# A Swarm Based Approach to Adapt the Structural Dimension of Agents' Organizations*

Paulo R. Ferreira Jr.†, Denise de Oliveira‡ and Ana L. C. Bazzan†
Instituto de Informática / PPGC – UFRGS
Caixa Postal 15064
91.501-970 - Porto Alegre RS - BRAZIL
{prferreiraj, edenise, bazzan}@inf.ufrgs.br

## Abstract

One of the well studied issues in multi-agent systems is the standard action-selection problem where a goal task can be performed in different ways, by different agents. Also the sequence of these actions can influence the goal achievement or its quality. This class of problems has been tackled under different approaches. At the high-level coordination one, the specification of the organizational issues is crucial. However, in dynamic environments, agents must be able to adapt to the changing organizational goals, available resources, their relationships to the presence of another agents, and so on. This problem is a key one in multi-agent systems and relates to models of learning and adaptation, such as those observed among social insects. The present paper tackles the process of generating, adapting, and changing multi-agent organization dynamically at system runtime, using a swarm inspired approach. This approach is used here mainly for task allocation with low need of pre-planning and specification, and no need of explicit coordination. The results of our approach and another quantitative one are compared here and it is shown that in dynamic domains, the agents adapt to changes in the organization, just as social insects do.

**Keywords:** Multiagent organization, Adaptation and learning in multiagent systems, Swarm intelligence, Self-organization

## 1 Introduction

The organizational structure of a multi-agent system (MAS) is one of the most significant aspect for its success [13]. The agents' organization depends on the system's goals, the perceived environment, and the relationships among agent's activities, as well as their interactions. One problem is to define which organization form fits those needs best.

A simple way to solve this is to define the organization statically, that means to find the system needs and design an appropriate organization. Once this is made off-line, the advantages of a well defined organization turn into disadvantages in an unstable environment. Agents working in dynamic environments must be able to deal with requests of service arriving at any time, changes in the available resources, unpredicted failures, etc. These assumptions make it difficult to design an organization that predicts all future situations. As multiagent systems are used in dynamic problems, static organizational structures with rigid definitions become ineffective.

A MAS needs to manage the problems dynamics, such as variation in the number of agents, changes in environment and in the system's goals. The question is how to derive such specific organizational structure given a particular situation. Most of the works in this area focuses on adapting some specific aspects of the organization or on structure generation. Each of these approaches shows good results in specific scenarios, but they are not general solutions to the problem.

The process of generating, adapting and changing organization dynamically at system runtime in a MAS is usually called self-organization. In these work we use *organizational adaptation* because our approach is based in organizational self-adaptation according to the dynamic changes in organizational needs.

This motivation for studying organizational adaptation in a MAS can also be found in some biological entities such as the social insects. Social insect colonies – also called swarms – show evidences of ecological success due to their *organization* which is observed in

*Project partially sponsored by the program CAPES–BAYERN
†Author partially supported by CNPq
‡Author partially supported by CAPES

division of labor, specialization, collective regulation, etc. [3]. The needs of the colony change over time. These changes are associated with the phase of colony development, time of year, food availability, predation pressure, and climatic conditions. Despite this 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 no one is in charge of central coordination, not even the queen, which has only reproductive function. From the aggregation over the individual workers, the colony behavior emerges without any type of explicit planning. The key feature of this emergent behavior is the plasticity in division of labor inside the colony [15]: Colonies respond to changing conditions by adjusting the ratios of individual workers engaged in the various tasks.

In this paper we propose an approach to adapt organization in multiagent systems inspired in the organization of social insects colonies. This means that our agents can adapt to changes in the environment with no need of commitments and explicit communication, just as social insects do. Consequently, the agents' architecture is defined after the behavior of such insects, especially in what regards the relationship inside an organization. Thus, it is not the aim to tackle open organizations here.

As a protocol for the structural part of an organization, to represent agents activities and interelationships, we use TÆMS [5] due to its domain-independence. TÆMS task structure is used to model the necessary activities to achieve the system goal. TÆMS is further discussed in Section 2.1, while the theoretical model of task allocation in social insect colonies appears in Section 2.2.

In Section 3 we present our approach in detail. The target scenarios and the simulations regarding them are presented in Section 4, together with the results and the performance of the current version of our approach. Section 5 concludes with further directions for this work.

## 2 Organization in Multiagent Systems and Social Insects

How to ensure that a community of individuals work together in a coherent manner to reach a common goal? The coordination process can help to answer this question by preventing chaotic behavior of decentralized systems.

It is necessary that agents coordinate their actions because they do not have a global view, and so the goals and knowledge are local, making it difficult to cooperate. Besides, the system should be able to deal with global constraints, which are not perceived by the individual agents' local view, and with inter-agents dependencies.

In a more general way, coordination increases the MAS performance. There are many ways to coordinate agents in a MAS, classified by Nwana et al. [14] in four categories: organizational structuring, contracting, multi-agent planning and negotiation. The authors provide an overview of each category, criticizing and comparing the characteristics.

We are interested in the coordination based on the organizational structure, where the community of agents behaves as an unit. The agents work toward the system goal due to their organization.

There are several works in the literature describing different approaches to define organization in multiagent systems. Lemaitre-Len and Exelent-Toledo [12] differentiate agent-centered from organization-centered approaches. In the former, the MAS does not have an explicit representation of its organization, each agent builds a representation of the organization according its own view of other agents behavior. In the latter, the organization is explicit defined; a model of the organization should be used to represent the aspects of the agents' interactions.

Kirn and Gasser [11] formalize the process of model organizations in multiagent systems, discussing the activities and elements related with the formal design of the organizational structure. Several models were proposed in the literature. Hübner et al. [9] propose to separate the models in two dimensions: functional and structural. In the former, the models tackle how to achieve the system goal. In the latter, the models deal with who will be engaged in each part of the solution. The authors present a model called $Moise^+$ where is possible to define both dimensions of the organization.

In our approach we use the functional model TÆMS [5] + GPGP [6] + Design-to-Criteria (DTC) [21] as the base of a MAS organization. The TÆMS language is used to model the task structure and the necessary activities to achieve the system goal. As for the structural dimension, we use a theoretical model of task allocation in social insect colonies.

The main difference of our proposal and the model by [9] is the structural dimension: Our agents are reactive, thus they do not reason about the organization. The structural dimension emerges from agent's inter-

action since self-organization is the key point of the approach.

Self-organization is the process of generating, adapting and changing a organization dynamically. Some authors prefer to call this process *organizational adaptation* [8] or *organization self-design*. Some critics to each of these definitions can be found in [16]. Several models of self-organizing multiagent systems have been presented [10, 17, 18]. However, most of them focus on adapting some specific aspects of the organization or on structure generation.

Earlier works were focused only on adaptive load balancing. Ishida et al. [10] present an approach to improve the performance of production systems. In their approach a particular agent (called problem solver) shares a collection of processor resources with other agents. The problem solving requests arrive continuously at variable rates. This approach relies on reorganization for the agents to deal with the requests. It exploits an adaptive trade-off of parallelism for time by composing and decomposing the agents and reallocating problem-solving knowledge. Shehory et al. [17] present an abstract approach following the same ideas. In this approach overloaded agents may migrate, pass tasks to others, merge, or clone. The agent cloning increases the portion of tasks performed by the MAS, increasing its performance.

So and Durfee [18] present an approach based on task allocation. In this approach a set of autonomous agents is engaged in cooperative problem solving. The agents perceive the efficiency of tasks' performance in local and global views. When a specific performance threshold is reached, a reorganization process is started. The organizational structure in this approach includes how the overall task is decomposed in subtasks, how the subtasks are allocated to available agents, how many agents are involved, etc. This approach has several limitations: the considered tasks are composed by independent subtasks, do not dealing with dependence between tasks; the agents do not interact in order to make commitments about task performance; and there is no resource sharing among agents.

Recently an approach that intends to be more general, based on the TÆMS framework plus system self-diagnosis was proposed [7]. This is a high-level coordination framework based on specification (of the organization goals, etc.), planning, and scheduling (which we call task allocation here). This approach shows good results but some questions about their general efficiency are still open: regarding communication issues, especially with a large number of agents, how efficient are the resulting organizations?

Our approach adapts organization in a MAS using the theoretical model of social insects' organization. This means that our agents can adapt to changes in the environment with no need of explicit coordination, communication, control hierarchy, or global view. Besides, our results confirm our intuition that multiagent systems can be self-organized just as social insects are, achieving the same success.

Next, we focus on details of organizational model (TÆMS framework), and on the swarm based model necessary to explain our approach.

## 2.1 TÆMS

TÆMS [5], GPGP [6], and the Design-to-Criteria (DTC) [21] have been used as a domain-independent language for description of tasks associated with agents and how they are organized. TÆMS allows the construction of a task model in which the relationship of the actions available to choice are shown, providing ways to model scenarios where tasks have deadlines and some kind of result must be reached.

Agent's 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 *sum()*, 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). This QAF, represented by *q_sum* in Figure 1, is used in TÆMS models further discussed in this paper. In that figure, the quality of $T_1$ is the best quality achieved when performing some or all of its methods. The quality of $\tau$ is the sum of all $T_n$ tasks' qualities. Other types of QAFs appear in Table 1.

Besides the QAF, there exist non-local effects (NLE) such as *enables*, *facilitates*, etc. Regarding the former, in the Figure 1 the method $m_{1_a}$ enables the method $m_{2_a}$ in the sense that the quality of $m_{2_a}$ cannot be accrued until $m_{1_a}$ is completed, i.e. the earliest start time of $m_{2_a}$ is the finish time of $m_{1_a}$. Therefore *enables* is a hard relationship, i.e. it has to be necessarily observed.

In TÆMS, besides actions and goals as shown above, it is possible to model many aspects related to the environment through the resources. A resource is an abstract concept being either consumable or non-consumable. The level of a resource is affected by the

*produces* and *consumes* interrelationships. The former indicates that, upon completation of a method, some amount of the target resource will be produced. The *consumes* interrelationship is used to indicate that a specific method uses some amount of resource while being performed.

By using these tools, it is possible to construct the task structure of a problem-solving situation. The actual structure is called an objective model of the environment, and is not observed by the agents. However, agents have each a subjective view of it, which they use to predict other agents actions'. The subjective view contains tasks and relationships the agent believes to be the complete model of its alternatives.

Non-local effects 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. GPGP performs analysis of the processes modeled in TÆMS, and decides about the commitments and appropriate courses of action for the agent given the constraints (deadline, resources, etc.).

The TÆMS task structure can be further employed by the task scheduler called Design-to-Criteria (DTC) [21]. DTC builds custom schedules for agents that meet constraints arising from task interactions or commitments made with other agents. In both cases the designer of a MAS must define the agents' interactions and commitments in an off-line fashion. If the off-line planning fails DTC is not able to deal with this issue. As we will show in the next sections, our approach is able to allocate activities to agents on the fly in order to optimize the schedule.

## 2.2 Swarm-Like Organization

Theraulaz et al. [19] present a model for self-organization inspired on the plasticity of division of labor in colonies of social insects [15]. Interactions among members of the colony and the individual perception of local needs result in a dynamic distribution of tasks.

This model describes the colony task distribution using the stimulus produced by tasks that need to be performed and an individual response threshold related to each task. Each individual insect has a response threshold to each task to be performed. That 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 increases if a task is not performed, or is not performed by enough individuals. 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. This double reinforcement process leads to the emergence of specialized individuals.

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, then each individual $i$ have a response threshold $\theta_{ij}$ associated to a task $j$. The individual $i$ engages in the task $j$ performance with a probability given by:

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

where:

$s_j$ stimulus associated with task $j$

$\theta_{ij}$ 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 $\theta$ is expressed as units of intensity of stimulus. The response threshold $\theta_{ij}$ of an individual $i$ when performing task $j$ during time interval $\Delta t$ is:

$$\theta_{ij} = \theta_{ij} - \xi \Delta t_{ij} \tag{2}$$

where $\xi$ is the learning coefficient ($\xi > 0$) and $\Delta t$ is the time interval.

The response threshold $\theta_{ij}$ of the agent $i$ when not performing method $j$ during time interval $\Delta t$ is:

$$\theta_{ij} = \theta_{ij} + \rho \Delta t_{ij} \tag{3}$$

where $\rho$ is the forgetting coefficient ($\rho > 0$).

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.

Variations in stimulus intensity can result from task performance or natural increase of task's demand. Bonabeau et al. [3] present two distinct ways to model these stimuli variation: performing a given task increases the demand for another tasks; and applying different success rates according to the task performance, changing to each specific task. The equations and results of these approaches are also presented in [3].

## 3 The Swarm Based Approach for Task Allocation

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 application in the actual simulated scenarios.

### 3.1 Stimulus

As mentioned in section 2.1, a method is the element in a TÆMS task structure that represents what the agent can actually do (hereafter we call the insects tasks as methods). All methods in the TÆMS task structure have probability distributions of quality, cost, and duration. These values describe the possible results of the method execution. Therefore, a method $j$ have quality ($q_j$), cost ($c_j$), and duration ($d_j$) and these are used to compute the stimulus $s_j$. The intensity of this stimulus is associated with the results of the methods execution. Each method $j$ have one stimulus $s_j$:

$$s_j = [\varphi*(\alpha*\hat{q_j} - \beta*\hat{c_j} - \gamma*\hat{d_j} + \beta + \gamma)] + [(1-\varphi)*x_j] \quad (4)$$

where:

$\hat{q_j}$ normalized quality of method $j$.

$\hat{c_j}$ normalized cost of method $j$.

$\hat{d_j}$ normalized duration of method $j$.

$x_j$ stimulus related to the method $j$, associated with a particular type of QAF.

$\alpha, \beta, \gamma, \varphi$ constants.

In Equation 4, constants are employed to set different weights to the quality ($\alpha$), cost ($\beta$) and duration ($\gamma$) values. The sum of those constants should be 1. The stimulus in our approach increases with the quality and decreases with cost and duration. As cost and duration assume values in the same order of magnitude of quality, we add $\beta$ and $\gamma$ (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 $\varphi$ to set different priorities to the stimulus associated with the results 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 QAF ($x_j$). 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). This stimulus updating is performed to model the emergent task succession discussed by Bonabeau et al. [3]. In colonies of social insects, performing a given task increases the demand for another related task. For instance, creating a waste pile at the entrance of the nest generates a need for cleaning.

In our approach, performing a method influences the stimulus associated with all methods of the same TÆMS task according to their QAFs. This influence is a way to implement the QAF aspect of the TÆMS/GPGP approach, as explained in Section 2.1. Equation 4 computes it using $x_j$.

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 $j$ will have the $x_j$ stimulus updated:

$$x_j = x_j + \kappa \quad (5)$$

where $\kappa$ is the constant related to each 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 $\kappa$ associated with the QAF is used to model the influence of interrelated methods. We adopt small values for $\kappa$ 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** | $\kappa$ |
|---|---|
| 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, as well as the interrelationship among methods. The latter is supported by the stimulus model presented above. However, this stimulus model does not take into ac-

count the changes in the number of agents and methods. Bonabeau et al. [3] 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 presented in Section 2.2. This modification is discussed next.

### 3.2 Polyethism

Division of labor, in which a set of workers specialize in different set of tasks, is an important and well-studied aspect of colony behavior [15]. The same specialization is useful for agents when there is a cost associated with the change of tasks. For instance, in a truck production line when a painting machine changes the brush color, there is a cost associated with clearing the old color and configuring the new one. In this case, the system performance would be better if the painting machines specialize in a specific color, avoiding setups costs.

According to theoretical models proposed by entomologists, each insect in the colony can perform all type of tasks, even where its morphology is special adapted to a specific one. This type of body specialization can also be modelled using the response threshold model presented in Section 2.2 (Equation 1). As said before, each agent has a response threshold ($\theta$) to each task. To differentiate the agents, and make the morphology specialization, it is enough to initialize sets of agents with different thresholds to sets of tasks. Besides, the flexible specializations is very important to the colonies development and survive. Because of this importante characteristic, all agents can potentially perform all tasks. We believe that it is a key point in the social insects approach, even if it is not true in some real application of multiagent systems. We intend to return to this question in the future.

The age oriented specialization are called by the biologists temporal polyethism. In *Apis mellifera* (honey bees), this is the main form of division of labor. Young workers perform tasks within the hive; older workers perform tasks outside the hive, such as foraging and colony defense.

Theraulaz et al. [19] only suggests an extension to model the temporal polyethism. His original model, without polyethism, uses two constants as learning and forgetting coefficients (Equations 2 and 3). To calculate the response thresholds with polyethism, we modify the Theraulaz et al. [19] specialization model. Our version uses two variables as coefficients of learning and forgetting based on temporal polyethism. The response threshold $\theta_{ij}$ of an individual $i$ when performing method $j$ is given by:

$$\theta_{ij} = \theta_{ij} - \frac{a_i}{A_i} * \frac{(\mathcal{A} - \Updownarrow_|)}{\mathcal{A}} \tag{6}$$

where:

$a_i$ age of the agent i.

$A_i$ maximum estimated age of the agent i.

$m_j$ age of the method j.

$\mathcal{A}$ age of the oldest available method.

The response threshold $\theta_{ij}$ of an individual $i$ when *not* performing method $j$:

$$\theta_{ij} = \theta_{ij} + \frac{a_i}{A_i} * \frac{m_j}{\mathcal{A}} \tag{7}$$

In this specialization model, all agents start with the same threshold $\theta_{ij}$ (usually an intermediate value). When a method is performed by an agent, the response threshold changes. For the agent to specialize in selecting a specific method, it is necessary that it select this method a few times. Thus, it is necessary to run the model for several rounds. In each round our approach produces a task allocation for the given task structure. Given the probabilistic nature of the model, these allocations are not necessarily the same.

In equations 6 and 7, we consider the agents' age (proportional to the task deadline) and the methods' age (proportional to the oldest method's age). The age of methods and agents is computed at each iteration. A method's age increases until the method is not finished. An agent can survive a single round or stay alive during several rounds. The agents have higher thresholds regarding old methods and lower thresholds regarding new ones. Besides, a young agent has a lower threshold than an old agent regarding the same method. The idea behind this is to specialize old agents regarding a wider range of methods, and young agents regarding specific methods as it occurs in Nature.

## 4 Experiments

The simulations presented in this section were generated using a simulation tool developed in JAVA and using the TÆMS API.

In our approach we adopt the tendency model proposed by Theraulaz et al. (Equation 1). This equation computes a probability distribution over the response

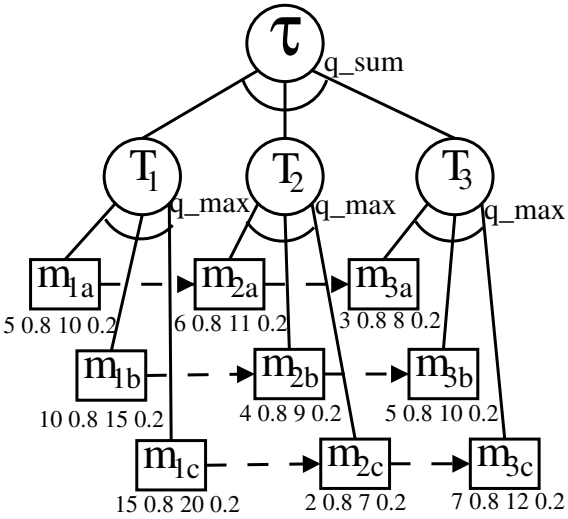


Figure 1: Objective TÆMS task structure.

of an agent regarding to each method's stimulus. As it is the case with social insects that inspire Bonabeu et al. models, with the use of this equation, each method in the task structure has a *probability* to be performed by an agent. Therefore, the best way to present and discuss the results is via the use of statistics: the results presented here are averages over 1,000 repetitions of each experiment.

In the next sub-sections we discuss the performance of our approach in three different scenarios with different aims. We start our experiments using scenario I, where the idea is to validate our approach checking if the structural dimension of an organization can emerge from the agents' interactions. In other words, we are interested only in verifying the agents abilities to perform all required tasks without explicit commitments between the agents.

Next, in scenario II we change dynamically the environment to experiment the adaptation capacity of the emergent organization. In this scenario, the dependences between tasks change; tasks disappear, and the deadline to perform the set of tasks is delayed. The agents should adapt on the fly to these.

Finally, in scenario III the aim is to check the impact of the number of agents. In dynamic situations the size of the organization can dynamically change and, in this case, the organization should be able to adapt.

### 4.1 Scenario I

We use the TÆMS task structure of Figure 1 as basis for our simulations. This task structurecan represent, for instance, a typical problem of job scheduling among multi-purpose machines [4] in which a small number of machines (2–4) are associated with wasps. This task structure can also be related to the aircraft servicing scenario discussed in [20].

Task $T_1$ is the first stage of production/servicing. Jobs or aircrafts of type a, b, or c can arrive. If it is of type a, for example, then $m_{1a}$ is allocated to an agent. This enables method $m_{2a}$ and so on. Notice that in this scenario, there are hard relationships of type *enables* which make the task allocation little flexible.

For the sake of clarity, Figure 1 shows only the duration distribution probability of each method. For instance, this distribution is *(5, 0.8; 10, 0.2)* for $m_{1_a}$ meaning that $m_{1_a}$ takes 5 time units with 80% of probability or 10 time units with 20% of probability.

All methods have equal cost and quality probability distributions: the cost for all methods is *(0, 0.8; 1, 0.2)*, which means cost 0 or cost 1 with 80% or 20% of probability respectively. The quality for all methods is *(5 0.8 4 0.2)*. Unless said, we use deadline equal to 25.

We run DTC to compare our task allocation to the standard output produced by it, which is shown in Table 2. This table presents the start and end time for each method, qualities, and costs. Not shown there: the total quality is 14.35, the total cost 0.6, and the total duration 17.0.

Table 2: DTC schedule

| method | start | finish | quality | cost | duration |
|---|---|---|---|---|---|
| $m_{1a}$ | 0.0 | 6.0 | 4.8 | 0.2 | 6.0 |
| $m_{2a}$ | 6.0 | 13.0 | 4.8 | 0.2 | 7.0 |
| $m_{3a}$ | 13.0 | 17.0 | 4.76 | 0.2 | 4.0 |

Although DTC deals with probability distributions of quality, cost, and duration coming from the semantic of TÆMS, it does not handle these distributions probabilistically (e.g. using a roulette wheel). Instead it computes an *expected average* for each probability distribution. We use both, the probabilistic and the DTC approaches. The latter is useful when we compare results; the former is used otherwise.

Using our approach in a non-probabilistic variant produces the task allocation shown in Table 3. In fact, our approach produces 1000 outputs (as mentioned before), one for each repetition. The one shown in this table is the more frequent: it is produced 32.7%

of the times. The total quality yielded is 14.4, the total cost 0.6, and the total duration 17.0. That means that our more frequent output is the one which resembles the output of DTC.

The same task structure was scheduled with probabilistic treatment of the quality, cost, and duration distributions (second variant). Table 4 shows the most frequent schedule, produced 22.3% of the 1000 repetitions. The total quality equals to 15, the total cost 0.0, and the total duration 24.0. All the three methods scheduled in the first variant are also present in the second one. However, due to the probabilistic variation of the duration, sometimes more methods can be scheduled within the deadline. In total, 41.5% of the repetitions scheduled at least the 3 methods DTC did, within the deadline.

Table 3: Most frequent schedule (first variant)

| method | start | finish | quality | cost | duration |
|---|---|---|---|---|---|
| $m_{1a}$ | 0.0 | 6.0 | 4.8 | 0.2 | 6.0 |
| $m_{2a}$ | 6.0 | 13.0 | 4.8 | 0.2 | 7.0 |
| $m_{3a}$ | 13.0 | 17.0 | 4.8 | 0.2 | 4.0 |

Table 4: Most frequent (second variant)

| method | start | finish | quality | cost | duration |
|---|---|---|---|---|---|
| $m_{1a}$ | 0.0 | 5.0 | 5.0 | 0.0 | 5.0 |
| $m_{2a}$ | 5.0 | 11.0 | 5.0 | 0.0 | 6.0 |
| $m_{3a}$ | 11.0 | 14.0 | 5.0 | 0.0 | 3.0 |
| $m_{1b}$ | 14.0 | 24.0 | 5.0 | 0.0 | 10.0 |

Of course, in both variants, the overall results of our approach are not as good as the one DTC computes. However, our approach is intended not for static environments but for dynamically evolving ones. This means that our agents can adapt to changes in the environment with no need of commitments and communication. Such environments are presented and discussed next.

## 4.2 Scenario II

In order to measure the performance of our approach in dynamic environments, we schedule four different TÆMS task structures appearing randomly with the same probability. The first task structure ($TS_1$) is depicted in Figure 1. The other three are variants of it: one has no enable relationships between the tasks ($TS_2$); one has not the task $T_3$ ($TS_3$); and the last has the deadline changed to 30 ($TS_4$). Beside, we have changed the quality distribution: in $TS_2$ it is

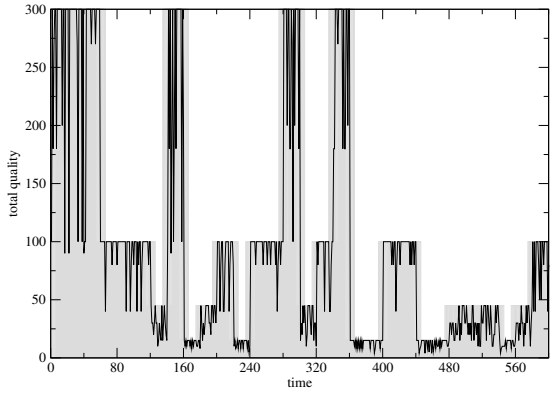


Figure 2: Change in quality over time (black line); changes in the environment are depicted as shadow boxes.

*(15 0.8 10 0.2)*, in $TS_3$ *(50 0.8 40 0.2)*, and in $TS_4$ *(100 0.8 90 0.2)*.

Of course, we can only compare *individual* task structures to each one of the four DTC outputs. In our simulation, these four outputs appear 48% of the 1000 repetitions. However, the real power of this approach is when the environment changes dynamically.

Figure 2 shows these changes and how agents adapt to them. We change task structures randomly. In the beginning (first 50 steps), agents are still adapting. Each gray shadow in the figure indicates which task structure is the actual one at a given time step. When the shadow goes up to 300, this means $TS_4$. Remember that the total quality is the sum over three tasks in $TS_1$, $TS_2$, and $TS_4$, and over two tasks in $TS_3$. When the shadow goes up to 100, 50, and 15, the actual task structure is $TS_3$, $TS_2$, or $TS_1$ respectively.

Ideally, each time a task structure changes, agents should adapt and so the quality would change instantly. Due to the time necessary for agents to adapt their stimuli and other parameters of the model, there is a small delay in this process which can be seen in Figure 2 (black lines do not match the shadows exactly).

Because of the probabilistic aspects of our approach (probabilistic distribution of methods' quality and probabilistic tendency to schedule a method) the black line is not constant, there is a variation in the schedules' quality for the same task structure. However, Figure 2 shows that when the task structure changes, so does the total quality associated with it.

This results show that modifying the task structure

on the fly disturbs only slightly the performance of the agents regarding the quality of the schedules produced because each time the task structure changes, agents do adapt to this situation.

### 4.3 Scenario III

With the same aim of the simulation described above, we now change dynamically the number of agents available to perform the task structure. In this scenario, we also employed the basis task structure of Figure 1. This time three of these task structures are subtasks of a new task group (root task) whose QAF is *sum()*. The new task group has now 3 times as many methods as the basis task structure, i.e. 27 methods. Therefore we vary the number of agents between 1 and 27. To cope with the probabilistic nature of the problem, we perform 100 repetitions each time we vary the number of agents.

Figure 3 shows the influence of the number of agents over the number of methods scheduled and also over the quality. Nine methods of those 27 do not have any *enable* interelationship. As we increase the number of agents, the number of this *enable*-free methods performed increases. When the number of agents is equal to the number of the *enable*-free methods, that means 9 agents, the number of performed methods stabilizes because even if we put more agents, they cannot perform methods which are not enabled. The same reasoning applies to the quality: the best one is achieved after this stabilization, that means when the number of agents is equal to the number of *enable*-free methods. The highest possible quality, around 40, is reached when we have 9 agents.

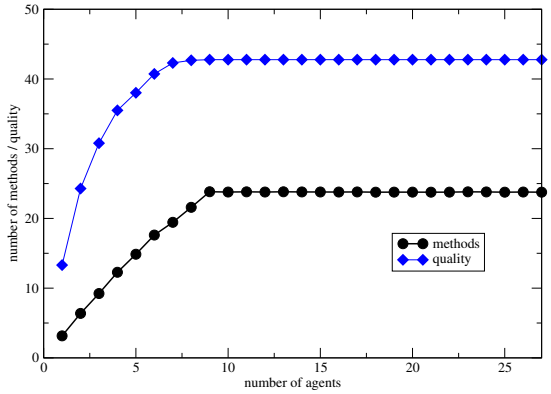


Figure 3: Number of methods and quality, for varying number of agents.

## 5 Conclusions

The approach presented here deals with the action-selection and sequencing problem. It aims at situations when the environment changes demanding different organizations of tasks and agents. In other approaches, this adaptation requires a learning component, normally based on explicit coordination and/or communication.

We focus on a paradigm based on colonies of social insects, where there are plenty of evidences of ecological success, 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 allocate tasks to agents in multiagent systems, when organizations change dynamically. As already pointed out in [2], there is no standard way of evaluating the performance of algorithms dealing with dynamic environments, because benchmark problems (e.g. travel salesman) are static problems. Therefore, we use the GPGP/DTC/TÆMS framework as comparison, although this is somehow limited to the static cases. When it comes to dynamic situations, we can only discuss the qualitative advantage of our approach. The main one is that it does not need the explicit commitments each time the number of agents changes. This is especially important when it comes to domains with large number of agents. In the scenarios we discussed here, although it takes some time for the agents to adapt, this adaptation is reached and deadlines were kept.

Also, in our approach we ensure the synchronization of team members and handle teamwork redundancy. As discussed in [1], GPGP mechanisms handle neither one nor the other. For instance, in scenario III, more than one agent can be performing the same task but no more than one agent is able to perform the same method.

In summary, there is a tradeoff between explicit coordination leading to highly accurate outputs *versus* implicit coordination via learning and adaptation leading to more relaxed outputs (less quality, higher costs or durations in the scenarios discussed here). Our approach is certainly not the best in static situations, while it is effective in dynamic ones. The efficiency

is an issue related to the specific scenarios. For instance, if, besides deadline constraints, there are also constraints related to quality or cost, then in some cases these may not necessarily be respected.

In order to tackle these limitations, we intend to work on different parameters of the functions discussed in Section 2.2 and also study new extensions to those equations so that we can 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), different thresholds to the tasks in order to respond faster or slower, or even agents unable to perform types of tasks to approximate the approach to real life, odd situations.

We also intend to test scenarios with a large number of agents. In our approach, having a large number of agents is straightforward since they all follow the same basic specialization/plasticity model. Even if we consider the extensions just discussed, having a large number of agents would not be a problem. What makes the comparison difficult now is the lack of such a result in the literature.

Also, resources are not explicitly modelled in our approach. We decided to do this because we are still looking for a suitable model (from the theoretical biology point of view) explaining whether or not insects have a different behavioral model for tasks and resources such as food. Handling resources and increasing the range of non-local relationships are necessary extensions in order to be able to compare other scenarios already used by the GPGP/DTC/TÆMS framework.

Finally, it would be desirable to have probabilistic definitions of non-local relationships (e.g. an *enables* exist between $T_x$ and $T_y$ with probability $p$). In this case, this would have to be extended in TÆMS as well.

## Acknowledgments

The authors would like to thank Dr. Franziska Klüegl for the valuable discussions regarding the swarm-inspired model. CAPES and the state of Bayern, Germany, have sponsored the project.

## References

[1] Abdallah, S., Darwish, N., and Hegazy, O. Monitoring and synchronization for teamwork in GPGP. In *Proceedings of the 2002 ACM Symposium on Applied Computing* (2002), pp. 288–293.

[2] Bonabeau, E., Sobkowski, A., Theraulaz, G., and Deneubourg, J.-L. Adaptive task allocation inspired by a model of division of labor in social insects. In *Biocomputing and Emergent Computation* (1997), D. Lundh, B. Olsson, and A. Narayanan, Eds., World Scientific, pp. 36–45.

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

[4] Cicirello, V., and Smith, S. Improved routing wasps for distributed factory control. In *Workshop on Articial Intelligence and Manufacturing: New AI Paradigms for Manufacturing* (August 2001).

[5] Decker, K. S., and Lesser, V. R. Quantitative modeling of complex computational task environments. In *The Eleventh National Conference on Artificial Intelligence* (Washington, 1993), AAAI Press.

[6] Decker, K. S., and Lesser, V. R. Designing a family of coordination algorithms. In *The First International Conference on Multi–Agent Systems* (San Francisco, CA, 1995), AAAI Press.

[7] Horling, B., Benyo, B., and Lesser, V. Using self-diagnosis to adapt organizational structures. In *Proceedings of the 5th International Conference on Autonomous Agents* (Montreal, June 2001), ACM Press, pp. 529–536.

[8] Horling, B., Lesser, V., Vincent, R., Bazzan, A., and Xuan, P. Diagnosis as an integral part of multi-agent adaptability. In *DARPA Information Survivability Conference and Exposition* (South Carolina, January 2000), IEEE Computer Society, pp. 211–219. see also UMASS CSTR 1999-03.

[9] Hübner, J. F., ao Sichman, J. S., and Boissier, O. A model for the structural, functional, and deontic specification of organizations in multiagent systems. In *Proceedings of the 16th Brazilian Symposium on Artificial Intelligence* (2002), Springer-Verlag, pp. 118–128.

[10] Ishida, T., Yokoo, M., and Gasser, L. An organization approach to adaptative production systems. In *National Conference on Artificial Intelligence* (1990), pp. 52–58.

[11] Kirn, St.; Gasser, L. Organizational approaches to coordination in multi-agent systems.

*Informationstechnik und Technische Informatik (IT+TI)*, 4 (1998), 23–29.

[12] Lemaitre-Len, C., and Excelente-Toledo, C. Multiagent organization approach. In *2nd Iberoamerican Workshop on Distributed Artificial Intelligence and Multi-Agent Systems* (1998).

[13] Lesser, V. Reflections on the nature of multi-agent coordination and its implications for an agent architecture. *Autonomous Agents and Multi-Agent Systems 1* (January 1998), 89–111.

[14] Nwana, H. S., Lee, L. C., and Jennings, N. R. Coordination in software agent systems. *The British Telecom Technical Journal 14*, 4 (1996), 79–88.

[15] Robison, G. E. Regulation of division of labor in insect societies. *Annual Review of Entomology 37* (1992), 637–665.

[16] Schillo, M., Fley, B., Florian, M., Hillebrandt, F., and Hinck, D. Self-organization in multiagent systems. In *Third International Workshop on Modelling Artificial Societies and Hybrid Organizations (MASHO)* (2002).

[17] Shehory, O., Sycara, K., Chalasani, P., and Jha, S. Agent cloning: an approach to agent mobility and resource allocation. *IEEE Communications Magazine 36*, 7 (1998), 58–67.

[18] So, Y., and Durfee, E. An organizational self-design model for organizational change. In *AAAI-93 Workshop on AI and Theories of Groups and Organizations: Conceptual and Empirical Research* (July 1993), pp. 8–15.

[19] Theraulaz, G., Bonabeau, E., and Deneubourg, J. Response threshold reinforcement and division of labour in insect societies. In *Proceedings of the Royal Society of London* (2 1998), vol. 265, pp. 327–332.

[20] Wagner, T., Guralnik, V., and Phelps, J. A key-based coordination algorithm for dynamic readiness and repair service coordination. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems* (2003), ACM Press.

[21] Wagner, T.and Lesser, V. R. Design-to-criteria scheduling: Real-time agent control. In *Proceedings of AAAI 2000 Spring Symposium on Real-Time Autonomous Systems* (Stanford, CA, March 2000), pp. 89–96.