Crew Intelligence Systems for Digital Objects Preservation

Josep Lluis de la Rosa[1], Esteve del Acebo[1], Albert Trias[1], Silvana Aciar[1], and Hugo Quisbert[2]

Abstract - Crew Intelligence Systems is a family of very simple algorithms for different classes of complex optimization problems in static and dynamic environments by means of reactive multi agent systems. Crew Intelligence systems are loosely inspired from the behavior shown by a staff of bartenders when serving drinks to customers in a bar or pub. In this paper we improve them by letting the bartenders also call (shout) for help, and we adapt them to Digital Objects Preservation, where agents explore file systems looking for victims, digitals objects that need change of format, or any other transformation to be preserved. When they find someone they "shout" so that agent mates can hear it. The louder the shout, the most important or urgent the finding. Louder shouts can also refer to closeness. We perform several experiments to show that this system works very scalably, showing that heterogeneous teams of agents outperform homogeneous ones over a wide range of task complexity. Finally, a properly designed combination of heterogeneous agents is more scalable when confronted with uncertain maps of digital objects to be preserved

1. INTRODUCTION

The term Swarm Intelligence, which has garnered much attention, arose in the late 1990s in the Artificial Intelligence, Robotics, Artificial Life, and Distributed Problem Solving communities, inspired by the observation of social insect colonies. A commonly accepted definition of it is: "the property of a system whereby the collective behaviors of (unsophisticated) agents interacting locally with their environment cause coherent functional global patterns to emerge". The chief paradigm of Swarm Intelligence is an ant colony. In it, individual ant behavior is controlled by a small set of very simple rules, but their interactions (also very simple) with the environment allow them to solve complex problems such as finding the shortest path from one point to another. Ant colonies (and we could say the same about human beings) are intelligent systems with great problem-solving capabilities, formed from a quantity of relatively independent and very simple subsystems that do not show individual intelligence. This is the "many dummies make a genius" phenomenon of emergent intelligence.

Swarm Intelligence problem-solving techniques present several advantages over more traditional ones. On one hand, they are cheap, simple and robust; on the other hand, they provide a basis for exploring collective (or distributed) problem-solving without centralized control or the provision of a global model. Over the last few years, they have been used in the resolution of a very heterogeneous class of problems. Two of the most successful Swarm Intelligence techniques currently in use are Ant Colony Optimization [1] and Particle Swarm Optimization [2]. Ant Colony Optimization techniques, also known as Ant Systems, are based on ants' foraging behavior, and have been applied to problems ranging from the determination of minimal paths in TSP-like problems to network traffic rerouting in busy telecommunications systems. Particle Swarm Optimization techniques, inspired from the way a flock of birds or a school of fish moves, are general global minimization techniques that deal with problems in which a best solution can be represented as a point or surface in an n-dimensional space. Other Swarm Intelligence applications include collective robotics, agent navigation, planetary mapping, streamlining of assembly lines in factories, coordinated agentic transport, and banking data analysis. For more details on self-organization and Swarm Intelligence theory and applications, please refer to [3], [4], [5], [6], [7] and [8].

No doubt, Swarm Intelligence techniques have proved its usefulness over the last years. Nevertheless, in our opinion, their applicability and effectiveness is somewhat limited by the simplicity of the individual agents in the swarm. In the typical Ant Colony Optimization systems, for example, ants behavior is purely reactive and communication between ants is only allowed through the environment, in the form of a pheromone trail. One can't help but wonder whether it would be possible to increase the individual communication and problem solving capabilities of the agents in a Swarm Intelligence system, while at the same time maintaining the desirable features of cheapness, locality, decentralization, simplicity and robustness and what impact would it have in the overall behavior of the system.

It turns out to be that it is possible to find such systems in the real world, especially in those situations where people have to coordinate themselves in a highly dynamic environment in order to solve some kind of scheduling process. Examples are a vessel crew, a staff of bartenders serving pints in a pub or a soccer team. These kind of systems are characterized by highly dynamic environments where tasks of different classes quickly appear and disappear and have to be carried out in a timely fashion. Coordination between people in this kind of systems is very important, but is not attained, typically, by means of some centralized global procedure. The behavior of the individual agents is mainly reactive (they react to the appearance and disappearance of tasks) but, at the same time, and this differentiate them from classical Swarm Intelligence systems,

[1] EASY Innovation Center University of Girona, Campus de Montilivi, E17071 Girona, Catalonia (EU) +34 972 418478, peplluis@eia.udg.edu, acebo@ima.udg.edu, albert.trias@gmail.com, saciar@eia.udg.edu
[2] Luleå University of Technology, University Area, 97187, Luleå (Sweden) +46 920 491810, Hugo.Quisbert@ltu.se

they make use of their "human" abilities (complex communication, reasoning, planning…) to coordinate themselves and increase the problem solving effectiveness of the system. We have chosen to Christianize those kinds of empowered Swarm Intelligence Systems with the name of Crew Intelligence Systems, though early versions were named as Bar Systems [9]. Crew Intelligence are reactive multi agent systems whose behavior is loosely inspired in that of a crew of bartenders. Three traits distinguish them:

- They are well suited for finding approximate solutions for large and complex scheduling real-time problems in highly dynamic environments.
- The behaviours of individual agents are directed towards the maximization of a local affinity function. This individual behavior results in the whole system tending to the minimization of a global cost function.
- Individual agents are endowed with more or less complex communication and local planning abilities which increase the problem solving capabilities of the system.

S&A is a type of Crew Intelligence System, reactive multi-agent systems whose behavior is loosely inspired on that of a staff of bartenders, and can be enclosed in the broader class of Swarm Intelligence systems. We will explore its application to Digital Objects Preservation. The preservation of digital objects is a new issue that comes up with the continuous growth and advance of information, that is created and stored digitally Few solutions have been given to solve how this information will be accessed in the future, even within the next decade or so [10]. Even if the information itself survives over time, the hardware and the software to access it may not. As a result "rescue information" is required to ensure ongoing access to digital preservation for as long as it is requires and for whatever. In this paper is presented how digital objects (text files, photography, audio, etc) can be "rescued" for Long-Term Digital Preservation using solutions inspired from rescue robots and swarm intelligence like algorithms.

This paper is organized as follows: section 2 will describe the Digital Objects Preservation problem and the type of algorithms it needs; the concept of Shout and Act as a type of Crew Intelligence System will be presented and formalized in section 3; in sections 4 and 5 we will present the results of preservation experiments, with several teams of agents and varying complexity of preservation grids; and in section 6 we present our conclusions.

2. THE PROBLEMS OF PRESERVING DIGITAL OBJECTS

Hardware and software obsolescence is jeopardizing the availability and accessibility to digitally recorded information. Moreover, large amounts of digital information have been created and this growth continues exponentially making digital preservation complex. Digital Preservation aims to alleviate threats caused by digital obsolescence and the rapid growth of information. [10] Digital Preservation can be approached as a technical issue using Agents. The most important demand from digital preservation is [14]] that the number of digital objects to be preserved in growing exponentially, and we need algorithms and solutions that could offer linear complexity, better if they offered 0 "plain" complexity.

There are many aspects that determine the difficulty of Digital Objects preservation: planning is done in real time, and heterogeneous multiagent systems work in dynamically changing and even hostile environments in which new and complex scenarios constantly appear. Softbots (they will be referred equally as "agents") interact with totally alien cognitive agents, for example, human beings, with the need for monitoring and constant optimization of scarce resources.

We conceive three approaches for the application of agents to digital preservation:

- First, agents work like recommenders. For example agents can work together for more accurate appraisal of the digital objects to be saved. This approach will help users share their points of view and solutions in a Web 3.0 way, as the automation of web 2.0 approaches to digital preservation, providing with easier tools for preservation. An example of this type of agents is Giulia, shown in [14]].
- Second, agents work like rescue robots. Agents look for digital objects, the victims, with expected short life (for example, format problems) and try to save them. This is expected to contribute with scalability to the exponentially growing complexity of digital preservation.
- Third, agents are the digital objects to be preserved. Agents look for surviving and compete for being preserved. The log of formats is kept inside the agent,. who negotiates with web services or other agents for being itself preserved, and earn credits. This is a native application of agents.

In this paper we will develop the second approach.

Well show how Crew Intelligence Systems are ideal to deal with Digital Preservation requirements, as long as agents are specifically designed to support the communication skills that these systems require.

3. DESCRIPTION OF THE ALGORITHM

For the sake of specificity we are going to call Shout and Act (S&A) the specific Crew Intelligence System algorithm Agents are generally exploring the file system, and will shout when they find an item of interest. Should the shouting agent modulate its volume to indicate distance, or do you simply mean that closer agents are "heard" more loudly? Both two cases apply. The volume of the transmitted shout corresponds to the importance of the finding; the volume received also depends on the distance. The more agents in a position, the more likely it will be for a victim to be at that position. Agents will be of several types, each with heterogeneous skills and capabilities, shouting to (calling) other agents when they detect a potential victim. Those agents with more sensing capabilities tend to be more costly and therefore less numerous. They might follow the hints given by the

shouts of inferior types of agents, and shout to summon even more superior types of agents when they themselves detect something.

Let us now describe S&A from the Crew Intelligence Systems point of view [9]: Anyone who has tried to get served a pint in a bar crowded with customers will have had more than enough time to wonder with boredom about the method used by waiters, if there is one, to decide which customer to pay attention to at each time. Sometimes, one may have to wait a long time before being served, even if shouting for the waiter. . Details like the area where the customer is located, his/her sex, whether the waiter knows him/her, and whether the waiter likes the customer's appearance determines to a large extent the way in which orders are served.

Let us examine the situation from the bartenders' point of view: many customers are ordering drinks at once, new ones arrive constantly, and the bartenders have to do all they can to serve them. Of course, they cannot do this in a random way; they have to try to maximize some kind of utility function, which will typically take into account aspects such as average service time, average service cost, and average customer/boss satisfaction. They will have to pay attention, then, to facts such as that some of them can prepare certain drinks more quickly or better than others, that the order in which the drinks are served influences the time or the total cost of serving them, and also that moving from one place to another takes time. All of this must be done without forgetting, on one hand, that the order in which orders take place has to be respected as much as possible and, on the other hand, that they have to try to favor the best customers by giving them preferential attention and keeping them waiting for a shorter time.

The problem is not at all trivial, and we proved it to be NP-hard in [9]. Bartenders have to act in a highly dynamic, asynchronous and time-critical environment, and no obvious greedy strategy (such as serving the best customer first, serving the nearest customer first, or first serving the customer who has arrived first) gives good results. Nevertheless, a staff of good bartenders usually can manage to serve customers in such a way that the vast majority of them are, more or less, satisfied. The way they accomplish the task seems to have little to do with any global planning or explicit coordination mechanisms but, arguably, with trying to maximize, every time they choose a customer to serve, some local utility function that takes into account aspects like the importance of the customer, the cost for the waiter of serving him/her, and the time that he/she has been waiting for service.

3.1 Outline the Behavioral Rules in Spirit of the Algorithm Formal Approach

We must design agents (softbots) with the following properties from soft-agency:

a) *Agents must be autonomous and proactive*, able to perform desired tasks in unstructured environments without continuous human guidance. A high degree of autonomy is particularly desirable because communication delays and interruptions are unavoidable. They must be able to:
   - Gather information about the environment.
   - Move either all or part of themselves throughout its operating environment without human assistance.
   - Avoid situations harmful to digital objects, property, or themselves.
   - Learn or gain new capabilities, including adjusting strategies for accomplishing tasks and adapting to changing environments.

b) *Agents must be self aware or introspective;* this is a soft-agency property that requires an agent to observe itself, optimize its behavior to meet its goals, and communicate its current limitations to others. Some examples of self-awareness in a system are:
   - State diagnosis: credit for the application of web services, available memory, available disk space, etc.
   - Ongoing activities: serving users, appraising of curating objects.
   - Choosing actions under external and internal constraints.
   - Knowledge and lack of knowledge.
   - Purpose, intentions, hopes, fears, likes, dislikes.
   - Mental state, e.g. long term goals, and beliefs (hard agency).

c) *Agents must be social, able to communicate directly and indirectly.* They must be able to negotiate among themselves to surmount unforeseen properties of a master plan.

d) *Agents must be heterogeneous*, preferably with complementary skills, expected to work better together than separately, though working as a team sometimes will not be possible.

e) *Agents must know where they are situated, not in an absolute but rather in a relative, social, way*, such as a grid of neighbors.

Crew Intelligence Systems should divide the work of exploring and rescuing digital objects among several types of agents with different sets of capabilities, perception, and communications skills, and performs social mapping while acting. When a team of agents, each very light and with limited perception capabilities, is tasked to explore for digital objects and believe they have detected someone, they start shouting for help. This is very helpful, indeed. They do not necessarily know where they are and they are not necessarily able to return, so they just shout around, hoping that other agents will arrive to help. This avoids the strict requirements of making plans, and is the origin of the shout and act principle: agents put emphasis on action, with simple reactive behavior.

Thus S&A will be used by agents to move around, without a map, in an unknown environment of unknown complexity, while at the same time performing preservation tasks. In fact, the map is the grid of agents placed throughout the environment, the file system, the intranet and the internet. Therefore, agents must be designed to work together and to be social, autonomous, heterogeneous, self-aware, and numerous

3.2 Definition

The *Shout and Act* system is a quadruple *(E, T, A, F)* where:

1. E is a (physical) environment. The state of the environment at each moment is determined by a set of state variables VE. One of those variables is the time. S is the set of all possible states of the environment E, that is, the set of all the possible simultaneous instantiations of the set of state variables VE.

2. $T = \{t_1, t_2, ..., t_M\}$ is a set of tasks to be accomplished by the agents within the environment E. Each task t_i has associated:
   - $pre(t_i)$. A set of preconditions over VE that determine whether task t_i can be done (*introspection* and *heterogeneity*)
   - $imp(t_i)$. A nonnegative real value that reflects the importance of task t_i. (*proactivity* and *autonomy*)
   - $urg(t_i)$. A function of VE that represents the urgency of task t_i in the current state of the environment E. It will usually be a nondecreasing function of time (*proactivity* and *autonomy*)

3. $A = \{a_1, a_2, ..., a_N\}$ is a set of agents situated in the environment E. Each agent a_i may have different problem-dependent properties (e.g., weight, speed, location, response time, maximum load, perception, and communication skills, all part of the *heterogeneity*). For each agent a_i and each task t_j, $cost(a_i, t_j)$ reflects the cost for agent a_i to execute task t_j in the current state of the environment E. This cost can be divided into two parts: first, the cost for a_i to make the environment fulfill the preconditions of task t_i (this can include the cost of stopping his current task), and then, the cost for a_i to actually execute t_j. If a agent a_i is unable to adapt the environment to the preconditions of task t_j or if it is unable to carry out the task by itself, then $cost(a_i, t_j)$ is defined ∞.

4. $F : S \times A \times T \rightarrow \Re$ is the function reflecting the degree to which agents are "attracted" by tasks. Given a state s of the environment, an agent a_i, and a task t_j, $F(s, a_i, t_j)$ must be defined so that it increases with $imp(t_j)$ and $urg(t_j)$ and decreases with $cost(a_i, t_j)$

In Shout and Act, agents operate concurrently in the environment in an asynchronous manner, eliminating, thus, the typical operation cycles of other Swarm Intelligence systems (Ant Systems, Particle Swarm Optimization Systems, Cellular Automata, etc.).

The crucial step in the algorithm below is the choice of the task that the agent will execute for the next timestep. In its simplest form, this can consist of choosing the task that maximizes the attraction function F. It can also involve some kind of negotiation between agents and even some kind of local planning. One of the tasks will be to *shout* for help. The individual behavior of agents is given by the following algorithm:

Algorithm: Individual agent's behavior

```
1: procedure ShoutAndActAgent
2:   repeat
3:       Find the most attractive free task M
4:       if the agent is doing M OR trying to fulfill pre(M) then
5:          Continue doing it
6:       else
7:          Stop doing the current task, if any
8:          if pre(M) holds then
9:             Start doing M
10:         else
11: Do some action in order to fulfill pre(M)
12: end if
13:      end if
14:  until no tasks left
15: end procedure
```

It is worth stressing that the algorithm allows the agents to respond in real time to changes in the environment like the appearance of new urgent tasks or the temporal impossibility of fulfilling the set of preconditions of a given task.

3.3 Inter-agent Communication

S&A requires simple communicative skills in the agents to attain the coordinated and self-organized behavior typical of Swarm Intelligence Systems. We can identify three main purposes that communication can serve in order to increase S&A problem solving capabilities:

- *Conflict resolution and negotiation*. The way we defined S&A makes unavoidable the occurrence of conflicting situations in which two or more agents choose the same task to carry out. Lack of communication will lead to a waste of resources because of several agents trying to fulfill the preconditions of the same task, even to the extent that only one of them may carry it out. In such situations, it would be convenient to have some kind of negotiation method, which can be as simple as "the first one to see it goes for it". In the case study in section IV, we will discuss more elaborate negotiation strategies.

- *Perception augmentation*. For those agents with limited perception capabilities (we refer to *capability to perceive the tasks* as, for example, whether they have little skill for appraisal of the need of an object to be preserved or not), communication can allow a agent to transmit information to the others about pending tasks they are not aware of, by means of shouts and pheromones by semantic tags. Let us suppose we want to do some kind of exploratory task in a vast file sytem, which, following with the analogy, is a terrain where points of interest must attract and be explored by agents. It would be useful if agents had the ability to share information about the points of interest that they have located during their exploratory activity. In this way, agents would have access to information about the location of points of interest that lie beyond their perceptual capabilities. If agents

have the capability to sense where the shout comes from, the rule is simple: go towards the *shout*.

- *Learning*. The attraction function *F* does not need to be fixed in advance. Agents can learn it through their own activity and their communicative interactions with other agents. For example, a agent can discover that a certain kind of task has a high cost and communicate this fact to other agents. Furthermore, agents can even learn from other agent's ways of carrying out new tasks.

On the other hand, it is worth differentiating between two main classes of inter-agent communicative processes:

- *Direct*. Agents establish direct communication with each other via some channel and following some kind of protocol.
- *Indirect*. Agents communicate with each other through their actions and shouts, causing changes in the environment. In the S&A framework, this can be seen as Agents generating "communicative tasks" that, when carried out by other agents, increase the information they possess (about the environment and the task set). This is the case for Ant Systems, and this is also the case for Crew Intelligence Systems and for S&A.

3.4 Coalitions of Agents

Given N agents, the number of coalitions that can be generated is 2^N. Each coalition may have a value that represents how efficiently it can perform tasks [13]]. This may be due to differences in capabilities or constraints. As a matter of fact, S&A will apply to a coalition as well as a single agent. The important thing for S&A is that there are diverse behaviors and capabilities that emerge for best adapting to *E*.

4. EXPERIMENTS

Fig 1, shows a rescue scenario with a number of digital objects (victims) v_i at unknown positions in an unknown 2D map representation of a file system. We can see a hierarchical file system like a tridimensional normal distribution, where the root of the file system is in the center of the tridimensional normal. There are more than 100,000 files in more than 15,000 directories.

Also we can superpose a grid, with the center square in the normal distribution center. The value of the normal shows us a heuristic of the connections under this square. It also helps to see a matrix like a file system with links the folders to their brothers, having a similar connectivity.

The structure of the file system is projected onto a grid to give intuitive idea of how agents move and locate the victims, the digital objects to preserve, as shown in Fig. 2.

There are several agents of type A and agents of type B. The two types of agents are designed to detect and confirm the locations of victims. Also we can superpose a grid, with the center square in the normal distribution center. The value of the normal shows us a heuristic of the connections under this square. It also helps to see a matrix like a file system with links the folders to their brothers, having a similar connectivity.

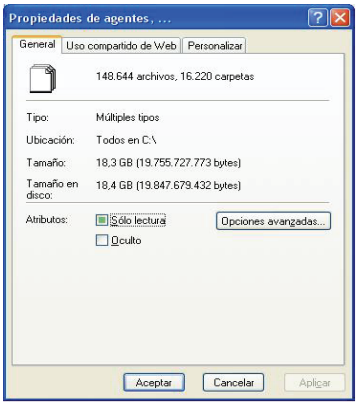

Figure 1. A file system to be made of 2DMap of Victims (objects to be preserved) and Agents

The structure of the file system is projected onto a grid to give intuitive idea of how agents move and locate the victims, the digital objects to preserve, as shown in Fig. 2.

There are several agents of type A and agents of type B. The two types of agents are designed to detect and confirm the locations of victims.

Repast (REcursive Porous Agent Simulation Toolkit http://repast.sourceforge.net/) [12] was chosen for developing the experiments, as shown in Fig 2. It is an open source freely-available agent-based simulation toolkit specifically designed for social science applications. Repast permits the systematic study of complex system behaviors through controlled and replicable computational experiments. It has been released in four versions supporting model development in three different programming languages: (1) RepastJ (Java based); (2) RepastPy (based on the Python scripting language); (3) Repast.Net (implemented in C#, but any .NET language can be used); and (4) RepastS (Repast Simphony, Java based). We use RepastS.

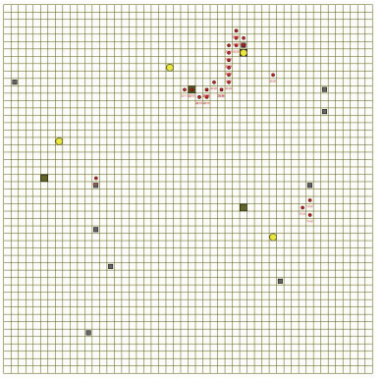

Figure 2. The radar screen in the grid of REPAST-Simphony. Victims are yellow, agents are squares, and shouts are red dots

The environment we designed in Repast Simphony consists of a 2D *radar screen* of 50 x 50 cells within an unlimited world. Agents can get lost by departing from the radar screen. In Repast, agents simulate Victims, Agents, and Shouts.

4.1 Victims

These represent the digital objects to be rescued. They are placed randomly in the grid and are static throughout the simulation. Each Victim agent is given a lifetime, with a clock that decreases with every tick of the simulation until he is rescued or dies. The lifetime of a Victim, which simulates the severity of its injuries (i.e. its format obsolescence), is one of the factors that influences the magnitude of a shout emitted by a agent that detects him, if the agent is able to determine it. Victims are shown in yellow when waiting to be rescued and green after they have been rescued. If they die, they are removed from the grid, as they might become undetectable for our agents.

So the catastrophe is a change of format for MS excel that requires.xls update to .xlsx. This update is the actual preservation action that avoids the contents of the sheet to be inaccessible in the long term.

The four digital objects are:

a) An **excel** containing a budget. High risk of losing information in the format updgrade with wrong calculations if updated.
b) A **word** document containing embedded excel objects. Moderate risk of losing information, because though it perhaps cannot update but can keep visible.
c) A **powerpoint** document containing embedded excel objects. Low risk, because it can keep the information of the excel object visible.
d) A .**rtf excel** with a budget. Very low risk, because the .rtf is designed (!) to be transportable between excel formats.

4.2 Agents

Their goal is to rescue victims or help to do so. There are 2 different types, A and B, which have different capabilities and are initially placed randomly in the grid.

- *A Type* agents are faster and more numerous than B type agents. Their main goal is to detect Victims. In this experiment, they spend most of their time moving randomly in the grid. They can perceive Victims by fast appraisal methods, and they spawn a Shout in that case. If, after a certain period of time, an A Type agent has not been able to perceive any Victim agents, it moves to the closest or strongest Shout agent emitted by another agent.

- *B Type* agents are less numerous and slower than the A Type agents, though they have superior ability to detect, appraise and rescue Victims. They follow the Shouts that A Type agents emit. When a B Type agent hears one or more Shouts, it will set the position of the Shout of highest magnitude as the target endpoint, provided that it can find a trajectory. In order to find out if this condition is met, it carries out some *introspection* and then communicates (if possible) with the other agents of its type. Thus, it determines its state (free / assisting a victim / moving to the position of a shout) and distance to the shout, and asks other Type B agents how close they are from the position where the Shout has been emitted. The next Type B agent does the same reasoning as well, deciding if it should call another agent or act by itself. With this, we achieve some basic coordination between the agents, avoiding the situation where two or more agents of equal type are assisting the same victim. Once a Type B agent reaches the position of a Shout, it performs a scan to locate and confirm the expected Victim.

4.3 Shouts

Shouts are emitted by A Type agents whenever they perceive a Victim agent. Its magnitude could be proportional to the severity of the injuries of the Victim agent. Shouts disappear some time after being emitted, and disperse with distance. The more A Type agents are shouting and the stronger the shout, the more likely it is that other B Type or A Type agents will arrive, consequently increasing the probability of detecting and rescuing a Victim.

Shouts are implemented by tagging the document, in a sort of pheromone, to help other agents detect and start appraising. They also can give pheromones in directories of several higher levels to help attract the mates. The instrument to write down all the shouts is a blackboard [14]].

5. RESULTS

First, a series of ten runs was carried out to test the stability of the experiments. Then, a series of experiments was performed with increasing complexity, different number of victims, and different number of agents of any type. Finally, three scenarios were tried, including new agents with the highest performance.

5.1 Preliminary Experiments

We run S&A 10 times, with random initial grid positions of victims and agents (controlled initial conditions), 4 victims, 10 A type agents, and 4 B type agents. The number of ticks (the discrete measure of time) it takes to rescue all the victims is recorded. We then run the algorithm with 10 B agents (we rename them C type to distinguish them from the first experiment) that move randomly like A Type agents in the same 10 initial conditions, and then compare the results. Fig 3 shows that the homogeneous agents performed slower in all cases; the Y axis represents the number of ticks needed to rescue the victims alive, and the X axis is the number of runs.

As we can see, the performance when using combined A and B type agents is higher than using only C type agents, even though C type agents have the same performance as type B agents and there is a larger number of them (10 against 4). The fact that C agents are not assisted by the faster but unreliable and low-performance A type agents makes them less efficient.

Further experiments show that to achieve similar performance, at least 18 C type agents are needed. Taking into account the fact that A type agents, due to their lower performance (in defective

capacity both to detect victims and to assist them), are assumed to be of much lower cost than B/C agents, combining A and B type agents appears more suitable than increasing the number of C type agents.

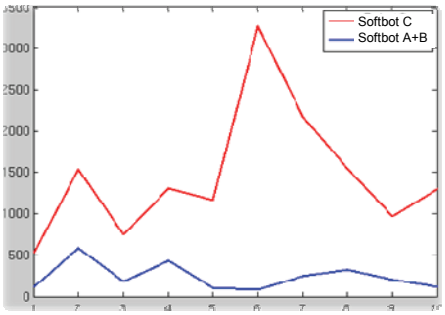

Figure 3. Preliminary experiments. 10 runs with random initial positions of 4 victims, 10 A agents, 4 B agents, and 10 C agents. Y axis is the time to rescue the 4 digital objects.

5.2 Heterogeneous Teams of A+B Agents vs. Homogeneous Teams of C Agents: Scalability Analysis

Let us see now how they behave with increasing complexity, represented as a function of the number of victims to rescue and the number of agents to coordinate. From Table 1, one can tell that with low complexity (1 to 4 victims and 1 to 4 agents), the team of C agents outperforms the heterogeneous A+B agents. On the other hand, the more complex the rescue problem (larger number of victims or agents), the more time a team of homogeneous agents needs, and consequently the worse it performs.

For the sake of simplicity of analysis, we compared the cases at three levels of complexity: low, medium, and high. This shows clearly how the growing complexity (in terms of more victims) causes the solution with C type agents to also increase in time, while the A and B types remain unchanged. In another analysis, increasing the number of agents while increasing the complexity (number of victims) causes A+B agents to rescue digital objects in a much shorter time, even with a small number of A+B agents, a trend much stronger than shown by the C agents.

The main conclusion is: in the case of *uncertain complexity* and an unknown number of available agents, the heterogeneous agents normally outperform the homogeneous agents.

Another interesting result is that 8-10 A and 4 B type agents outperform 14 B type agents. This assertion has a drawback: the B agents really need of A agents to work, so one can claim this comparison is not fair. The following section will address this issue.

Number of Robots of Type A (4 Type B robots)

Number of Victims	14	13	12	11	10	9	8	7	6	5	4
1	300	220	670	235	340	230	270	650	4090	6700	2500
2	155	190	790	890	490	390	1300	100	1500	690	650
3	420	220	120	530	110	840	170	930	1000	500	5070
4	415	170	125	45	270	370	520	1500	320	1600	630
5	120	590	270	395	180	290	130	1270	1050	530	4030
6	150	210	330	140	410	580	310	2200	2100	2100	3050
7	370	195	240	95	90	240	220	700	800	2600	900
8	1150	75	130	175	95	740	310	970	710	3200	1800
9	350	230	95	1450	850	550	110	1200	200	1300	490

High		Medium		Low	
Avrg	315	Avrg	340	Avrg	2522
Dev.	330	Dev.	382	Dev.	2256
Median	230	Median	310	Median	1500

Number of Robots of Type C (no Type A robots)

Numberof Victims	14	13	12	11	10	9	8	7	6	5	4
1	70	840	450	210	800	127	95	157	200	1400	630
2	280	420	1050	350	230	890	450	620	450	2500	3200
3	650	450	1100	180	780	290	560	1490	3290	3000	2670
4	1200	470	1150	530	1020	1160	1970	750	2900	1600	1350
5	780	650	330	510	570	1200	1300	1070	3270	800	5200
6	970	1050	970	1800	1000	1500	2700	2050	1630	950	4900
7	680	1100	1100	950	750	2000	3200	6390	3300	2570	3700
8	1500	1350	1570	1700	790	1400	2400	4320	5300	1300	9300
9	1670	2400	530	4500	1600	2300	1780	2640	3900	4680	8790

High		Medium		Low	
Avrg	1322	Avrg	1380	Avrg	1927
Dev.	561	Dev.	625	Dev.	1257
Median	1350	Median	1200	Median	2500

Table 1. Behavior of Heterogeneous A+B and Homogeneous C type agents with different complexities of the Rescue problem

5.3 Creating Perfect Agents and the Impact on Design

There remains the question: *What if we make a team with the best properties of A and B agents, the D type super-agents?* The answer is that the mixture of A and B agents have the analogous performance as the same number of D agents. Let us examine table 2 for the features of the agents.

Case	N A	N B	N D	V A	V B	V D	P A	P B	P D	Cost (k€)
1: A+B	10	4	0	2	1	-	5	1	-	25,00
2: C	0	14		-	-	1	-	-	1	70,00
3: D	0	0	14	-	-	2	-	-	5	700,00

Type	A	B	D
Features	6	8	9
Cost (k€)	0,50	5,00	50,00

Table 2. Complexity of Case 1 Heterogeneous A+B Agents, and Cases 2 and 3 Homogeneous C and D Agents. N X means number of agents of type X, V X means the speed of the agents of type X, and P X means the perception of the agents of type **X**

	Case	Mean	Std. Deviation
Time (ticks)	1: A+B	531,37	364,87
	2: C	859,27	622,54
	3: D	432,36	274,27
Rescued	1: A+B	9,79	0,37
	2: C	9,63	0,53
	3: D	9,82	0,32

Table 3. Descriptive statistics of the experiments

The experiments are shown in Table 3, 50 runs of every case, through a complexity range of 1-9 victims, identical initial conditions in every case. The result is that the same number of A+B type heterogeneous agents perform similarly to D type homogeneous agents, even though the D type agents combine the best features of types A and B, and consequently are the most expensive. The shorter time to save digital objects implies that more digital objects are rescued.

The conclusion that the heterogeneous agent teams perform reasonably as well as the best team of homogeneous agents will have an important impact on the design of teams of agents, because, increasing the number (by 3) of A agents will perform as well as 14 D agents, that is, slight cost increments of the heterogeneous teams imply big increase of performance.

This result impacts on the design of teams of digital preservation agents: a proper mix of skilled agents with many lower skill agents should work as well though with much lower cost than the best team with only skilled powerful and expensive agents.

6. CONCLUSIONS

Creating new types of agents specifically to work cooperatively will greatly improve the efficiency of preserving digital objects, by following the analogy of rescuing people in unknown environments. This approach copes with the exponentially increasing complexity of the digital objects preservation. The fact that heterogeneous agents outperform homogeneous ones was stated in [11]] and is now shown again using algorithms like Shout and Act (S&A) that take advantage of the finding. Finally some engineering principles should be taken into account in designing Digital Objects Preservation teams, since lower-cost heterogeneous agents can achieve the same efficiency as more costly homogeneous teams.

7. ACKNOWLEDGMENTS

This research was partially funded by the European Project Num. 216746 PReservation Organizations using Tools in AGent Environments (PROTAGE), FP7-2007- ICT Challenge 4: Digital libraries and content, the PR2007-0432 Salvador de Madariaga Grant.

8. REFERENCES

[1] Dorigo, M. and Stützle, T. 2004. Ant Colony Optimization, MIT Press.

[2] Parsopoulos, K.E. and Vrahatis, M.N., 2002. Recent approaches to global optimization problems through particle swarm optimization, Neural Computing, 1 (2-3), 235–306.

[3] Holland, E. O., Beckers, E., R. and Deneubourg J.L., 1994. From Local Actions to Global Tasks: Stigmergy in Collective Robotics, 181–189, Artificial Life IV, MIT Press, Cambridge, MA.

[4] Bonabeau, E., Dorigo, M. and Théraulaz, G. 1999. Swarm Intelligence. From Natura to Artificial Systems, Oxford University Press, 1st edn.

[5] Bonabeau, E. and Théraulaz, G. 2000. 'Swarm smarts', Scientific American, March 2000, 72–79.

[6] Engelbrecht P. A.2006. Fundamentals in Computational Swarm Intelligence, John Wiley and Sons.

[7] Lewis, M. A. and Bekey, G. A. 1992. The Behavioral Self-Organization of Nano agents Using Local Rules, Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Agents and Systems.

[8] Resnick Turtles, M. 1997. Termites and Traffic Jams, Explorations in Massively Parallel Microworlds, MIT Press.

[9] Acebo E., de la Rosa J.L., Bar Systems. 2006. A Class of Optimization Algorithms for Reactive Multi-Agent Systems in Real Time Environments, 17th European Conf. on AI. (ECAI2006) Intl Workshop on New Trends in Real Time AI., pp:128-133, Riva de Garda, Italy, Aug 28-29.

[10] Quisbert, Hugo. 2008. On Long-term Digital Information Preservation Systems – a Framework and Characteristics for Development. Doctoral Thesis. 2008:77. Luleå University of Technology.

[11] de la Rosa J.Ll., Muñoz I. 2007. Learning and Adaptation in Physical Heterogeneous Teams of Agents, VIII Workshop in Physical Agents WAF 2007, pp: 9 – 18.

[12] Tesfatsion L., Iowa State University http://www.econ.iastate.edu/tesfatsi/repastsg.htm

[13] Rahwan, T. , Ramchurn, S. D. , Dang, V. D., Giovannucci, A. and Jennings, N. R. 2007. Anytime optimal coalition structure generation. In Proc. 22nd Conf. on Artificial Intelligence (AAAI), pages 1184–1190.

[14] de la Rosa, J. L., Bengtsson, J., Ruusalepp, R, Hägerfors, A, and Quisbert, H. 2008. Using Agents for Long-Term Digital Reservation the PROTAGE Project, Book Series Advances in Soft Computing Publisher - International Symposium on Distributed Computing and Artificial Intelligence 2008 (DCAI 2008), Vol. 50/2009 pp:118-p:122, ISSN 1615-3871(Print) 1860-0794 (Online), Springer Berlin / Heidelberg.

[15] Dong, J., Chen, S., Jeng, J.J., 2005. Event-Based Blackboard Architecture for Multi-Agent Systems, itcc, pp.379-384, International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume II.