eXtreme-Ants: Ant Based Algorithm for Task Allocation in Extreme Teams

Fernando dos Santos and Ana L. C. Bazzan
PPGC – Universidade Federal do Rio Grande do Sul
Caixa Postal 15064, CEP 91501-970, Porto Alegre, RS, Brasil
{fsantos,bazzan}@inf.ufrgs.br

ABSTRACT

This paper addresses the problem of multiagent task allocation in extreme teams. An extreme team is composed by a large number of agents with overlapping functionality operating in dynamic environments with possible inter-task constraints. We present an approximate algorithm for task allocation in extreme teams, called eXtreme-Ants. The algorithm is inspired in the division of labor in social insects and in the process of recruitment for cooperative transport observed in ant colonies. The model of division of labor offers fast and efficient decision-making, while the recruitment ensures the allocation of constrained tasks that require simultaneous execution. We show that eXtreme-Ants outperforms other two algorithms regarding communication and computational effort.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Algorithms

Keywords

Multiagent Task Allocation, Swarm Intelligence

1. INTRODUCTION

How to efficiently allocate tasks among agents in large-scale and dynamic environments? A large-scale environment means thousands of agents that must coordinate themselves to allocate and perform the available tasks. Scerri *et al.* call these scenarios *extreme teams* [7]. Task allocation in extreme teams is associated with four features: (i) dynamic environments, in which task can appear and disappear; (ii) agents perform multiple tasks given their available resources; (iii) agents have overlapping functionality to perform the tasks but with differing levels of capability; and (iv) inter-task constraints can be present, imposing simultaneous execution requirements.

Extreme teams can be formalized as an extended generalized assignment problem (E-GAP) [7]. The E-GAP model captures precisely the characteristics of extreme teams and defines the solution as the allocation which maximizes a reward measure, given by the capabilities of the agents that take part of the allocation. Efficient multiagent techniques to deal with E-GAP are a prerequisite to build teams of robots to act in extreme situations. Besides the reward, the communication channel must be used in the best way possible to avoid an excessive amount of communication. Moreover, the computational effort employed by the agents to decide which tasks to accept must be as low as possible, enabling they to act in environments where the available time to make a decision is highly restricted.

Social insects (e.g. ants) have the characteristics of extreme teams. Thus, we can conclude that Nature, despite the simplicity of the insects and over years of evolution, has provided these insects with the capability to effectively act in these teams.

To perform the tasks related to the nest survival, social insects adopt a division of labor among workers. Theraulaz *et al.* [8] present a mathematical model to replicate some mechanics of division of labor. This model is based on individual response thresholds and tasks stimuli. Moreover, it is not required that individuals have complete information about the environment and there is no need of team leaders.

Simultaneous execution of tasks also exists in social insects, as for instance in some species of ants. The task in question is the transportation of large preys. Instead of seize and transport individually a large prey, some species form groups of ants to cooperatively transport a prey. These groups are formed via a process called recruitment [2]. In this sense, the large prey can be seen as a set of interdependent subtasks, where each one is simultaneously executed by an ant.

We propose a multiagent approximate task allocation algorithm, called eXtreme-Ants, which is inspired in the division of labor in social insects and in the process of recruitment present in ants. Agents running eXtreme-Ants are efficient to act in extreme teams, with low computational effort and communication. We empirically evaluate eXtreme-Ants in a domain independent simulator and compare it with two other algorithms that are GAP-based: Swarm-GAP [1] and LA-DCOP [7]. The algorithm eXtreme-Ants achieves total rewards close to the ones achieved by LA-DCOP, but with lower communication and computational effort. Regarding Swarm-GAP, eXtreme-Ants yields better total rewards, particularly in the presence of inter-task constraints that impose simultaneous execution.

The remaining of this paper is organized as follows. Section 2 discusses the GAP and its extension (E-GAP), the model of division of labor in social insects, and the process of recruitment used by ants to cooperatively transport large preys. Section 3 details other two algorithms for dealing with GAP-based task allocation, Swarm-GAP and LA-DCOP. Section 4 presents the proposed algorithm. Section 5 presents the empirical evaluation via a series of experiments. Finally, section 6 points out the conclusion and future directions.

2. BACKGROUND

2.1 GAP and EGAP models

The generalized assignment problem (GAP) is a model used to formalize the multiagent task allocation problem [3]. A GAP is composed by a set $\mathcal{J}$ of tasks to be performed by a set $\mathcal{I}$ of agents. Each agent $i \in \mathcal{I}$ has a capability to perform each task $j \in \mathcal{J}$ denoted by $Cap(i, j) \rightarrow [0, 1]$. Each agent also has a limited amount of resource $i.res$ and uses $Res(i, j)$ when performs task j. An allocation matrix M is used to represent the allocation, where m_{ij} is given by Equation 1.

$$m_{ij} = \begin{cases} 1 & \text{if } i \text{ performs } j \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

The goal is to find M that maximizes the allocation reward, which is given by the agents' capabilities, as shown in Equation 2.

$$M = \underset{M'}{\operatorname{argmax}} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} Cap(i, j) \times m'_{ij} \tag{2}$$

The allocation M must respect all agents' resources limitations (Equation 3) and each task must be allocated to at most one agent (Equation 4).

$$\forall i \in \mathcal{I}, \sum_{j \in \mathcal{J}} Res(i, j) \times m_{ij} \leq i.res \tag{3}$$

$$\forall j \in \mathcal{J}, \sum_{i \in \mathcal{I}} m_{ij} \leq 1 \tag{4}$$

The GAP model was extended by Scerri *et al* [7] to incorporate two features related to extreme teams: scenario dynamics and inter-task constraints. This extended model was called extended generalized assignment problem (E-GAP).

Inter-task constraints are interdependencies among tasks. We focus on AND constraints here, but the formalization can be extended to other constraint types as well. In the case of an AND constraint the agents only receive the reward if all constrained tasks are *simultaneously* executed. The AND constrained tasks can be viewed as a decomposition of a large task into interdependent subtasks. The execution of some subtasks does not leads to the successful execution of the large task, wasting the agents' resources and not producing the desired effect in the system. Moreover, in the case of physical robots, they can be damaged attempting to perform an effort greater than their capabilities (e.g. trying to remove a large piece of collapsed building from a blocked road).

To formalize AND constrained tasks, the E-GAP model defines a set $\bowtie \; = \{\alpha_1, \ldots, \alpha_p\}$ containing p sets α of AND constrained tasks in the form $\alpha_k = \{j_1 \wedge \ldots \wedge j_q\}$. Each AND constrained task j belongs to at most one set α_k. The number of tasks that are being performed in a set α_k is given by Equation 5.

$$x_k = \sum_{i \in \mathcal{I}} \sum_{j \in \alpha_k} m_{ij} \tag{5}$$

Let $v_{ij} = Cap(i, j) \times m_{ij}$. Given the constraints of $\bowtie$, the reward $Val(i, j, \bowtie)$ of an agent i performing the task j is given by Equation 6.

$$Val(i, j, \bowtie) = \begin{cases} v_{ij} & \text{if } \forall \alpha_k \in \bowtie, j \notin \alpha_k \\ v_{ij} & \text{if } \exists \alpha_k \in \bowtie \text{ with } j \in \alpha_k \wedge x_k = |\alpha_k| \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

To represent the dynamics in the scenario all E-GAP variables are indexed by a time step t. The goal is to find a sequence of allocations $\overrightarrow{M}$ one for each time step t, as shown in Equation 7. A delay cost function $DC^t(j^t)$ can be used to define the cost of not performing a task j at time step t.

$$\begin{aligned} f(\overrightarrow{M}) = & \sum_t \sum_{i^t \in \mathcal{I}^t} \sum_{j^t \in \mathcal{J}^t} (Val^t(i^t, j^t, \bowtie^t) \times m^t_{ij}) \\ & - \sum_t \sum_{j^t \in \mathcal{J}^t} (1 - \sum_{i^t \in \mathcal{I}^t} m^t_{ij}) \times DC^t(j^t) \end{aligned} \tag{7}$$

Furthermore, the agents' resource limitations must be respected at each time step t (Equation 8) and each task must be allocated to at most one agent (Equation 9).

$$\forall t, \forall i^t \in \mathcal{I}^t, \sum_{j^t \in \mathcal{J}^t} Res^t(i^t, j^t) \times m^t_{ij} \leq i^t.res \tag{8}$$

$$\forall t, \forall j^t \in \mathcal{J}^t, \sum_{i^t \in \mathcal{I}^t} m^t_{ij} \leq 1 \tag{9}$$

2.2 Division of Labor in Social Insects

An effective division of labor is responsible for the ecological success of insect societies. A social insect colony with hundreds of thousand members operates without the existence of explicit coordination. An individual cannot assess the needs of the colony; it just has a fairly simple local information, and no one is in charge of coordination. From individual workers aggregation, the colony behavior emerges without any type of explicit coordination or planning. The key point is the plasticity of the individuals, in other words, the existence of a behavioral flexibility. This flexibility allows the individuals to engage in different tasks responding to changing conditions in the colony.

Observations regarding this behavior are the basis of the theoretical model described by Theraulaz *et al.* [8]. In this model, interactions among members of the colony and individual perception of local needs result in a dynamic distribution of tasks. The model is based on individuals' internal response threshold related to tasks stimuli. Assuming the existence of $\mathcal{J}$ tasks to be performed, each task $j \in \mathcal{J}$ has an associated stimulus s_j. The stimulus is related to the demand for the task execution, and can be a number of encounters, a chemical concentration, or any quantitative cue sensed by individuals. Given a set of $\mathcal{I}$ individuals which can perform the tasks of $\mathcal{J}$, each individual $i \in \mathcal{I}$ has an internal response threshold θ_{ij}, which is related to the likelihood of reacting to the stimulus associated with task j.

The threshold can be seen as a genetic characteristic (also called polymorphism, which is responsible for the existence of differences in the morphologies of insects belonging the same society), or as a temporal polyethism (in which individuals of the same age tend to perform identical sets of tasks), or simply as individual variability.

In the model of Theraulaz *et al.* [8] the individual internal threshold θ_{ij} and the task stimulus s_j represent the probability (tendency) $T_{ij}(s_j)$ of the individual i to perform task j, as shown in Equation 10.

$$T_{ij}(s_j) = \frac{s_j^2}{s_j^2 + \theta_{ij}^2} \quad (10)$$

This tendency means that any individual is able to perform any task if the corresponding task stimulus is high enough to overcome the individual's internal threshold. This flexibility enables the survival of the colony in an eventual absence of specialized individuals, since other individuals start to perform the tasks when the stimulus exceeds their thresholds.

2.3 Recruitment for Cooperative Transport

In some species of ants the transportation of large preys in a cooperative way involves two or more ants that cannot do the transport alone [6]. The main purpose of the cooperative transport is to maximize the trade off between the gained energy (food) and the energy spent to take it to the nest. Further, this process speeds up the transport.

The group involved in the cooperative transport is formed by a process called *recruitment*. When a single scout ant discovers a prey, it firstly attempts to seize and transport it individually. After unsuccessful attempts, a recruitment process starts. To recruit nestmates the ants employ a mechanism called long-range recruitment (LRR). Some species also employ a second mechanism, called short-range recruitment (SRR). In both mechanisms the ants use communication through the environment (stigmergy) [2].

In the SRR the scout that discovers the prey releases a secretion. Shortly thereafter, nestmates in the vicinity are attracted to the prey location by the secretion odor. When the prey cannot be moved by the scout and the ants recruited via SRR, one of the ants begins the LRR. Hölldobler *et al* report that SRR is sufficient to summon enough ants to transport the prey in the majority of the cases [2].

In the LRR the scout ant that discovers the prey returns to the nest to recruit nestmates. In the course towards the nest, the scout lays a pheromone trail. Nestmates encountered in the course are stimulated by the scout via direct contact. After, the stimulated nestmate also begins to lay a pheromone trail even though it had not yet experienced the prey stimulus itself, thus establishing a chain of communication among nestmates. When the scout arrives at the nest, nestmates are attracted by the pheromone and run to the prey site.

After the recruited ants arrive where the prey is, they begin the cooperative transport. The number of ants engaged in the transport is regulated at the prey site and depends on its characteristics, such as weight, size, rotational forces, and difficulty to move. When the number of ants present is not enough to move the prey, more ants are recruited by one of the aforementioned processes, until the prey is successful transported. Although in a more economic approach the scout ant should recruit an exact number or nestmates, it was suggested that the scout cannot make a fine assessment of the number of ants required to retrieve the prey [6]. Therefore, the most effective strategy may be to recruit a constant number of ants followed by a regulation of the group size during the transport.

In summary, the recruitment for cooperative transport consists in three steps:

1. The scout ant that discovered the prey starts the recruitment, inviting netstmates with pheromones;
2. The ants that accept to join the recruitment move to the prey site;
3. The size of the transportation group is regulated to the prey characteristics.

3. RELATED WORK

The research regarding multiagent task allocation has shown significant advances in the last few years (auction, contracting, coalition formation, organizations, etc). A complete review of the subject is outside the scope of this paper. We just mention that auctions are normally centralized mechanisms, in which agents put bids to an auctioneer, depending on their capabilities and resources. After receiving all bids, the auctioneer makes the allocation of the tasks among the bidders. Centralized auctioneers can have severe bottlenecks. Further, auctions require high amounts of communication [9].

Here we concentrate in the line of research that deals with coordination for task allocation. Within this research line, one approach is the framework of distributed constraint optimization problem (DCOP). A DCOP consists of a set of variables that can assume values from a discrete domain. Each variable is assigned to one agent which has the control over its value. The goal of the agents is to choose values for the variables to optimize a global objective function. This function can be described as an aggregation over a set of cost functions related to pairs of variables. A DCOP can be represented by a constraint graph, where vertices are variables and edges are cost functions between variables. Despite the existence of complete algorithms for DCOP, such as Adopt [4], and DPOP [5], these are not efficient to deal with the problem of multiagent task allocation. Due to dense constraint graphs generated to represent the problem, Adopt and DPOP demand high communication and space respectively.

To deal with the particular characteristics of extreme teams, Scerri *et al.* present an approximate algorithm called Low-communication Approximate DCOP (LA-DCOP), which uses tokens to represent tasks and further minimize communication [7]. An agent decides whether or not to accept a task based both on its capability and on a threshold associated to the task. To deal with inter-task constraints, LA-DCOP uses a differentiated kind of token, called potential token. If an agent in LA-DCOP is able to allocate more than one task, it must select the ones that maximize its capability given its resources. This selection is a maximization problem, which can be reduced to a binary knapsack problem (BKP), proved to be NP-Complete. The computational complexity of LA-DCOP thus depends of the complexity of its function to deal with the BKP.

Another approximate algorithm which can deal with extreme teams is the Swarm-GAP [1]. An agent in Swarm-GAP decides whether or not to accept a task based on the model of division of labor used by social insects colonies. This algorithm also uses tokens to represent tasks. To deal with inter-task constraints, the agents in Swarm-GAP just increase the tendency to allocate a constrained task by a factor called execution coefficient. The execution coefficient is computed using the rate between the number of constrained tasks which are allocated and the total number of constrained tasks.

4. eXtreme-Ants

4.1 Basic Ideas

eXtreme-Ants is an approximate algorithm that solves E-GAPs. Agents running eXtreme-Ants use the model of division of labor in social insects (Equation 10) to decide whether or not to perform the tasks. The notation used hereafter to represent agents, tasks, and all other terms is the one from the E-GAP model (Section 2.1) and from the model of division of labor (Section 2.2). The internal threshold θ_{ij} of an agent i related to a task j is defined via the concept of polymorphism and corresponds to the inverse of the capability $Cap(i,j)$, as shown in Equation 11. If an agent is not capable regarding a particular task, then its internal threshold is set to infinity, avoiding the allocation of the task to the agent. This makes sense if we consider the capability as a kind of morphism. For example, a fire brigade agent is more capable of fighting fires than rescuing civilians. Thus it have low thresholds related to fire fighting tasks and high thresholds to rescue civilians.

$$\theta_{ij} = \begin{cases} 1 - Cap(i,j) & \text{if } Cap(i,j) > 0 \\ \infty & \text{otherwise} \end{cases} \tag{11}$$

Each task $j \in \mathcal{J}$ has an associated stimulus s_j. The stimulus controls the allocation of the tasks by the agents. Low stimuli mean that the tasks will only be accepted by agents with low thresholds (thus, more capable). High stimuli increase the chance of the tasks to be accepted, even by agents with high thresholds (less capable).

Since in the E-GAP each task must be allocated to at most one agent, eXtreme-Ants uses tokens to represent the tasks and ensure this mutual exclusion constraint. A token contains a list of tasks it represents. An agent that holds a token has the exclusive right to accept the tasks contained in a token. If the agent does not accept all tasks, it passes the token to another teammate. In this way, eXtreme-Ants avoid conflicts in the allocation and reduces the communication.

To deal with AND constraints among tasks, agents in eXtreme-Ants reproduce the recruitment process of ants. When an agent detects that it is not capable of accepting all AND constrained tasks perceived, it recruits other agents to form a group committed with the simultaneous execution.

4.2 Algorithm Details

Algorithms 1 and 2 present the details of our approach. Each agent i reacts to two events: perception of a set of tasks (which can be AND constrained), and receipt of messages. In the following we detail the algorithm operation.

When the agent perceives a set $\mathcal{J}$ of tasks (line 1) it creates a token to store the perceived tasks. The agent then decides whether or not to accept the tasks contained in the token, given its tendency and the available resources (lines 21-30). When a task is allocated to agent i, the available resources at i is decreased by the amount required. If some tasks contained in the token remains unallocated, the agent sends the token to a randomly selected teammate. As in [7], to avoid agents passing token back and forth, each token maintains a list of visited agents. The token can revisit an agent only after all were visited.

Algorithm 1: eXtreme-Ants for agent i

```
1  when perceived set of tasks J
2      token := newToken();
3      add each j ∈ J to token.tasks;
4      evaluateToken(token);
5  end

6  when perceived set of AND constrained tasks α_k
7      /* firstly try to accept all tasks by itself */
8      foreach j ∈ α_k do
9          if roulette() < T_ij and i.res ≥ Res(i, j) then
10             accept task j and decrease i.res;
11         end
12     end
13     if there are non accepted tasks in α_k then
14         discard previous accepted tasks of α_k (lines 7-9);
15         performsRecruitment(α_k);
16     end
17 end

18 when received token
19     evaluateToken(token);
20 end

21 procedure evaluateToken(token)
22     /* decides whether or not to accept the tasks */
23     foreach j ∈ token.tasks do
24         if roulette() < T_ij and i.res ≥ Res(i, j) then
25             accept task j and decrease i.res;
26             token.tasks := token.tasks − j;
27         end
28     end
29     if there are non accepted tasks in token.tasks then
30         send token to a teammate;
31     end
32 end
```

When the agent perceives a set of AND constrained tasks α_k (line 6), it acts as a scout ant. Firstly it attempts to accept all the constrained tasks. If it fails, it begins a recruitment process. We develop a protocol that reproduces the three steps of the recruitment process of ants via the use of messages. There are five kinds of messages used in the recruitment protocol of eXtreme-Ants:

request: to invite an agent to join the recruitment for a task $j \in \alpha_k$ and to commit to it;

committed: to inform that the agent joins the recruitment for a task $j \in \alpha_k$ and commits to it;

engage: to inform that the agent was indeed selected to perform the task $j \in \alpha_k$;

release: to inform that the agent was not selected to perform the task $j \in \alpha_k$ and must uncommit with it.

timeout: to inform that a request for a task $j \in \alpha_k$ reaches its timeout.

For the first step of the recruitment, the scout agent sends a certain number of **request** messages for each task $j \in \alpha_k$ (lines 33-37). These requests are sent to randomly

Algorithm 2: eXtreme-Ants for agent i (cont.)

33 **procedure performsRecruitment**(AND set α_k)
34   **repeat**
35   j := pick a task from α_k;
36   send("request", j) to a teammate;
37   **until** *the maximum number of requests sent for all* $j \in \alpha_k$ *is reached* **or** *the recruitment for* α_k *is finished or aborted.*
38 **end**

39 **when received** ("request", j) from agent i_s
40   /* decides whether or not to commit */
41   **if** $roulette() < T_{ij}$ **and** $i.res \geq Res(i, j)$ **then**
42   commit to j;
43   send("committed", j) to i_s;
44   **else**
45   **if** *request timeout reached* **then**
46   send("timeout", j) to i_s;
47   **else**
48   forward("request", j) to a teammate;
49   **end**
50   **end**
51 **end**

52 **when received** ("committed", j) from i_c
53   α_k := AND group which contains j;
54   **if** *recruitment for* α_k *is finished or aborted* **then**
55   send("release", j) to i_c; **return**
56   **end**
57   **if** *at least one agent committed with each* $j \in \alpha_k$ **then**
58   recruitment for α_k is finished;
59   /* forms the group of engaged agents */
60   **foreach** $j \in \alpha_k$ **do**
61   pick a committed agent i_p with probability proportional to $Cap(i_p, j)$;
62   send("engage", j) to i_p;
63   send("release", j) to non selected agents;
64   **end**
65   **end**
66 **end**

67 **when received** ("engage", j)
68   accept task j and decrease $i.res$;
69 **end**

70 **when received** ("release", j)
71   uncommit to j;
72 **end**

73 **when received** ("timeout", j)
74   α_k := AND group which contains j;
75   **if** *the number of received timeouts for each* $j \in \alpha_k$ *is equal the number of requests sent* **then**
76   recruitment for α_k is aborted;
77   **foreach** $j \in \alpha_k$ **do**
78   send("release", j) to committed agents;
79   **end**
80   **end**
81 **end**

selected teammates and act as the pheromone released in the air (SRR) or released on the way to the nest (LRR). As it occurs with the scout ant, which recruits a fixed number of nestmates independently of prey characteristics, eXtreme-Ants fixes a maximum number of requests that must be sent for each AND constrained task. This maximum number must be experimentally determined to maximize the total reward.

In the second step, the agents must decide whether or not to join the recruitment. When an agent receives a request originated by a scout agent i_s for a task j (lines 39-48), it uses the tendency (Equation 10) to decide if it accepts the request and then joins the recruitment, avoiding double commitment. If the request is accepted, the agent commits to perform the task, reserving the amount of resources required by the task. A **committed** message is send to the scout to inform the commitment. If the request is not accepted, the agent forwards it to another randomly selected teammate, reproducing the chain of communication present in the LRR.

In the third step, the size of the group of agents engaged in the simultaneous execution of the AND constrained tasks must be regulated. In eXtreme-Ants the regulation is done by the scout agent. When the scout receives enough commitments for each constrained task $j \in \alpha_k$ (line 57), it forms the group of agents which will execute the tasks simultaneously. Following the E-GAP definition, just one agent must be selected among those committed for each constrained task. The scout then performs a probabilistic selection, picking an agent i_p with probability proportional to its capability $Cap(i_p, j)$. The scout then informs i_p that it was the selected one and thus must engage in the execution of j (via an **engage** message, line 62). All other non selected agents are released (via a **release** message, line 63). Agents that commit to an already allocated task are also released to avoid deadlocks. At this moment the recruitment is finished. As the result, a group of agents is formed, in which each agent is allocated to a task $j \in \alpha_k$, enabling the simultaneous execution of all AND constrained tasks in α_k.

After the group of engaged agents is formed, the requests not yet accepted by some agent become obsolete. To avoid agents passing obsolete requests back and forth, eXtreme-Ants introduces a timeout mechanism. The timeout is a number of agents that a recruitment request is allowed to visit. When the timeout of a request is detected (line 45), the scout is notified via a **timeout** message. When the scout agent receives a timeout notification for all requests sent, it aborts the recruitment and releases the committed agents.

It is important to note that due to the algorithm asynchronism, the scout agent can perform another actions while the recruitment occurs. These actions comprise the perception of another tasks, and even a recruitment for other AND constrained task groups. Although eXtreme-Ants reproduce the inter-agent communication via messages, it can be easily modified to use some kind of indirect communication (e.g. pheromones) when the environment allows it.

5. EXPERIMENTS AND RESULTS

We compare eXtreme-Ants to Swarm-GAP[1] and LA-DCOP[7]. We have evaluated eXtreme-Ants in a domain independent simulator that allows experimentation with large number of agents and tasks, performing experiments similar to Swarm-GAP and LA-DCOP which have also used such a simulator.

Basically, each experiment consists of 2000 tasks, grouped in five classes, where each class determines the task characteristics. The number of agents varies from 500 to 4000. This means that the load (ratio between tasks and agents) is 4 in the first case and 0.5 in the latter. Each agent has a 60% probability of having a non-zero capability for each class. In this case the agent has a randomly assigned capability ranging from 0 to 1. Regarding the AND constraints among tasks, 60% of the tasks are related in groups of five tasks. The simulated communication channel is reliable (every sent message is received) and fully connected (each agent is connected to every other agent). Each experiment consists of 1000 time steps. The total number of tasks is kept constant. At each time step, each task has a probability of 10% to be replaced by a task potentially requiring a different capability. The tasks are persistent, which means that non allocated tasks are kept in the next time step. At each time step, each token or message is allowed to move from one agent to another only once. Despite that each task can have a particular stimulus value, we adopt the same value for all tasks. Each datapoint in the plots we show here represents the average over 20 runs. The standard deviations are not shown due to their low values.

As defined by the E-GAP, the goal is to maximize the total reward, which is the sum of the reward at each time step over the length of the simulation. The first experiment compares the total reward achieved by each algorithm. The parameters used for each algorithm are shown in Table 1. These parameters, which were selected among a large set of tested values, yield the maximum total reward in each scenario, and will be used in the comparisons. Additionally, in the case of eXtreme-Ants the total rewards are obtained with five recruitment requests for each AND constrained task, and with a timeout of 20 visited agents.

Table 1: Parameter values that yield the maximum total reward for each algorithm.

Agents	eXtreme-Ants (Stimulus)	Swarm-GAP (Stimulus)	LA-DCOP (Threshold)
500	0.3	0.2	0.0
1000	0.3	0.3	0.4
1500	0.2	0.2	0.6
2000	0.2	0.2	0.6
2500	0.2	0.2	0.6
3000	0.2	0.2	0.6
3500	0.2	0.2	0.7
4000	0.2	0.2	0.7

Figure 1 shows the total rewards achieved by each algorithm. On average, eXtreme-Ants yields rewards that are 25% higher than those of Swarm-GAP and 19% lower than those of LA-DCOP (t-test, 99% confidence).

When an agent accepts a task, it uses an amount of its resources. Thus, the agents must avoid to waste their resources accepting tasks that do not yield any reward (e.g. tasks that belong to an AND constrained, but are not simultaneously accepted). The second experiment, shown in Figure 2, compares the percentage of resources used by each agent to accept tasks at each time step. As we can see, all algorithms use almost the same percentage of resources. There is no significative difference between eXtreme-Ants and Swarm-GAP in the cases with 3000, 3500, and 4000 agents, and between eXtreme-Ants and LA-DCOP in the cases with 500 and 1000 agents (t-test, 99% confidence).

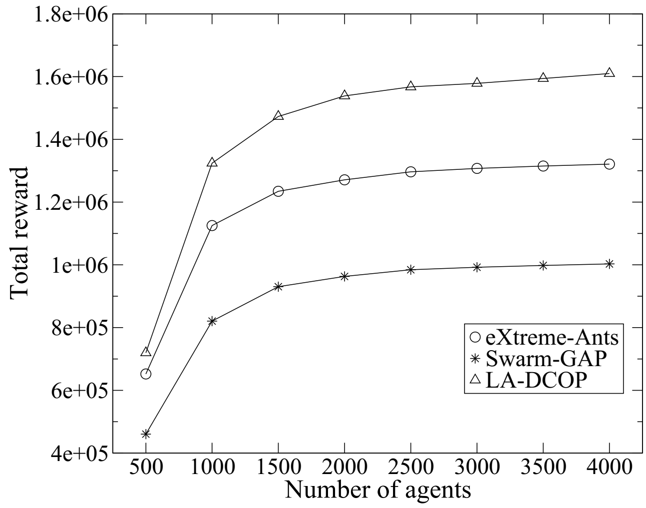

Figure 1: Total reward versus the number of agents.

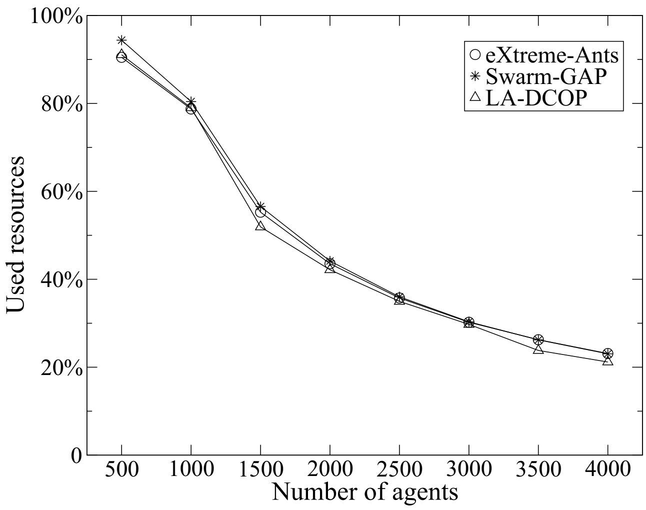

Figure 2: Percentage of resources used by each agent at each time step to allocate tasks.

From these two experiments, we can see that despite the fact that agents in Swarm-GAP use almost the same percentage of resources, the achieved total rewards are worse than those achieved by eXtreme-Ants and LA-DCOP. This is due to the way Swarm-GAP deals with AND constrained tasks. The use of an execution coefficient (see Section 3) does not ensure the simultaneous allocation of the AND constrained tasks. Thus, the agents use their resources to accept tasks, but this allocation does not translate into a reward. Both eXtreme-Ants and LA-DCOP outperform Swarm-GAP regarding the total reward. This is due to the existence of explicit coordination mechanisms to deal with AND constrained tasks, ensuring their simultaneous allocation.

LA-DCOP yields higher rewards than eXtreme-Ants because each agent maximizes its capability when accepting the tasks, taking into account the available resources. On the other hand, agents in eXtreme-Ants make a simple one-shot decision to allocate tasks. The maximization leads to a better exploitation of the agents' resources. However, as we show in the next experiments, there is a tradeoff between the achieved reward and the communication/computational effort.

In the next experiment, shown in Figure 3, we compare the amount of communication used in each algorithm. The communication is measured as the sum of messages sent by the agent over all time steps, regardless of message type (e.g. token, recruitment request, etc.). The results are statistically significant at 99% confidence t-test. On average, agents in eXtreme-Ants sent 121% fewer messages than those in LA-DCOP and 80% more messages than those in Swarm-GAP. The smallest difference to LA-DCOP occurs with 3000 agents. Even in this case LA-DCOP sends 66% more messages than eXtreme-Ants.

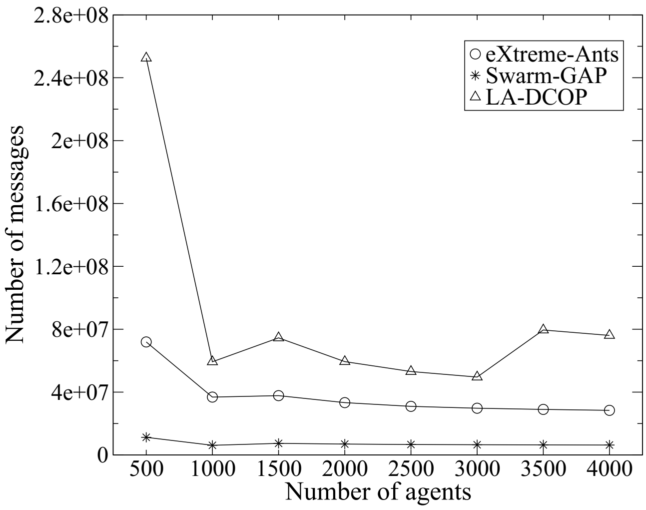

Figure 3: Total number of messages sent versus the number of agents.

As mentioned Swarm-GAP sends fewer messages than eXtreme-Ants and LA-DCOP due to its difficulty to deal with AND constrained tasks. The absence of an explicit coordination mechanism to ensure the simultaneous allocation leads to a small number of messages, but has a great impact in the total reward of Swarm-GAP.

The last experiment aims at evaluating the computational effort of the agents in each algorithm. We define the computational effort as the number of evaluated tasks by each agent at each time step. This number is computed as follows. Each time an agent decides whether or not to accept a task, an internal counter is incremented. In the case of eXtreme-Ants and Swarm-GAP, each probabilistic decision causes just one increment in the counter. On the other hand, since an agent in LA-DCOP solves a BKP to decide which tasks to accept, the increment in the counter is related to the number of retained tasks. To solve a BKP our implemented version of LA-DCOP uses a greedy approach, which sorts the tasks by the agent's capability and then selects the tasks to accept constrained by the agent's resources. If n is the number of retained tasks, the sort causes a increment of $n \log n$ in the counter, followed by a increment of at most n to select the accepted tasks.

Low computational effort means that the agents are more efficient to act in environments in which the available time to make a decision is restricted. Figure 4 shows the average computational effort of each agent at each time step. The external plot emphasizes the area which concentrates the majority of the points. The internal plot shows the full area just to present the points not shown in the external plot. The results are statistically significant at 99% confidence t-test.

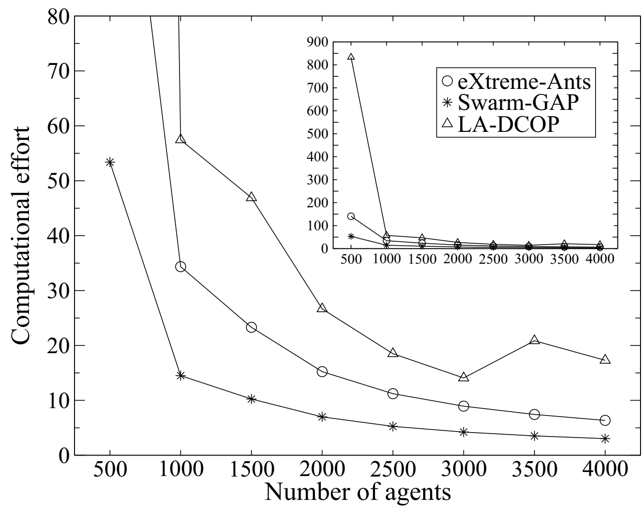

Figure 4: Computational effort as the number of evaluated tasks by each agent at each time step.

The computational effort of Swarm-GAP is, on average, 55% lower than those from eXtreme-Ants. Since in Swarm-GAP there is no explicit coordination mechanism to deal with AND constrained tasks, the agents do not have to make additional evaluations regarding the simultaneous allocation of constrained tasks, reducing the computational effort of Swarm-GAP. However, the absence of such mechanism affects the total reward, as shown previously. The higher computational effort of both eXtreme-Ants and LA-DCOP are due to the presence of an explicit coordination mechanism to deal with constrained tasks.

eXtreme-Ants outperforms LA-DCOP, with computational efforts on average 151% lower than those from LA-DCOP. The most significant result is for the case of 500 agents, in which the computational effort of LA-DCOP is 493% higher than that of eXtreme-Ants.

As shown in the experiments, the probabilistic allocation of eXtreme-Ants, based on the model of division of labor, reduces the amount of communication and the computational effort. The reduction in the computational effort is due to the simple one-shot decision, which does not require any local maximization. The low computational effort causes the reduction in the number of messages sent, since in LA-DCOP the tasks which are not selected in the local maximization are sent do other agents. In both eXtreme-Ants and LA-DCOP the presence of an efficient coordination mechanism to deal with inter-task constraints leads to better total rewards regarding Swarm-GAP.

Finally, we emphasize that the choice of one particular algorithm must be related with the constraints of the scenario. When the total reward is a key point and there are no constraints in the communication and in the time the agents have to make a decision, LA-DCOP is a good choice. On the other hand, in scenarios with such constraints eXtreme-Ants is more appropriate. It achieves low total rewards, but the decision is faster and there is a better use of the communication channel.

6. CONCLUSIONS

In this paper we have presented a multiagent approximate algorithm for task allocation in extreme teams, called eXtreme-Ants. The algorithm is inspired in the division of labor in social insects and in the process of recruitment present in ants that transport preys cooperatively.

The experimental results show that the use of the model of division of labor to decide whether or not to allocate the tasks allows the agents to make reasonable coordinated actions. Since the decision is probabilistic, it is fast, efficient, and requires a reduced communication and computational effort, enabling the agents to act in environments where the available time to make a decision is highly restricted. Moreover, the incorporated recruitment process provides efficient allocation of constrained tasks that requires simultaneous execution. This avoid that agents waste they resources and leads to better total rewards. The efficiency of eXtreme-Ants regarding communication and computational effort suggests that techniques which are inspired in social insects can be considered for multiagent task allocation.

We intend to work in the direction of changing the stimuli values dynamically, indicating different priorities in the execution of the tasks. More than one kind of resource for an agent can also be considered. Besides, these resources can change over time, as for instance, a battery charge of a robot. We also intend to evaluate the performance in unreliable communication channel, with failures and noises, and to apply this approach in the RoboCup Rescue simulator.

7. ACKNOWLEDGMENTS

This research is partially supported by the Air Force Office of Scientific Research (AFORS) (grant number FA9550-06-1-0517) and by the Brazilian National Council for Scientific and Technological Development (CNPq).

8. REFERENCES

[1] P. R. Ferreira, Jr., F. Boffo, and A. L. C. Bazzan. Using swarm-gap for distributed task allocation in complex scenarios. In N. Jamali, P. Scerri, and T. Sugawara, editors, *Massively Multiagent Systems*, number 5043 in Lecture Notes in Artificial Intelligence, pages 107–121. Springer, Berlin, 2008.

[2] B. Hölldobler, R. C. Stanton, and H. Markl. Recruitment and food-retrieving behavior in *Novomessor* (formicidae, hymenoptera). *Behavioral Ecology and Sociobiology*, 4(2):163–181, 1978.

[3] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, New York, NY, USA, 1990.

[4] P. J. Modi, W.-M. Shen, M. Tambe, and M. Yokoo. An asynchronous complete method for distributed constraint optimization. In *Proc. of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 161–168, New York, USA, 2003. ACM Press.

[5] A. Petcu and B. Faltings. A scalable method for multiagent constraint optimization. In L. P. Kaelbling and A. Saffiotti, editors, *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pages 266–271, Edinburgh, Scotland, August 2005. Professional Book Center.

[6] S. K. Robson and J. F. A. Traniello. Resource assessment, recruitment behavior, and organization of cooperative prey retrieval in the ant *Formica schaufussi* (hymenoptera: Formicidae). *Journal of Insect Behavior*, 11(1):1–22, 1998.

[7] P. Scerri, A. Farinelli, S. Okamoto, and M. Tambe. Allocating tasks in extreme teams. In *Proc. of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 727–734, New York, USA, 2005. ACM Press.

[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.

[9] Y. Xu, P. Scerri, K. Sycara, and M. Lewis. Comparing market and token-based coordination. In *Proc. of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006)*, pages 1113–1115, New York, NY, USA, 2006. ACM.