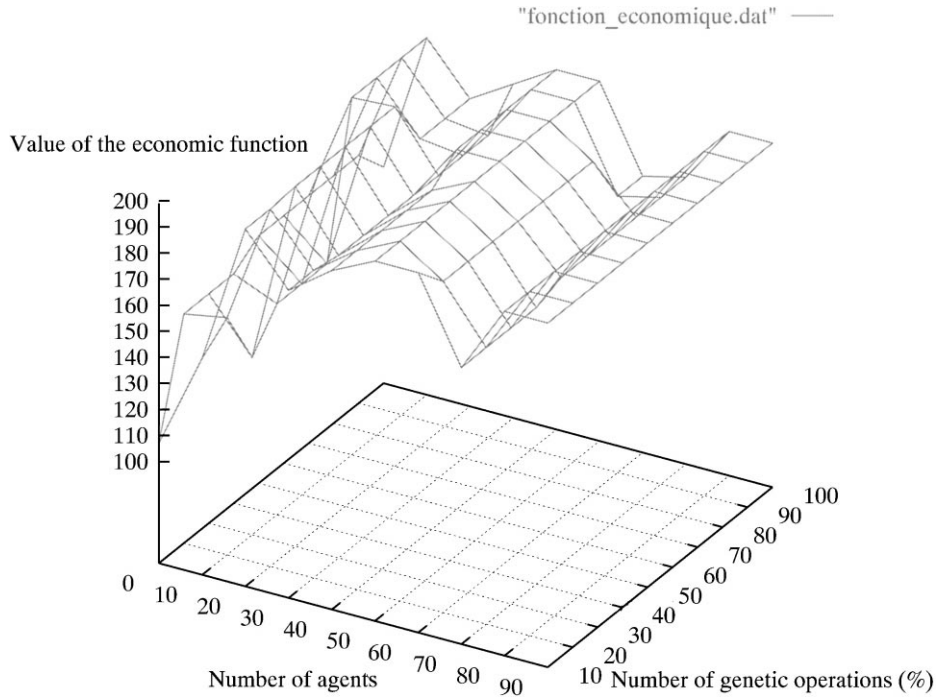

Fig. 8. Evolution of the economic function according to the number of agents and the number of genetic operations.

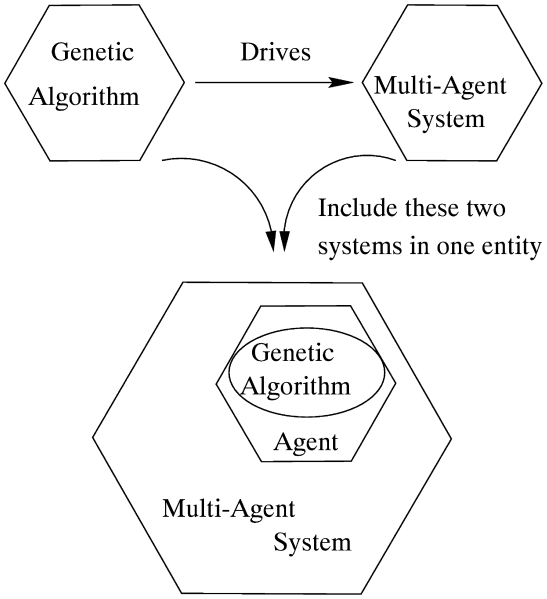

Fig. 9. MAS and GA in the environment.

caused by the possible exposure to external events originating in the environment and during the genetic code replication phase.

If we also introduce the notion of motivated behavior for agents [5], we go deeply into artificial life problematics. The genetic autonomy and the notion of motivation for an agent may lead to a drastically new kind of emergence phenomenon [1] (different social behavior, auto-referring evaluation process, etc.) in self-organizing MASs. It is certainly a difficult task but it may sow the seeds of a prolific approach concerning artificial life.

10. Conclusion

Determining an optimal solution is almost impossible, but trying to improve an existing solution is the way to improve task allocation. During the simulation process, agents granularity appears with the mutation behavior introduced by GA [38]. At the end of the simulation, communications between global and local

agents, due to their actions, lead to the appearance of agents of intermediate granularity and general optimization in production scheduling. This communication reflects the genetic integration in an MAS. The distributed vision of a scheduling problem of job-shop enables us to reach a more realistic vision because each agent has a vision of its environment. That is to say that each is capable of reacting to a particular problem just like a foreman in his workshop. He is going to react immediately following the breakdown of a machine. Similarly, he is going to take decisions, as agents, that are going to have a repercussion on the environment. Consequently, the representation by an MAS using a contract net is a close vision of reality that can be easily transferred to an industrial organization.

Research corresponding to our original dynamic approach is in progress.

References

[1] W.R. Ashby, Design for a Brain: The Origin of Adaptive Behavior, Chapman & Hall, London, 1960.

[2] T. Bäck, Self-adaptation in genetic algorithms, in: Proceedings of the First European Conference on Artificial Life, MIT Press, Cambridge, MA, 1992, pp. 263–271.

[3] R.K. Belew, L.B. Booker, in: Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, 1991.

[4] A.H. Bond, L. Gasser, Readings in Distributed Artificial Intelligence, Morgan Kaufman, San Mateo, CA, 1988.

[5] A. Cardon, F. Lesage, Toward adaptive information systems: considering concern and intentionality, in: Proceedings of the KAW'98, 1998.

[6] A. Cardon, F. Lesage, An interpretation process of communication between actors in a distributed system for crisis management, in: Proceedings of the Symposium on Informatics Economics, 1997.

[7] A. Cardon, S. Durand, A model of crisis management system including mental representations, in: AAAI Spring Symposium, Stanford University, CA, 1997.

[8] A. Cardon, J.-P. Vacher, Rapport Technique pour Ouverture de Compte au Crihan sur Machine Parallèle Illiac8, Crihan, 1998, http://www.crihan.fr.

[9] A. Cardon, J.-P. Vacher, Algorithmes Génétiques dans un Système Multi-Agents pour l'Ordonnancement, Crihan, 1998.

[10] J. Carlier, P. Chretienne, Problèmes d'Ordonnancement Modélisation/Complexité/Algorithmes, Masson, Paris, 1988.

[11] W. Chainbi, M. Jmaiel, B. Abdelmajid, Conception, behavioural semantics and formal specification on multi-agent systems, in: Chengqi Zhang, D. Lukose (Eds.), Multi-Agent Systems: Theories, Languages and Applications, Lecture Notes in Artificial Intelligence, Vol. 1544, Proceedings of the Fourth Australian Workshop on Distributed Artificial Intelligence, Springer, Berlin, 1998, pp. 16–28.

[12] V. Chankong, Y.Y. Haines, Multiobjective decision making: theory and methodology, North-Holland Series in System Science and Engineering, Vol. 8, North-Holland, Amsterdam, 1983.

[13] E.H. Durfee, V.R. Lesser, Negotiating task decomposition and allocation using partial global planning, in: L. Gasser, M.N. Huhns (Eds.), Distributed Artificial Intelligence, Research Notes in Artificial Intelligence, Pitman, London, 1989, pp. 229–243.

[14] D. McFarland, T. Bosser, Intelligent Behavior in Animals and Robots, MIT Press, Cambridge, MA, 1993.

[15] J. Ferber, Les Systèmes Multi-agents, vers une Intelligence Collective, InterEditions, Paris, 1995.

[16] C.M. Fonseca, P.J. Fleming, Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization, in: Proceedings of the Fifth International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, 1993, pp. 416–423.

[17] S. Forrest, Proceedings of the Fifth International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, 1993.

[18] T. Galinho, A. Cardon, J.-P. Vacher, Genetic integration in a multiagent system for job-shop scheduling, in: H. Coelho (Ed.), Progress in Artificial Intelligence — IBERAMIA'98, Lecture Notes in Artificial Intelligence, Vol. 484, Springer, Berlin, 1998, pp. 76–87.

[19] S. Gass, T. Saaty, The computational algorithm for the parametric objective function, Naval Research Logistics Quarterly 2 (1955) 39–65.

[20] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, MA, 1989.

[21] J.J. Grefenstette, Lamarckian Learning in Multi-agent Environments, in: Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, 1991, pp. 303–310.

[22] J.J. Grefenstette, International Conference on Genetic Algorithms and Their Applications, Lawrence Erlbaum Associates, Hillsdale, NJ, 1985.

[23] B. Hayes-Roth, A. Collinot, A satisfying cycle for real-time reasoning in intelligent agents, in: Expert Systems with Applications, Hermes, Paris, 1993, pp. 31–42.

[24] T. Haynes, Online adaptation of search via knowledge reuse, in: J.R. Koza, K. Deb, M. Dorigo, D.B. Fogel, M. Garzon, H. Iba, R.L. Riolo (Eds.), Genetic Programming, Proceedings of the Second Annual Conference, Morgan Kaufmann, Los Altos, CA, 1997, pp. 156–161.

[25] T.D. Haynes, Collective Adaptation: The Sharing of Building Blocks, University of Tulsa, Tulsa, 1998.

[26] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, 1975.

[27] J.H. Holland, Adaptation in natural and artificial systems: An introductory analysis with applications in biology, control and artificial intelligence, Complex Adaptive Systems, MIT Press, Cambridge, MA, 1992.

[28] IBM Corporation, IBM Agent Building Environment Developer's Toolkit: User's Guide, IBM Intelligent Agent Center of Competence, 1997, http://www.networking.ibm.com/iag/iaghome.html.
[29] IBM Corporation, IBM Agent Building Environment Developer's Toolkit: Components and Adapter Reference, IBM Intelligent Agent Center of Competence, 1997.
[30] R. Jackendoff, Semantics and Cognition, MIT Press, Cambridge, MA, 1983.
[31] N.R. Jennings, K. Sycara, M. Wooldridge, A Roadmap of Agent Research and Development, Kluwer Academic Publishers, Dordrecht, 1998.
[32] P. Jorion, Principes des Systèmes Intelligents, Masson, Paris, 1989.
[33] J. Koza, Genetic Programming, MIT Press, Cambridge, MA, 1992, pp. 329–355.
[34] P. Lamy, Ordonnancement et Gestion de Production, Hermes, Paris, 1987.
[35] C.G. Langton, Artificial Life, Proceedings of an Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems, Los Alamos, NM, September, 1987, Addison-Wesley, Redwood City, CA, 1989.
[36] J. Le Moigne, La Modélisation des Systèmes Complexes, Dunod, Paris, 1990.
[37] K. Mesghouni, S. Hammadi, P. Borne, Production job-shop scheduling using genetic algorithms, in: Proceedings of the IEEE/SMC, Information, Intelligence and Systems, Vol. 2, 1996, pp. 1519–1524.
[38] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer, Berlin, 1992.
[39] H. Mühlenbein, Evolution in time and space — the parallel genetic algorithm, in: G. Rawlins (Ed.), Foundations of Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, 1990, pp. 316–338.
[40] N. Muscettola, S. Smith, A probabilistic framework for resource-constrained multi-agent planning, in: Proceedings of the IJCAI-87, Milan, Italy, 1987, pp. 1063–1066.
[41] J.-P. Pécuchet, et al., FISIAS, fast interactive system for intelligent automated scheduling, Revue Automatique et Productique Appliquées 2 (4) (1989) 23–38.
[42] C. Poloni, Hybrid GA for multi objective aerodynamic shape optimisation, in: G. Winter, J. Périaux, M. Gálan, P. Cuesta (Eds.), Genetic Algorithms in Engineering and Computer Science, Wiley, Chichester, UK, 1995, pp. 397–416.
[43] M.C. Portmann, Méthodes de décomposition spatiale et temporelle en ordonnancement de production, Doctorat d'Etat, Vol. 1, Université de Nancy, 1987.
[44] G. Rawlins, Foundations of Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, 1991.
[45] J.D. Schaffer, Multiple objective optimization with vector evaluated genetic algorithm, in: Proceedings of the International Conference on Genetic Algorithms and their Applications, Lawrence Erlbaum Associates, Hillsdale, NJ, 1985, pp. 93–100.
[46] J.R. Searle, Speech Acts, Cambridge University Press, Cambridge, 1969.
[47] G. Seront, External concepts reuse in genetic programming, in: E.V. Siegel, J.R. Koza (Eds.), Working Notes for the AAAI Symposium on Genetic Programming, AAAI Press, Menlo Park, CA, 1995, pp. 94–98.
[48] R.G. Smith, The contract net protocol: High-level communication and control in a distributed problem solver, IEEE Transactions on Computers C-2912 (1980) 1104–1113.
[49] S.P. Stafford, Caring about knowledge: The importance of the link between knowledge and values, in: Proceedings of the AAAI Spring Symposium, 1997.
[50] J. Stender, E. Hillebrand, J. Kingdon, Genetic algorithms in optimisation, simulation and modelling, in: Frontiers in Artificial Intelligence and Applications, Vol. 23, IOS Press, Amsterdam, 1994.
[51] E. Tzafestas, Vers une systémique des agents autonomes: Des cellules, des motivations et des perturbations, Thèse de Doctorat, Universite Paris VI, 1995.
[52] J.-P. Vacher, T. Galinho, Z. Mammeri, Une application des algorithmes génétiques à l'ordonnancement d'atelier, in: M. Itmi, J.-P. Pécuchet, H. Pierreval (Eds.), MOSIM'97, Actes de la Première Conférence Francophone, Modélisation et Simulation des Systèmes de Production et de Logistique, Hermes, Paris, 1997, pp. 43–50.
[53] J.-P. Vacher, A. Cardon, T. Galinho, Genetic algorithms in a multi-agent system, in: Proceedings of the EUROGEN'97 Conference, Trieste, Italy, 1997, http://eurogen.cira.it.
[54] J.-P. Vacher, A. Cardon, T. Galinho, Heuristics granularity in a multi-agents system (MAS): Application to the production management, in: Proceedings of the YOR10 Conference, 1998.
[55] J.-P. Vacher, A. Cardon, F. Lesage, T. Galinho, Genetic algorithm in a multi-agent system, in: Proceedings of the IEEE International Joint Symposium on Intelligence and Systems, IEEE Press, New York, 1998.
[56] F.J. Varela, P. Bourgine, Towards a Practice of Autonomous Systems, Proceedings of the First European Conference on Artificial Life, MIT Press, Cambridge, MA, 1992.
[57] F.J. Varela, Organism, a meshwork of selfless selves, in: Proceedings of the East–West Symposium on the Origins of Language, 1996.
[58] R. Weyrauch, Building conscious artifact, in: Consciousness, Distinction and Reflexion, Bibliopolis, Napels, 1995.