Swarm-GAP: A Swarm Based Approximation Algorithm for E-GAP

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

Coordinating efforts of teams of professionals to search and rescue victims of emergency situations is a key point in disaster management. The disasters scenarios have been modelled as a distributed and large scale Extended Generalized Allocation Problem (E-GAP). We propose a novel algorithm to approximate the solution of E-GAP based on the theoretical division of labor models from colonies of social insects (swarms), called Swarm-GAP. This algorithm uses a probabilistic decision model, based on the social insects tendency of performing certain tasks. Each agent has an individual threshold used together with a stimulus associated with the tasks to compute that tendency. The Swarm-GAP means low communication and uses simple mechanisms. We show that the Swarm-GAP achieves rewards very close to the ones achieved by a greedy centralized approach.

Keywords

large scale multiagent systems, task and resource allocation in agent systems, collective and emergent agent behavior

1. INTRODUCTION

Agents technology is part of the global effort of improving disaster management. Multiagent systems offer a wide range of tools and techniques that have been used as a new way to design information systems to provide decision support in emergency situations.

When fire fighters, paramedics, and other professionals work in teams to search and rescue victims of disaster these teams must coordinate their actions considering the others' activities and the general performance of their efforts. This kind of situation is dramatically dynamic, with new fire spots appearing while others are extinguished, streets closing by landslide or flooding, etc. Besides, disaster always involves a large number of people, both as victims as well as rescuers.

The motivation for studying coordination among actors in a rescue scenario can also be found in the multiagent systems' community. The research regarding multiagent systems coordination through distributed task allocation has shown significant advances in the last few years [14, 11, 15]. Dynamic and large scale environments introduce new challenges in this research. Traditional approaches are very inefficient in this kind of scenario and there are few new studies regarding this subject [17].

The task allocation problem in disaster rescue can be seen as a Extended Generalized Assignment Problem (E-GAP)[17]. E-GAP is an allocation problem which examines the assignment of tasks to agents maximizing a total reward. This total reward is computed dynamically. Tasks demand resources and classes of capabilities to be allocated. The agents have a certain amount of resource and different levels associated with each class of capability. The tasks can be interrelated and their total number or required capabilities can change over time.

We propose a novel approximation algorithm to E-GAP based on the theoretical models of division of labor in social insects colonies, called Swarm-GAP. This algorithm intend to be simple and effective to solve large scale instances of E-GAP where the allocation is achieved in a distributed fashion. Cooperative agents running Swarm-GAP are allowed to coordinate its actions with low communication.

We focus on a approach based on colonies of social insects (also called swarms), where plenty of evidences of ecological success exist, despite the apparent lack of explicit coordination. These hundreds of thousands insects adapt to the changes in the environment and to the needs of the colony using the plasticity in division of labor. There are well experimented theoretical models that captures this plasticity. We adapt and extend this models to use in Swarm-GAP.

Swarm-GAP was experimented in a simple centralized simulation environment. First, we empirically define the parameters of the adopted swarm model that maximizes the system reward. After, we compare the performance of the Swarm-GAP with a centralized greedy algorithm. As ex-

pected, the centralized approach outperforms Swarm-GAP. However, our algorithm performs well in the E-GAP scenario achieving rewards 20% lower (on average) than the greedy ones. Finally, we measure the impact in the Swarm-GAP performance when tasks are several interrelated. Our extension in the swarm model to deal with the tasks interrelation improves the average performance in 25%.

This paper is organized as follows: Section 2 describes the E-GAP is details; Section 3 discuss the relevant aspects of labor division in social insect colonies; Section 4 introduces the Swarm-GAP; Section 5 shows the empirically evaluation of Swarm-GAP; and Section 7 presents our conclusions and future directions of this work.

2. GAP AND E-GAP

The Generalized Assignment Problem (GAP) [18] is a general allocation problem which examines the assignment of tasks to agents, respecting the agents capacities, maximizing a total reward. The GAP was extended by [17] to capture dynamic domains and interdependencies among tasks. This extension is called Extended-GAP (E-GAP). As said before, the task allocation in large scale and dynamic environments can be modelled as an E-GAP. Next, we describe and formalize GAP and its extension, which is the focus of this paper.

The gap can be formalized as follows. Let us define $\mathcal{J}$ as the set of tasks to be allocated and $\mathcal{I}$ the set of agents. Each agent $i \in \mathcal{I}$ has a limited amount of resource r_i to perform all tasks (a single type of resource is used). When a task $j \in \mathcal{J}$ is executed by agent i, task j consumes c_{ij} units of i's resource. Each agent i also has a capability to perform each task j given by k_{ij} $(0 \leq k_{ij} \leq 1)$.

The allocation matrix A, where a_{ij} is the value of the i-th row and j-th column, is given by Equation 1.

$$a_{ij} = \begin{cases} 1 & if \text{ j } is\ allocated\ by \text{ i} \\ 0 & otherwise \end{cases} \quad (1)$$

The GAP examines the maximum profit matrix A, which maximizes the system reward given by 2, subject to the agents resource limitations and the constraint of having only one agent allocated to each task.

$$A = argmax_{A'} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} k_{ij} * a'_{ij} \quad (2)$$

such that

$$\forall i \in \mathcal{I},\ \sum_{j \in \mathcal{J}} c_{ij} * a_{ij} \leq r_i$$

and

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

E-GAP improves GAP in two different ways:

Allocation constraints among tasks. Tasks in E-GAP can be interrelated by an and constraint. All interrelated tasks by this constraint must be allocated at the same time to be considered by the reward computation.

Following [17], let us define $\bowtie = \{\alpha_1, ..., \alpha_p\}$, where $\alpha_k = \{j_{k_1}, ..., j_{k_q}\}$ denotes the k-th set of AND constrained tasks. Thus, the partial reward w_{ij} for allocating task j to agent i is given by Equation 3.

$$w_{ij} = \begin{cases} k_{ij} * a_{ij} & if\ \forall \alpha_k \in \bowtie,\ j \notin \alpha_k \\ k_{ij} * a_{ij} & if\ \exists \alpha_k \in \bowtie\ with\ j \in \alpha_k \wedge \\ & \forall j_{k_u} \in \alpha_k,\ a_{ij_{k_u}} \neq 0 \\ 0 & otherwise \end{cases} \quad (3)$$

Reward dynamically computed over time. The total reward W is computed in E-GAP as the sum of the reward in the last t time steps. In this case, the sequence of allocations over time is considered against the single allocation used into the GAP. Additionally, a delay cost d_j can be used in order to punish the agents when then task j was not allocated at time t. The objective of the E-GAP is to maximize W given by Equation 4.

$$W = \sum_t \sum_{i^t \in \mathcal{I}^t} \sum_{j^t \in \mathcal{J}^t} w^t_{ij} * a^t_{ij} - \sum_t \sum_{j^t \in \mathcal{J}^t} (1 - a^t_{ij}) * d^t_j \quad (4)$$

such that

$$\forall t \forall i^t \in \mathcal{I}^t,\ \sum_{j^t \in \mathcal{J}^t} c^t_{ij} * a^t_{ij} \leq r^t_i$$

and

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

3. DIVISION OF LABOR IN SWARMS

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 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 with hundreds of thousand of members operates without any explicit coordination. An individual worker cannot access 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 feature of this emergent behavior is the plasticity in division of labor inside the colony [16]. Colonies respond to changing conditions by adjusting the ratios of individual workers engaged in the various tasks.

Theraulaz et al. [19] present a model for task allocation inspired on the plasticity of division of labor in colonies of social insects [16]. 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. 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. 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.

Assuming the existence of $\mathcal{J}$ tasks to be performed, each task j have a s_j stimulus associated. If $\mathcal{I}$ 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:

$$T_{\theta_{ij}}(s_j) = \frac{s_j^2}{s_j^2 + \theta_{ij}^2} \tag{5}$$

The physical specialization in social insect societies is called morphological polyethism or simply polymorphism by biologists. The polymorphism has a key rule to determine the division of labor in ant colonies [9]. The soldiers of an ant colony are usually the largest ants with big heads and sword-like jaws. Usually, other ants are medium sized and forage for food. This is quite adequate to its normal activities. By differentiating individual thresholds, it is possible to capture this physical variety in the theoretical model. The individual thresholds for the tasks decreases proportionally to the individual capabilities to perform this tasks. In this sense, individuals with large capability for a set of tasks have a higher tendency to perform tasks of this set.

The social insects behavior seems to fit the requirements of dynamic and large scale environments. Agents can decide which task to perform just as social insects do. The Swarm-GAP algorithm is based on the theoretical models shown in this section, including the probabilistic decision process guided by the tendency and the polymorphism.

4. THE SWARM-GAP

The aim of Swarm-GAP is to allow the agents to decide individually which task to execute, without any kind of communication. As in [17], we assume that the communication does not fail, and that the agents know all tasks that should be executed as well as the number of agents involved in the allocation process. This assumption is reasonable if one thinks that this knowledge can come from the fact that the agents know the repertoire of tasks and this can be learned only once. As for the communication, in the future we intend to relax this assumption and perform tests with unreliable communication channels.

Agents in Swarm-GAP decide which task to execute based on the same mechanism used by social insects. The tendency of the agent i to execute task j is given by its internal threshold θ_{ij}, the tasks stimulus s, and the coefficient of execution x_j of task j which is related with others by an AND constraint, as shows Equation 6. The constant ω is used as a discount rate to decrease the weight of the coefficient of execution on the computation of the tendency.

$$T_{\theta_{ij}}(s) = \frac{s^2}{s^2 + \theta_{ij}^2} + \omega x_j \tag{6}$$

Each task has the same associated stimulus, there is no priority on task allocation. The stimulus s is the same for every task j and its value was empirically determined to maximize the system reward. This was done through the experiments discussed in the next section. The execution coefficient x_j associated to the task j is computed using the rate between the number of allocated tasks and the total number of tasks which are related to task j by means of an and relationship. The terms $|n_j|$ and $|N_j|$ represent these parameters respectively.

$$x_j = \begin{cases} \frac{1+|n_j|}{|N_j|} & se \;\; |n_j| \neq |N_j| \wedge |n_j| > 1 \\ 0 & otherwise \end{cases} \tag{7}$$

Swarm-GAP uses the polymorphism to setup the agents thresholds according to the agents capabilities. Equation 8 set the agents threshold θ_{ij} as 1 minus the capability $capacity_i(j)$ of agent i to perform task j (because Threshold and capability are inversely proportional), given by Equation 8.

$$\theta_{ij} = 1 - capability_i(j) \tag{8}$$

Agents running Swarm-GAP communicate synchronously using a token based protocol. When an agent receives the token, it has the right to determine which task it will execute. Once it is done with the execution, it sends the token to another agent. To complete the allocation, all agents must receive the token and take its decision.

The token message also carriers a set of tuples (i, j) with pairs of agent and task. This message informs the agents about which are the available tasks and which agents received the token yet (to avoid deadlocks). As we can see, the number of changed messages in the algorithm is equal to the number of agents. The size of the token message is proportional to the number of tasks. Figure 1 shows the linear growing in the number of messages changed among the agents to achieve an allocation.

At each time a different agent starts the process, creating the token and sending the first message. The agents selects randomly the next agent to send the token. This is important to allow all agents to make their decisions with different amounts of available tasks.

Figure 2 shows the communication process starting with agent C, which determines the tasks it wants to execute. After, C sends the token to agent A, randomly selected. This process finishes when all agents receive the token.

Algorithm 1 details the Swarm-GAP implementation. The agents start with a unique integer identification to allow the alternating token creation. The agents thresholds are set

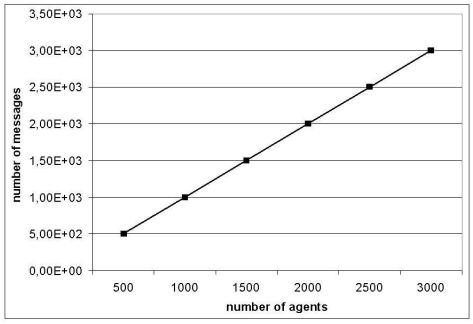

Figure 1: Number of changed messages according to the number of agents

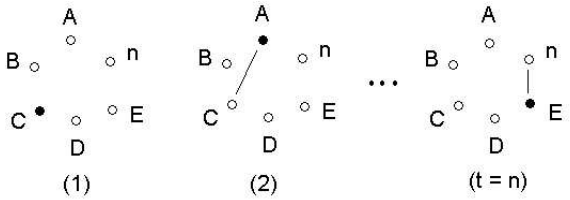

Figure 2: Illustration of agents' communication process

according each agent capability, implementing the polymorphism given by Equation 8.

Each agent controls whether it is its time to initiate an allocation (line 8). If it is the case, the agent creates the token message including all tasks and pointing that no task is allocated yet (line 11). If the agent is not in charge of token creation, it wait until receives a token message (line 14).

Once each agent has received the token message, they all decide whether or not to execute one task (line 18) according to their tendency to execute this task, given by equation 6. This decision also depends on whether the agent has the resource which is required by the task. The token message is modified to contain the information that the agent is executing its selected tasks. The quantity of resource one agent has is decreased by the amount required execution of the task (line 20).

At the and of this process, the token message is sent to an agent randomly selected among those agents which have not received the token in this allocation (lines 23 to 25).

5. EXPERIMENTAL RESULTS

To execute the experiments, a specific simulator was implemented in Java. In each experiment we have 2000 tasks and change the number of agents. The rewards are computed over 1000 rounds, where in each round one allocation is performed. The rewards shown in the graphics are the average over 20 runs of the simulation.

A first experiment was performed to find the stimulus value (Equation 6) that maximizes the reward when the number of agents changes proportionally to the number of tasks. In this case, there is neither a variation in the capabilities required

Algorithm 1 Swarm-GAP(int *agentId*)

```
1: allocCounter ← 0;
2: loop
3:    J ← set of all tasks
4:    r ←
      avaliableResource();
5:    for all j in J do
6:       θ_j ← 1 − capability(j)
7:    end for
8:    if allocCounter % agentId = 0 then
9:       tokenMsg = newTokenMsg()
10:      for all j in J do
11:         tokenMsg.add(j, -1)
12:      end for
13:   else
14:      tokenMsg = receiveMsg()
15:   end if
16:   T ← tokenMsg.getAvaliableTasks()
17:   for all t in T do
18:      if rouletteWeel < T_θt(s) and r <=
      requiredResource(t) then
19:         tokenMsg.set(j, agentId)
20:         r ← r - requiredResource(t)
21:      end if
22:   end for
23:   I ← tokenMsg.getAvaliableAgents()
24:   i ← getRandom(I)
25:   sendTokenMsg(i)
26:   allocCounter++
27: end loop
```

for each task, nor any task related by an and constraint.

Figure 3 shows the achieved reward for different values of stimulus and different number of agents. We use different quantities of agents to experiment different proportions related to the number of tasks (100 means 5% of the 2000 tasks, 500 means 25%, 1000 means 50%, 1500 means 75%, 2000 means 100%, 3000 means 150% e 4000 means 200%).

As we can see, when the number of agents is equal to or greater than the number of tasks, the stimulus that maximizes the reward IS 0.02 (i.e. for 2000, 3000, and 4000 agents). However, when the number of agents decreases,

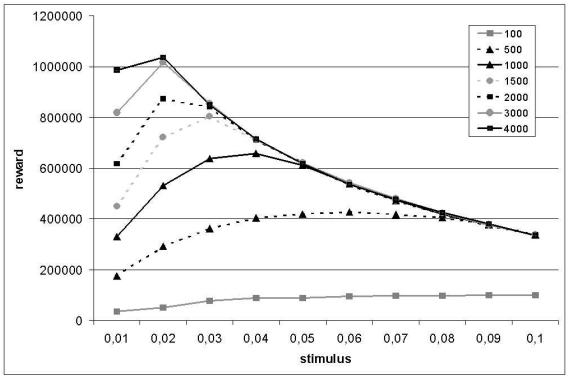

Figure 3: Comparing stimulus and achieved rewards

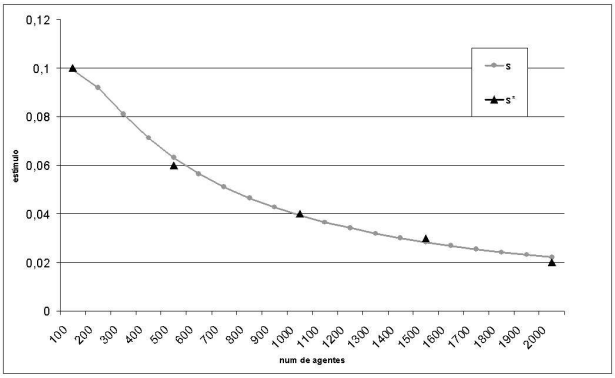

Figure 4: Best stimulus for each number of agents

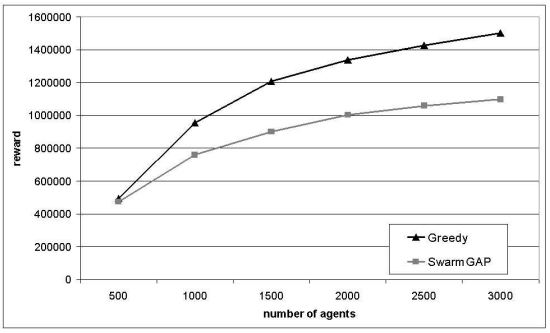

Figure 5: Comparison OF rewards of Swarm-GAP and a Greedy strategy

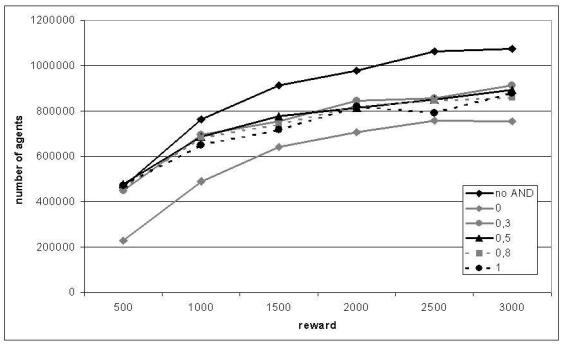

Figure 6: Rewards for different values of ω

the stimulus related to the best reward increase. This experiment shows that, to maximize the rewards, the stimulus must variate according the proportion between the number of agents and the number os tasks.

Figure 4 shows the stimulus values, labelled s^*, that maximizes the rewards for each number of agents. Here we do not use more than 2000 agents because in this cases the stimulus is constant.

We fit the Equation 9 that captures the relationship between the best stimulus values and the number of agents. Figure 4 also shows the curve of this equation, labelled s. As mentioned in Section 4, we adopt this empirically build equation to compute stimulus in Swarm-GAP.

$$s = \frac{1 - e^{\frac{-0.5}{\frac{numAgentes}{10^{int(\log_{10} numTarefas)}}}}}{10} \quad (9)$$

In the second experiment, we evaluate Swarm-GAP comparing its results with the results achieved by a greedy, centralized algorithm. In this experiment we change the task requirements but there is no AND constraints.

The greedy algorithm allocates the best qualified agent (available and having enough resources) to each available task. As expected, the greedy approach outperforms Swarm-GAP. However, Swarm-GAP performs well in the E-GAP scenario achieving rewards 20% only lower (on average) than the greedy ones.

The last experiment measures the impact of computing the rewards, this time considering the AND constraints between tasks. In this experiment, 60% of the tasks are AND constrained in groups of 5.

Figure 6 shows the expected decay in Swarm-GAP performance when we consider the AND constraints and several values for ω (Equation 9). The rewards grow when $\omega \neq 0$, improving the average performance in 25%.

6. RELATED WORK

Swarm based approaches for optimization problems have been presented before in the literature. The Ant Colony Optimization algorithms [6] are successful centralized solutions for several problems on this area [7, 12]. However, there is a lack of studies about distributed versions of optimization problems. Few distributed approaches are reported, manly focused on simple problems.

A distributed approach for manufacturing dynamic scheduling based on social insects model was proposed in [3] and extended by [4]. In this case, wasp like agents 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 in a production line. The machines should choose whether or not to process the job. Their target is to minimize the setup time in order to minimize the total processing time.

To achieve this minimization, each of the machines should specialize in performing one or a few types of jobs. This specialization is achieved by a mechanism inspired by the adaptive task allocation behavior of swarms. Their results show that the social insect model is competitive, or in some cases superior to previously successful agent based systems.

The Swarm-GAP uses basically the same probabilistic swarm task allocation model used in [3, 4]. However, in Swarm-GAP the agents are specialized by default because the information about the agents capabilities is part of the problem definition. Through this capabilities we specialize

the agents using the polymorphism model shown in Equation 6.

Furthermore, we propose to modify the tendency equation to deal with the interrelationships between the tasks, as shown in Equation 6. In [3, 4] the tasks are totaly independent.

The work that introduces the E-GAP ([17]) also presents an approximation algorithm to solve instances of this problem called Low-communication Approximation DCOP (LA-DCOP). An E-GAP can be modelled as a DCOP, which means that algorithms to solve DCOPs can be applied to solve the E-GAP. DCOP algorithms are the state-of-the-art in task allocation under the multiagent community perspective. Several algorithms with different approaches were recently proposed (e.g. ADOPT[14, 13], OptAPO[11] and DPOP[15]). Studies regarding this algorithms performance under complex scenarios show that to search for the optimal solution in this kind of problem is expensive in terms of communication and computational time [10, 8, 5, 1]. Instances of the E-GAP modelled as DCOP result in problems dramatically more complex then the ones cited above.

To deal with large scale teams of agents solving E-GAP is necessary to minimize the communication among the agents as much as possible. Besides, in this kind of problem, it is better to get an approximated solution as fast as possible than to find the optimal in an unfeasible time.

LA-DCOP is a DCOP algorithm developed to deal with this E-GAP special characteristics. E-GAP is solved by LA-DCOP in an approximated fashion. The algorithm dynamically computes a minimal capability threshold for each task, based on the problem specific details (all agents capabilities, task requirements, available resources, etc), and use this to maximize the expected total reward. Agents decide to allocate a task if its capability is grater then this threshold.

LA-DCOP uses a token based protocol to improve communication performance. An agent receives a token, decides which tasks to execute using the threshold communicated in the token message, and send the token to another randomly chosen agent. Additional tokens, called *potential tokens*, are used by the agents to make commitments regarding the allocation of an AND constrained tasks. The authors shown that LA-DCOP outperforms other approximate DCOP algorithm in communication and total reward quality.

Swarm-GAP differs from LA-DCOP in several ways:

- LA-DCOP computes the threshold used by the agents to decide their actions based on global information, including all agents capabilities. Our algorithm uses only local information. Each agent decides which task to allocate based only on its capability and the number of tasks and agents in the system.

- The LA-DCOP threshold is global, which means that one agents computes the threshold, as commented above, and communicates it to the others. The Swarm-GAP threshold is internal to each agent and reflects the agent's capabilities.

- Agents decision in Swarm-GAP is probabilistic according to the swarm's model of division of labor. Even when the probability of executing a task is large, if it is not one, than an agent can decide not to execute it (with small probability). In LA-DCOP, the agent always selects the task when its capability is greater than the threshold.

- Swarm-GAP uses a simple token protocol to allow agents to synchronize. Through this protocol each agent knows the available tasks and communicates its actions to the others. LA-DCOP also uses tokens to improve communication efficiency, but its protocol is more sophisticated. The token messages carries commitments information and the computed threshold, which is used by the agents to decide their actions.

- LA-DCOP deals with AND constrained tasks through communication, increasing the number of token messages (*potential tokens*). Swarm-GAP uses a simple model that modifies the agents' tendency according to the AND constrained tasks allocation.

One can broadly compare the results achieve here with those achieved by LA-DCOP and DSA (approximate algorithm for DCOP)[17], since the experiments were performed with parameters as close as. Figure 5 shows that our approach is equivalent to the DSA, achieving piratically the same rewards. The LA-DCOP statistically outperforms the Swarm-GAP. The greedy method implemented both by us and the one used to evaluate the LA-DCOP did not achieve the same results. This can be explained by unintended differences in the implementations.

7. CONCLUSIONS AND FUTURE WORK

The approach introduced here deals with the Extended Generalized Allocation Problem (E-GAP) based on the theoretical models of division of labor in swarms. The presented algorithm, called Swarm-GAP, solves E-GAP in an approximated and distributed fashion. The E-GAP has been used in the literature to model scenarios of search and rescue in emergency situations, where actors must coordinate their actions to help in disaster management.

Swarm-GAP intends to be a simple and effective algorithm. The experimental results show that the probabilistic decision, based on the tendency and polymorphism models, allows the agents to make reasonable coordinated actions. The Swarm-GAP performs well, achieving rewards, on average, only 20% worse than the ones achieved by a greedy centralized approach. The execution coefficient improves the average reward in 25% in E-GAP instances with several interrelated tasks.

In the future, we intend to compare the performance of Swarm-GAP with DSA and LA-DCOP in a way that runs all algorithms in the same simulation environment with exactly the same setup. this may confirm the present results, achieved by a rough comparison.

Besides, we intend to introduce failures in the simulation regarding the communication channel, agents task perception, and incomplete knowledge about others capabilities.

This failures contribute to a realistic analysis of all algorithms. The idea is to evaluate our intuition that swarm like algorithms are able to deal with failures with simple mechanisms.

8. REFERENCES

[1] S. Ali, S. Koenig, and M. Tambe. Preprocessing techniques for accelerating the dcop algorithm adopt. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 1041–1048, New York, NY, USA, 2005. ACM Press.

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

[3] M. Campos, E. Bonabeau, G. Thraulaz, and J. Deneubourg. Dynamic scheduling and division of labor in social insects. In *Adaptive Behavior*, volume 8–2, pages 83–96, 2001.

[4] 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.

[5] J. Davin and P. J. Modi. Impact of problem centralization in distributed constraint optimization algorithms. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 1057–1063, New York, NY, USA, 2005. ACM Press.

[6] M. Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, 1992.

[7] M. Dorigo, G. Di Caro, and L. Gambardella. Ant colony optimization: A new meta-heuristic. In P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, editors, *Proceedings of the Congress on Evolutionary Computation*, volume 2, pages 1470–1477, Mayflower Hotel, Washington D.C., USA, 6–9 1999. IEEE Press.

[8] P. Ferreira Jr. and A. Bazzan. Distributed meeting schedule through cooperative mediation: analysing optapo's performance in a complex scenario. In A. Meisels, editor, *Proceedings of the Sixth International Workshop on Distributed Constraint Reasoning (DCR2005) - Nineteenth International Conference on Artificial Intelligence (IJCAI2005)*, pages 101–113, July 2005.

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

[10] R. T. Maheswaran, M. Tambe, E. Bowring, J. P. Pearce, and P. Varakantham. Taking DCOP to the real world: Efficient complete solutions for distributed multi-event scheduling. In *Third International Joint Conference on Autonomous Agents and Multiagent Systems*, volume 1, pages 310–317, Washington, DC, USA, July 2004. IEEE Computer Society.

[11] R. Mailler and V. Lesser. Solving distributed constraint optimization problems using cooperative mediation. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, 3.*, pages 438–445, New York, 2004. New York, IEEE Computer Society.

[12] D. Merkle, M. Middendorf, and H. Schmeck. Ant colony optimization for resource-constrained project scheduling. *IEEE Transactions on Evolutionary Computation*, 6(4):333–146, 2002.

[13] P. J. Modi, W. Shen, M. Tambe, and M. Yokoo. Adopt: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161:149–180, January 2005.

[14] P. J. Modi, W.-M. Shen, M. Tambe, and M. Yokoo. An asynchronous complete method for distributed constraint optimization. In *Second international joint conference on Autonomous agents and multiagent systems*, pages 161–168, New York, NY, USA, 2003. ACM Press.

[15] A. Petcu and B. Faltings. A scalable method for multiagent constraint optimization. In *IJCAI 05*, Edinburgh, Scotland, Aug 2005.

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

[17] P. Scerri, A. Farinelli, S. Okamoto, and M. Tambe. Allocating tasks in extreme teams. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 727–734, New York, NY, USA, 2005. ACM Press.

[18] D. B. Shmoys and E. Tardos. An approximation algorithm for the generalized assignment problem. *Math. Program.*, 62(3):461–474, 1993.

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