Self-Organization of Agents to solve Machine Sequencing Problems

Paulo R. Ferreira Jr. and Ana L. C. Bazzan
Instituto de Informática
Universidade Federal do Rio Grande do Sul
Caixa Postal 15064 - CEP 90501-970
Porto Alegre / RS, Brasil
{prferreiraj, bazzan}@inf.ufrgs.br

ABSTRACT

In dynamic environments, agents must be able to adapt to the changing organizational goals, available resources, their relationships to another agents, and so on. This problem is a key one in multiagent systems and relates to models of learning and adaptation, such as those observed among social insects. The present paper tackles the Machine Sequencing Problem using a swarm inspired approach. In this kind of problem, limited shop-floor resources must be allocated under technological constraints to produce a specific artifact minimizing the production time. This is a critical problem in the modern production style because of its dynamism. We focus specifically on the case when the necessary activities to produce a specific artifact change over time. Our results show that, in dynamic domains, the agents adapt to changes in the environment just as social insects do.

1. INTRODUCTION

When cooperative agents work towards a common goal they must coordinate their actions considering the others' activities. A simplistic way to solve this problem is to define who-does-what statically, that means to find the system needs and design an appropriate organization. Multiagent systems need to manage the problems dynamics such as variation in the number of agents, changes in environment, and in the system's goals.

Manufacturing scheduling is a critical problem in the modern production style: the set of artifacts to be produced changes over time; the resources can become unavailable or additional ones can appear, the necessary activities to produce a specific artifact can variate, the time consumed by each activity can change, etc. Multiagent systems (MASs) are powerful to represent and solve complex problems, as those posed by manufacturing systems. Agents working on this scenario must be able to deal with requests of service arriving at any time, changes in the available resources, unpredicted failures, etc. These assumptions make the apriori design of an organization difficult because it is not possible to predict all future situations. As MASs are used in dynamic problems, static organizational structures with rigid definitions become ineffective.

The motivation for studying how agents organize and adapt can also be found in some biological entities such as the social insects. Social insect colonies show evidences of ecological success due to their *organization* which is observed in division of labor, specialization, collective regulation, etc. [2]. The needs of the colony organization change over time. These changes are associated with the phase of colony development, time of the year, food availability, predation pressure, and climatic conditions. Despite these drastic variations in colony's conditions, social insects do have ecological success. A social insect colony operates without any explicit coordination. An individual worker cannot assess the needs of the colony, it just has a fairly simple local information, and none is in charge of coordination [6]. The key feature of this emergent behavior is the plasticity in division of labor inside the colony. Colonies respond to changing conditions by adjusting the ratios of individual workers engaged in the various tasks.

In a previous work [5] we proposed an approach to adapt organization in MAS inspired on social insects colonies organization. We used a TÆMS [4] task structure to model the necessary activities to achieve the system goal: given a task structure, our approach determines the allocation of the tasks with no need of explicit coordination, communication, control hierarchy, or global view. This is done using the model which is based on task allocation used in social insect colonies. Our approach was experimented in abstract scenarios showing a good performance. These results encouraged us to model and experiment a real world problem.

In this paper we propose a way to solve dynamic manufacturing scheduling problems using our approach for MAS organizational adaptation. We focus on the Machine Sequencing Problem (MSP) [7]. In this problem, a set of activities must be sequenced under a set of constraints to optimize a given criterion. In the MSP, n parts of a product arrive in a production line and must be processed by m machines. Each part j must pass through the m machines in a *specific sequence*. This sequence is established by technological constraints. Each machine spends a specific amount of time to process each part. The sequence in which the n parts will be processed by each machine i must be chosen to minimize the production time. We show that the MSP can be modelled with TÆMS and solved in a distributed fashion by a

swarm-like organization.

Another approach for manufacturing dynamic scheduling based on the same social insects model of task allocation was presented in [3]. This approach was applied on a dynamic shop floor routing problem where wasp-like agents represent machines capable of processing different jobs. New jobs arrive in a production line and there is a setup cost when machine change from one type of job to another. Their target is to minimize the total processing time through specialization.

However, there are two main differences between our scenario and their job routing one. First, more than one machine is necessary to process the jobs (parts of an artifact). Second, there are technological constraints which imposes a specific sequence of this machines to process each job. The approach in [3] is not able to deal with the MSP because their solution is beyond the specialization of the machines. The necessary improvements to allow their approach to deal with MSP, or more generally with different scheduling problems, are open questions. More details about these limitations are discussed in Section 4.

This paper is structured as follows. Section 2.2 presents our approach in detail. The MSP scenario and the simulations are presented in Section 3. Also in this section, we discuss the results and the performance of our approach. The related work is discussed in Section 4. Section 5 concludes with further directions for this work.

2. ADAPTATION AND TASK ALLOCATION USING A SWARM-INSPIRED APPROACH

Designing an organization of agents generally involves a functional and a structural dimension. Regarding the former, in our approach we use the TÆMS language [4] to model the tasks, their interrelationships and the necessary activities to achieve the system goal. For the structural dimension, we use our approach, based on a theoretical model of task allocation in social insect colonies.

2.1 Functional Dimension: TÆMS

In TÆMS, agents' activities are represented as a graph in terms of their task groups aiming at achieving agent's goals. The leaves of the graph are called executable methods, which have probability distribution on their characteristics like quality, cost, and duration. The quality of a task group depends on what is executed and when. For example, quality can be accrued by a quality accumulation function (QAF) like *all()*, which indicates that the quality of the task is equal to the sum of the qualities of its subtasks (regardless of order or which methods are actually invoked) and all subtasks in the structure need to be accomplished for the task to have quality. Besides the local effects of the execution of methods on the quality and duration of their supertasks, there exist non-local effects (NLEs) such as enables, facilitates, etc. NLEs which involve more than one agent are called coordination relationships. Coordination mechanisms can recognize the features of the agent's subjective view, such as redundancies and soft and hard relationships.

2.2 Proposed Structural Dimension

Theraulaz et al. [8] present a model for self-organization inspired on the plasticity of division of labor in colonies of social insects. This model describes the task distribution using the stimulus produced by tasks that need to be performed and an individual response threshold related to each task. This means, at individual level, each task has an associated stimulus (e.g. food needs to be carried to the nest, if the task is to forage). The level of the stimulus increase if a task is not performed, or is not performed by enough individuals, etc. An individual that perceives (e.g. after walking around randomly) a task stimulus higher than its associated threshold, has a higher probability to perform this task. This model also includes a simple way of reinforcement learning where individual thresholds decreases when performing some task and increase when not performing. Assuming the existence of M tasks to be performed, each task j have a s_j stimulus associated. If N different individuals can perform them, each individual i have a response threshold θ_{ij} associated to a task j. The individual i engages in the task j performance with probability:

$$T_{\theta_{ij}}(s_j) = \frac{s_j^2}{s_j^2 + \theta_{ij}^2} \quad (1)$$

where: s_j is the stimulus associated with task j, and θ_{ij} is the response threshold of individual i to task j. Each individual in the model has one response threshold to each task. Those thresholds are updated (increase or decrease) according to two different coefficients. The response threshold θ is expressed as units of intensity of stimulus. The response threshold θ_{ij} of an individual i when performing task j during time interval of duration Δt is $\theta_{ij} = \theta_{ij} - \xi\Delta t_{ij}$ where ξ is the learning coefficient and Δt is the time interval.

The response threshold θ_{ij} of the agent i when not performing method j during time interval of duration Δt is $\theta_{ij} = \theta_{ij} + \rho\Delta t_{ij}$, where ρ is the forgetting coefficient.

Each particular task in the model has one associated stimulus. The intensity of this stimulus can be associated with a pheromone concentration, a number of encounters among individuals performing the task, or any other quantitative cue sensed by individuals.

We use the swarm-based model to assign insects-like agents to perform specific methods of a TÆMS task structure. This means that each agent deals with a dynamically changing TÆMS task structure and schedule its methods according to the TÆMS semantic. Next, we discuss how the ideas of social insect organizations are used to assign agents to tasks, and their applications.

In TÆMS, a method is the element in a task structure that represents what the agent can actually do (hereafter we call the insects tasks as methods). All methods in the TÆMS have probability distributions over quality (q_j), cost (c_j) and duration (d_j). Here, these are used to compute the stimulus s_j of the task to be performed (or not) by the agent. The intensity of this stimulus is associated with the results of the methods execution. Each method j have one stimulus s_j:

$$s_j = \varphi * \left(\alpha * \hat{q_j} - \beta * \hat{c_j} - \gamma * \hat{d_j} + \beta + \gamma\right) + (1 - \varphi) * x_j \quad (2)$$

where $\hat{q_j}$, $\hat{c_j}$, and $\hat{d_j}$ are the normalized quality, cost, and duration of method j; x_j is the stimulus associated with the QAF related to the method j; and $\alpha, \beta, \gamma, \varphi$ are constants.

In Equation 2, the constants are employed to set different weights to the quality (α), cost (β) and duration (γ) values

(the sum of those constants should be 1). The stimulus in our approach must increase as the quality increases (directly proportional) and decrease as cost and duration (inversely proportional) decrease. To obtain this behavior we subtract cost and duration from quality. As cost and duration assume values in the same order of magnitude of quality (each variable is normalized using the sum of its respective values for all methods.), we add β and γ (the cost and duration constants) to the equation in order to balance the influence of both directly and inversely proportional variables.

Besides, we use the constant φ to set different weights to the stimulus associated with the quality, cost and duration of the methods execution ($\alpha * \hat{q_j} - \beta * \hat{c_j} - \gamma * \hat{d_j} + \beta + \gamma$) and to the stimulus related to the emergent task succession (x_j), explained next. In this paper, these constants have the following values: $\alpha = \beta = \gamma = 1/3$ (in order to give quality, cost, and duration the same weight), and $\varphi = 0.5$.

The stimulus s_j for each method j is recalculated every time one method is performed by an agent (hereafter we call this an iteration). In our approach, performing a method influences the stimulus associated with all methods of the same TÆMS task according its QAF. This influence is computed in Equation 2 by x_j.

Let us assume the existence of M methods in the TÆMS task structure perceived by a given agent (only methods that are allowed to be performed in the current interaction). When any method k of the set of M methods is performed, all related methods j have the x_j stimulus updated as follows: $x_j = x_j + \kappa$, where κ is the constant related to the type of QAF, as defined in Table 1. This influence is recursive to each method of the parents tasks in the task structure graph. A constant κ associated with the QAF is used to model the influence of interrelated methods. We adopt small values for κ ($0 < \kappa \leq 1$) because the stimulus x_j is cumulative (increasing in each iteration) and takes values only between 0 and 1 (we assume 1 as an upperbound, not increasing the stimulus more than this value).

Table 1: QAF related constants

QAF	κ
SeqMax, Max, SeqMin, Min	0
SeqSum, Sum, All	0.01
ExactlyOne	-0.01

Our approach was developed focusing on dynamic environments where the TÆMS task structure can be modified on the fly: methods can appear or disappear; the number of available agents can change; and the interrelationship among methods can also change. The latter is supported by the stimulus model presented above. However, this model does not take into account the changes in the number of agents and methods. Bonabeau et al. [2] show that emergent task succession can be achieved using fixed thresholds, but with limited applicability. In order to overcome these limitations of the stimulus model, our approach uses a modification of the specialization model discussed next.

3. EXPERIMENTAL RESULTS

In this section, we discuss the scenario where our approach was applied as well as the achieved results in the domain of MSP (see general description in Section 1).

Figure 1 shows one instance of the MSP, modelled in

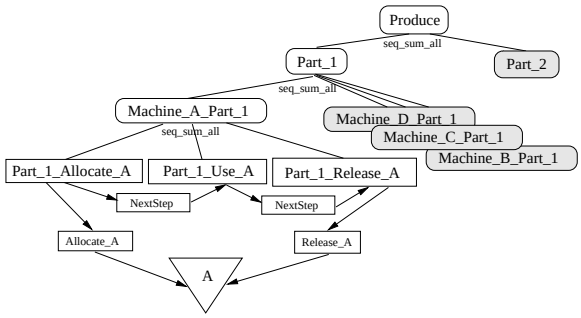

Figure 1: Objective TÆMS task structure for the MSP (subtasks for Machines B, C , D with part 1 and for all the part 2 omitted).

TÆMS. The goal here is to **produce** an artifact which is composed by two parts. and 2). In our approach, the agents play the role of a manufacturing worker. One agent gets some part and goes through each required machine processing this part. If all machines are busy when a new part arrives, the agent must stay in the queue waiting for a machine to became available. Furthermore, there is no information in the TÆMS task structure about the agents. They are all free to choose any available task to perform. Also, there is no need for information about commitments in the structure. Our approach allows each agent to decide by itself which task to perform according the constraints without any explicit commitment among agents.

In this figure we detail only the task structure for the machine *A* processing part 1 (*Machine_A_Part_1*). Additional tasks are necessary (*Machine_B_Part_1*, *Machine_C_Part_1* and *Machine_D_Part_1*) but are not shown here due to lack of space. However, they are all similar. For simplicity, tasks which are not further detailed in Figure 1 are shown using gray rectangles. Besides, each artifact must have n subtasks, one for each part. These subtasks are related by the QAF *seq_sum_all* meaning that both must be completed. Any instance of the MSP can be modelled this way, with more or less parts or machines, different technological constraints, resources, etc.

Table 2 shows the processing sequence for each part. For instance, part 1 must be processed by machine A, then D, C and finally by B. This sequence is not represented in Figure 1 to keep it clear and more simple. We model these constraints as enable interrelationships between the sub tasks of each part task (parts 1 and 2).

Table 2: Parts Sequence Constraints

Part	Machine Sequence
1	A - D - C - B
2	B - A - C - D

Each machine is a consumable resource that can only be used by a single agent at each time. When the agent needs to use a machine, it first requests it thus making it unavailable to other agents (interrelationship *Allocate_A*). When the agent finishes the task, it releases the machine (interrelationship *Release_A*). There are also enable interrelationships (*NextStep*) relating the methods **allocate** (e.g.*Part_1_Allocate_A*), **use** (e.g.*Part_1_Use_A*) and

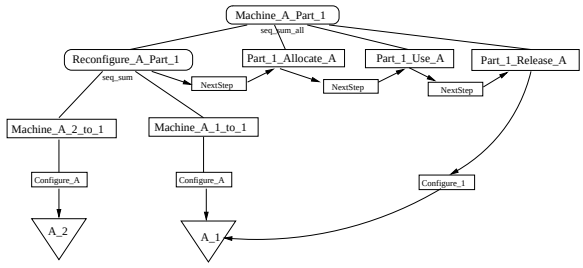

Figure 2: Task Structure for part 1 and machine A (all others omitted) with Machine Setup.

release (e.g.*Part_1_Release_A*) which determine the sequence of this process.

Table 3 shows the amount of time spent by each machine when processing the parts. It was modelled as duration in the TÆMS methods. For simplicity, cost and quality were not considered and we adopt 0 for both. However, it would be no problem to model, say, that part 1 can be processed by machine A with quality q_A or by machine B with quality q_B. The same holds for cost. Since both are handled by our approach (see Equation 2), no extension is necessary if the designer wants to consider all these aspects.

Table 3: Parts Processing Time

Part	A	B	C	D
1	6	3	1	1
2	5	5	6	2

In the MSP, a setup time could be necessary to reconfigure a machine when processing different parts in a sequence. Figure 2 shows the setup model for part 1 when processing parts using machine A. For simplicity we do not show the entire MSP model here; the model shown in Figure 2 must be combined with the one in Figure 1 to be complete. The allocate method can only be performed after the end of the configuration task. We use two consumable resources to model the configuration state of each machine. When a machine is released, the resource becomes available. There is a interrelationship between the release task and the related resource (e.g. the *Configure_1* between task *Part_1_Release_A* and resource *A_1*). One method of the reconfiguration (*Reconfigure_A_Part_1*) task is chosen (e.g. *Machine_A_2_to_1 or Machine_A_1_to_1*) by the agent according to the resource state. Only the method that has available resources will be performed. The performed method consumes the resource through the Consumes interrelationship (*Configure_A*). If the setup is necessary, it takes 10 units of time.

3.1 A Simple Scenario

In this simplified scenario there are two agents to produce one artifact (made with the two parts described before) and the machines do not need to be reconfigured (no setup time in this case). The aim of this experiment is to verify the agents abilities to perform all required tasks cooperatively.

To validate our approach, we compare our two part scheduling with the optimum selected by the Akers and Friedman algorithm [1]. Using this algorithm, in each machine we must know which part must be processed first. Each schedule indicates the sequence of parts for each machine. In a four machine scenario, there exists 2^4 possible schedules. Processing times are as in Table 3. After elaborating the 16 schedules, we eliminate the ones that are not technically possible. The remaining schedules are examined in order to remove the non-optimum plans. Finally we calculate the processing time for the four remaining schedules possibilities.

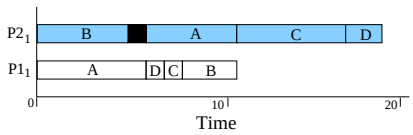

Figure 3: Best scheduling to produce 1 artifact without setup.

Figure 3 shows the best scheduling, which takes 19 time units. This is the same scheduling achieved in our system because the agents do not stay idle when any available task to be executed exists. Our agents, like the social insects, try to perform all available tasks as soon as possible. In this Figure we can see the time interval each machine was used to process each part. The white rectangles represents part 1 and the grey ones part 2. The dark rectangle represents the amount of time the part is not in any machine during its total processing time.

In the scenario discussed here, when the best solution could be manually computed, our approach was very efficient. We expect a similar behavior in other simple scenarios: our approach should achieve the best scheduling or, in the worst case, achieve a good scheduling closed to the best.

3.2 Scenario with Dynamic Changes

Here we simulate a dynamic change in the production demand. Initially, the aim is to produce 5 artifacts (five parts 1 and five parts 2) **without** setup time. Later, the task structure is changed on the fly. Five more artifacts (five parts 1 and five parts 2) must be produced **with** setup time. Here we are interested in letting our approach decide about the processing ordering in both stages without explicit coordination and/or communication.

We cannot apply the Akers and Friedman algorithm in neither stages of this scenario because the number of possible solutions grows exponentially with the number of parts.

To compare the performance of our approach with other possible solutions, we developed a greedy algorithm with two different strategies: to produce each artifact as soon as possible; or to produce all parts 1 and, once this is done, produce all parts 2.

Figure 4 shows the production time of each of the 5 artifacts (10 parts) for the first greedy strategy (a), the second greedy strategy (b), and for our approach (c). As we can see, for this case, both greedy strategies show a worse performance than our approach. Moreover, the swarm organization of the agents has achieved a better solution with no previous strategy being defined regarding this scenario. The organization of agents in terms of which task to perform has emerged.

After finishing the production of that 5 artifacts, where setup is not necessary, agents have their thresholds modified according to the methods they perform. The stimulus regarding each method also changes according to the TÆMS semantic. The agents tendency to perform each method at

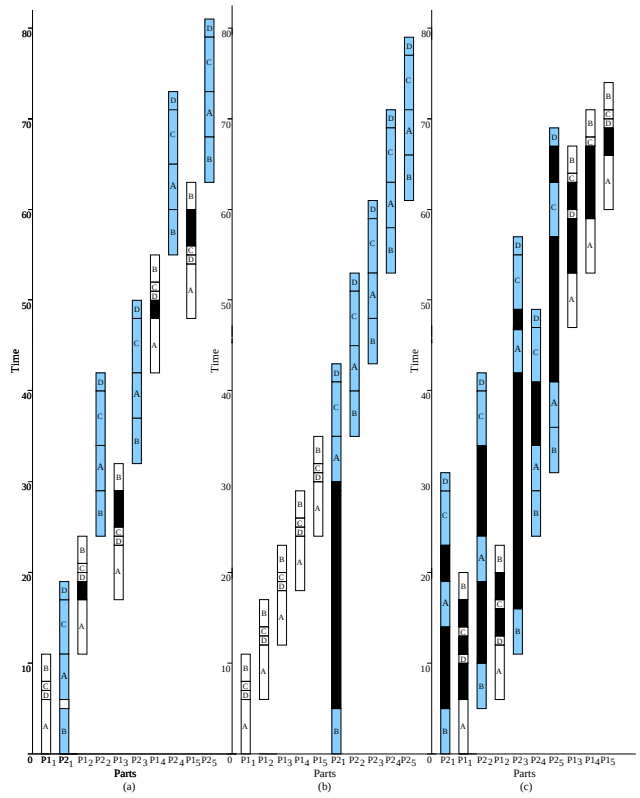

Figure 4: Production time of 5 artifacts without setup.

each time was modified during every time step according the characteristics of the problem. The decision about which task to perform in order to achieve the goal was guided by this tendency.

At this point we introduce a change in the production demand: 5 more artifacts are demanded and this time setup is necessary. Thus the agents must keep the number of setups as low as possible. One simple way to solve this problem is to accumulate demands of artifacts and produce them in batches of different parts. For example, the batch of parts 1 is produced first (again according to machine sequences and times as in Tables 2 and 3). Then the batch of parts 2 is produced. The second greedy strategy makes sense when considering the setup time (the first strategy are not reasonable because it intercalates parts 1 and 2, resulting in several unnecessary setups).

Now, the performance of the solution is related with the production of batches of parts with the same characteristics. The decision process according the tendency computed before should adapts as soon as possible to tackle this new demand. This adaptation occurs as shown below.

Figure 5 shows the production time of each part with setup time (represented by an hypothetic X machine). We can see that the our approach (a) creates the production batch, reducing the total setup time. The different types of parts overlap twice: in the first case, because the agents need to learn with the first results to adapt (part $P2_1$ with all parts $P1$). As we can see, this overlap happens at the beginning of the simulation. The second overlap occurs because there is no more parts 1 to be processed (part $P2_2$). Our approach generates 4 more setups than the greedy strategy (b). Machine B is used to process one part 2 in the scheduling begging causing more 3 setups. However, even with twice the number of setups, our approach achieve a total time closed to the greedy strategy. This shows that the agents react to the setup cost and adapt the organization to improve the scheduling process.

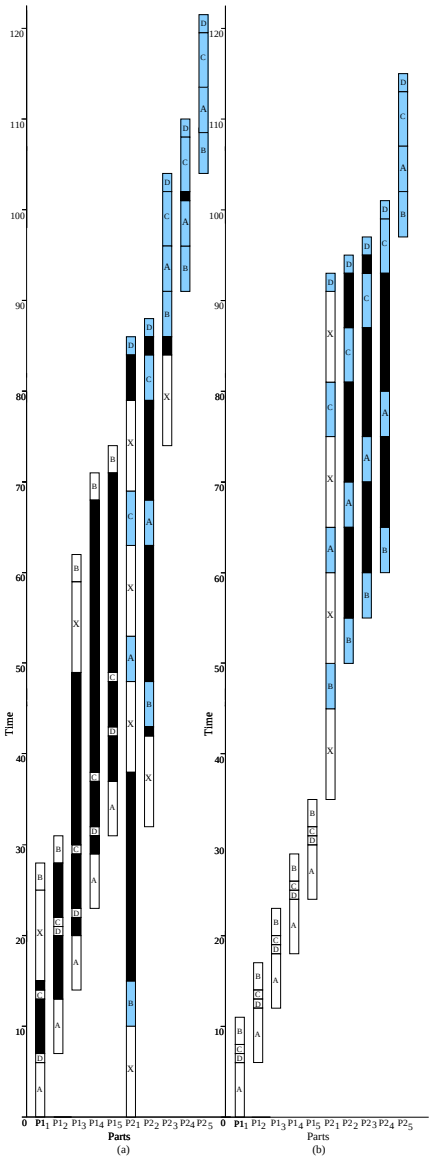

Figure 5: Production time of 5 artifacts with setup.

4. RELATED WORK

Several methods have been used trying to solve scheduling problems: heuristics, constraint propagation techniques, constraint satisfaction problem formalism, simulated annealing, taboo search, genetic algorithms, neural networks, etc. Agents technology is part of this effort, bringing a bottom-up view of the problem.

The YAMS (Yet Another Manufacturing System) was one of the earliest agent-based manufacturing systems. In

YAMS, each agent plays the role of one factory and has a collection of plans, representing its capabilities. The MAS coordination is done by the Contract Net that is used for inter-agent negotiation. Further, two classes of agent-based approaches exist: the functional decomposition and the physical decomposition. In the former, agents perform the scheduling as a local incremental search process very similar to centralized schedulers. This approach is as inflexible and inefficient for dynamic scenarios as the centralized approaches. In the physical decomposition approach, agents are used to represent entities in the physical world, such as workers, machines, or parts. The scheduling is generally realized through negotiation among agents. Several protocols for negotiation have been used as coordination mechanism, which cause high communication costs. In our approach, there is no explicit communication or coordination among the agents whose behavior is modelled as social insects.

As said, a previous approach for manufacturing dynamic scheduling based on social insects model was proposed [3]. However, this approach only tackles *routing of jobs*, with machines specializing in performing one or a few types of jobs. They use wasp like agents to represent multi-purpose machines capable of processing different jobs. There is a cost to setup the machine from one type of job to another. New jobs arrive and the machines choose whether or not to process the jobs. Their target is to minimize the setup time in order to minimize the total processing time. Initially, three cases with different probability for two types of jobs were analyzed: 50-50, 75-25 and 100-0. Their approach was capable to specialize the machines to these probabilities. Later, in a "controlled" way, the initial distribution of jobs changes after the first half of the simulation. The results shown that the adaptation requires a significant amount of time. Despite this limitation and the simple dynamic aspects of this problem, their results shown that the social insect model was competitive, or in some cases superior to previously successful agent based systems.

However, there are open questions in their work about the specificity of the proposed solutions. When applied to MSP, the machines specialization used to reduce the setup time are not enough to route the parts through the machine sequence when *more than one machine is required to process the part*. In this case, after being processed, the part should return to the production line to be processed by the next machine in the sequence and at this point it is not possible to know what machine will select the part next.

Our approach is more general: the stimulus changes with all characteristics of each task (quality, cost, and duration), as well as with the interrelationships among them. In [3], this kind of adaptation is left as a future work.

5. CONCLUSIONS

The approach presented here deals with the machine sequencing problem, and with situations when the environment changes and demands different organizations of tasks and agents. In other approaches, this adaptation requires an explicit learning component, and/or explicit communication and coordination.

We focus on a paradigm based on colonies of social insects, where plenty of evidences of ecological success exists, despite the apparent lack of explicit coordination. These insects adapt to the changes in the environment and to the needs of the colony using the mechanisms explained here. The key issues are the learning/forgetting specialization and the plasticity in division of labor. Our aim is to show that such an approach can be used to organize agents in MAS, when the systems' needs change dynamically.

In the scenarios we discussed here, the agents adapt in order to complete all tasks in the minimal time. As seen in the Dynamic Scenario, when there was setup time, agents produce parts of equal characteristics (which do not require machine setup) in order to minimize the global manufacturing time.

Besides, our results confirm our intuition that MAS can be self-organized just as social insects are, achieving the same success it does when working with cooperative distributed problem solving.

In the future we intend to analyze the characteristics of the emergent organization, mainly regarding agents specialization. Besides, we intend to use temporal polyethism and accommodate a wider range of types of agents. For instance, we might need agents with shorter life spans than others (this would imply different life probability functions), or different thresholds for the tasks to respond faster, slower or do not respond the stimulus.

6. REFERENCES

[1] S. Akers and J. Friedman. A non-numerical approach to production scheduling problems. *Operations Research*, 3:429–442, 1955.

[2] E. Bonabeau, G. Thraulaz, and M. Dorigo. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford Univ Press, 1999.

[3] V. Cicirello and S. Smith. Improved routing wasps for distributed factory control. *Journal of Autonomous Agents and Multi-Agent Systems*, 8(3):237–266, May 2004.

[4] K. S. Decker and V. R. Lesser. Quantitative modeling of complex computational task environments. In *Proceedings of the 12th International Workshop on Distributed Artificial Intelligence*, pages 67–82, Hidden Valley, Pennsylvania, 1993.

[5] P. R. Ferreira Jr., D. Oliveira, and A. C. Bazzan. A swarm based approach to adapt the structural dimension of agents' organizations. *Journal of Brazilian Computer Society - JBCS - Special Issue on Agents Organizations*, 11(1):63–73, July 2005.

[6] D. Gordon. The organization of work in social insect colonies. *Nature*, 380:121–124, 1996.

[7] V. Strusevich. Shop scheduling problems under precedence constraints. *Operations Research*, 69:351–377, 1955.

[8] G. Theraulaz, E. Bonabeau, and J. Deneubourg. Response threshold reinforcement and division of labour in insect societies. In *Royal Society of London Series B – Biological Sciences*, volume 265, pages 327–332, 2 1998.