DOI: http://dx.doi.org/10.7551/978-0-262-33027-5-ch080

Towards Failure-Resistant Mobile Distributed Systems Inspired by Swarm Intelligence and Trophallaxis

Arles Rodríguez[1], Jonatan Gómez[1] and Ada Diaconescu[2]

[1]ALIFE Research Group, Universidad Nacional de Colombia
[2]Telecom ParisTech, LTCI CNRS
{aerodriguezp,jgomezpe}@unal.edu.co, ada.diaconescu@telecom-paristech.fr

Abstract

Designing distributed algorithms for mobile ad-hoc sensor systems is difficult, not at least because of their asynchronous communication, mobility, absence of shared memory and high risk of failures. To deal with these challenges, some techniques like replication and consensus are proposed in the literature. However, techniques like consensus depend on a leader election, and this leader can fail. In this paper, we present some advances inspired from nature for the design of a decentralized, scalable way for getting and synchronizing information among components in a point-to-point way. To achieve this, we address the problem of getting and synchronizing information by defining a Distributed System as a swarm of agents (termites), which look for information. Termites are designed with the task of exploring a simulated environment, sensing some desired data distributed throughout the space, and sharing their local knowledge regarding the environment with other nestmates only if they are neighbors. However, when failure rates increase it is less probable than a termite completes the entire task by itself before all termites fail. In order to allow at least one termite to gather the complete information from the environment, several solution approaches are proposed, like sequential exploration with one agent as reference, random movements with local information exchanges, Levy walks and a pheromone-based exploration algorithm inspired by Ant Colony System. This algorithm allows a termite to explore data in a world and enables a termite to search other nestmates with more information by using a trace and by defining a search status given the amount of local information than a termite has. Results show, how swarms manage to collect and replicate information from the entire space even when failures occur. By local interactions, almost all the termites get complete information from a defined world before failing, without a central control and with simple local rules.

Introduction

A Distributed System consists of a collection of components (e.g. processes, agents, robots and nodes) connected via a network, that coordinate their activities and share system resources, offering different services to users and appearing to them as a single system (Tanenbaum and Steen, 2006). Coordination and cooperation between processes are necessary and communication is a key point of Distributed Systems from a design point of view because each process has partial knowledge of the environment, acts in a local way and can fail (Raynal, 2013).

Distributed systems must be scalable. It means being adaptable to changes in terms of size (to provide an easy way to add or delete resources), geographical localization of components and provide easy management independently of its size (Tanenbaum and Steen, 2006). Scalability involves the design and implementation of decentralized algorithms dealing with components that can fail, have no complete information, take decisions in a local way and use point-to-point communication because broadcast is not possible. Lack of scalability usually implies loss of performance of a system while a system grows up (Tanenbaum and Steen, 2006).

Autonomic Computing addresses complexity with the idea of a computer system that adapts to changes without human intervention (Lalanda et al., 2013). Inspired by nature, the idea is that systems elements manage themselves while also providing their services (Kephart and Chess, 2003). The internal behavior of an autonomic element and the set of relationships with other elements are based on goals that a designer has embedded in it and on goals incorporated by other systems through subcontracts with other elements with its tacit or explicit consent (Kephart and Chess, 2003). A set of Self-* properties are required to achieve from the Self-management goal: adapting to the addition or deletion of components (Self-configuration), detecting and recovering from failures without disruption in the system operation (Self-healing), finding improvements in the efficiency of a system (Self-optimization), and anticipating and preventing of threats (Self-protection) (Lalanda et al., 2013; Kephart and Chess, 2003).

On the other hand, replication is proposed as an essential component of failure tolerance in distributed systems. A well known mechanism of replication in Distributed Systems is consensus. By consensus, it is expected that replicas have agreement over a value given local information in each process (Aguilera, 2010). Different areas of application includes state and log machine replication in databases (Aguilera, 2010; Ongaro and Ousterhout, 2013), motion planning, alignment problems (Nedic et al., 2010) and information processing in sensor networks (computing averages of local observations) (Ozdaglar and Nédic, 2007).

Some existing consensus algorithms like Paxos (Lamport, 1998) and Raph (Ongaro and Ousterhout, 2013) propose distributed consensus algorithms based on a leader election process. This leader receives information and replicates it to other processes. However, natural systems do not require a leader directing them and self-organisation relies on collective behaviors that emerge from local interactions as happens in insect colonies where collaboration emerges from Stigmergy (Doursat et al., 2012).

An important challenge to deal with failures is to collect, replicate and synchronize information in a fast way based on the development of adaptable ways of point-to-point communication. In this paper, we propose a solution to the sharing information problem that deals with some of the challenges introduced before. A simulated world is defined with some $data$ of interest distributed in the environment. To get information in this world, we define agents called termites with the task of exploring and obtaining this desired $data$. A termite can move, sense $data$ in its current location and can share its local information with other neighbors. However, termites are unreliable and can crash with a given probability p_f. Given these conditions, the problem is how to get at least one termite to collect whole information of a world before they all crash.

Agents are called "termites" because they follow social organization and their feeding is carried out via *trophallaxis*, meaning that food is stored in their stomach and it is transferred among nestmates through mouth-to-mouth feeding (Rodriguez and Gomez, 2011). In this paper, local information is also inside each agent and local exchange of information is performed between neighbors that look for more information present in other nestmates using stigmergy. *Trophallaxis* is important in nutritional dynamics and communication of many social insects, individual foragers return from a food resource and transfer a portion of their gut material to one or several nestmates. These recipients subsequently become donors to others, and the process continues (Suárez and Thorne, 2000).

This paper uses stigmergy for guiding termites in their search of new information enabling termites to explore a terrain, to get data information from other nestmates and to synchronize local information with others. Our approach inspired by trophallaxis and swarms allows termites to get the whole data before they crash in a decentralized way. A swarm features self-organization that allows it to continue working even if some termites die looking for new information. Inspired by this feature, by adding failures to termites we aim to determine if our proposal can achieve terrain exploration and failure resistance in a simple way inspired by nature; and also establish limits. The remaining of this paper is organized as follows: next section presents a detailed description of the problem, the following section shows some approaches to solve the problem including our proposal of sharing information based on stigmergy and trophallaxis. Finally a result analysis is performed and some conclusions are drawn.

The Problem: World exploration

A world is a bidimensional toroidal space defined as a matrix of properties $Props^{width \times height}$. $Props$ is a collection $Props = \{\tau_w, data\}$, defined with the following values: amount of pheromone in the world $\tau_w : \tau_w \in \mathbb{R} \wedge \tau_w \in [0,1]$ and $data : data \in \mathbb{R} \wedge data \in [0,1]$. $data$ represents some information of interest in this world (e.g. temperatures, altitudes, distance to a determined objective). By now, $data$ values do not change because the main objective is to find a way to get all $data$ information and share it in a fast way. At the beginning all the world positions have a pheromone value of 0.5 and $data$ is generated for each location in a random fashion.

Termites Definition

Multi-agent systems define agents capable of independent actions with the ability of interacting with others. In order to achieve their tasks they are required to cooperate, coordinate and negotiate which each other (Balaji and Srinivasan, 2010). Each termite senses information from the environment and acts by using actuators (Russell and Norvig, 2004). Agents by now are reactive. It means, that each termite operates to respond to changes and to satisfy its design objectives (Russell and Norvig, 2004). Objectives are defined in a termite program which determines the action to be executed by an agent according to its local knowledge. The main thread of each agent looks like algorithm 1. While an agent is alive `(status != Action.DIE)` this agent senses its environment, then chooses an action based on its perceptions and finally the action has an effect on the environment.

```
while (status != Action.DIE) {
    Percept p = environment.sense(this);
    Action action = compute(p);
    environment.act(this, action);
}
```

Algorithm 1: Termite main program

In this paper, each termite is designed with the objective of getting $data$ information in a simulated world of size $width \times height$. $data$ values are represented as a matrix of continuous values $\mathbb{R}^{width \times height}$. The idea is that termites cooperate and coordinate among them to get the global $data$ information starting from local perceptions. The main idea is that each agent looks for and senses new $data$ locally and shares its collected $data^{width \times height}$ at the same time.

Different perceptions have been defined for termites $Percept = \{pheromone, data, socialStatus, neighbor, msg, loc\}$. $pheromone$ is a vector $\mathbb{R}^n$ with values in $[0, 1]$ representing the amount of pheromone that a termite has in its vicinity (Moore neighborhood $r = 1$ with center in the termite location (Gray, 2002)); $data$ is the information in the current location of the termite, $socialStatus = \{$SEEKER, CARRIER$\}$ indicates the status of a termite, $neighbor$ returns the id of a nestmate randomly selected from its Moore neighborhood if one exists, msg stores new messages received from other nestmates and loc returns the current termite location as a matrix position $(row, column)$.

A termite has the following actions $Actions = \{none, down, left, right, up, upleft, upright, downright, downleft, Die, Collect, Send, Receive\}$. First nine actions make reference to movements in the world and Die stops an agent's thread to simulate failure. $Collect$ is performed in each round and means to store the $data$ collected in the local memory of termite. A termite s sends its $data$ collected I_s to a termite r in a msg encoded as a collection $msg = [I_s]$ in a process defined as $Send(r, msg)$. Sending this information is inspired by traditional Asynchronous Distributed Systems where there are FIFO communication channels (Messages received first are processed first by each agent (Raynal, 2013)), local communication (each agent receives and sends messages only to its neighbors), and there is no timing assumptions regarding message delay, clock drift or time taken to send a message. The control returns back to the invoking process after the data is copied in the buffer of the process that receives (Chandra and Toueg, 1996; Kshemkalyani and Singhal, 2008). To implement this mechanism there is a FIFO queue mailbox in the world for each neighbor and an internal queue in each agent that loads new data using the msg perception. In an analogous way, each time a termite r receives new information from a neighbor $Recv(msg)$, it decodes the received message $msg = [I_s]$ and takes decisions based on this information depending on the solution approach.

Trophallaxis inspires the information interchange of this approach. When two termites are occupying adjacent locations in the world, they exchange information. In this way, and just like in nature, termites are donors to other neighbors in a cascade scheme called *Trophallactic cascade*. This pattern of transfer may prove to be more efficient and result in more equitable distribution than direct transfer in nature (Suárez and Thorne, 2000). Communication is added in a way that each time a termite s senses a neighbor r, it gets its id, and exchanges its local $data$ information with r by using $Send$ and $Receive$ in the following way:

- Termite s performs $Send(r, msg)$ where $msg = \{I_s\}$. In this way a termite s sends its current information I_s to r.
- Termite r receives $Recv(msg)$ and completes its information $I_r = I_r \cup I_s$.

Failure Definition

One known type of Distributed System failure is a crash. That is, a process of a Distributed System halts but is working fine until it halts. When a process fails nothing is heard from it. This failure can be identified because the node stops sending messages and does not report a failure (Tanenbaum and Steen, 2006; Satzger, 2008). In the proposed model, a crash is equivalent to the death of a termite. This kind of failure is implemented by defining a probability of failure (p_f) for the agents, see Algorithm 2. For example, $p_f = 0.1$ means that a process has a probability of failure in 1 of 10 rounds.

```
//process can fail with a probability of 0.1
double probFailure = 0.1;
Percept p;
Action actions;

while (status != Action.DIE) {
    if (Math.random() < probFailure) {
           status = Action.DIE;
    }

    p = environment.sense(this);
    actions = compute(p);
    environment.act(this, actions);
}
```

Algorithm 2: Crash model for Agent Program of termite i

Solution Approaches

In this section, some approaches are proposed to solve the aforementioned problem. First, a sequential solution is defined given one termite. After that, more termites that communicate among themselves are added into the environment featuring a random strategy of exploration. Finally a solution based on stigmergy is proposed. For experiments, a fixed size of the environment is defined ($width = 50$, $height = 50$) and several values of p_f are defined. The idea is to estimate the amount of information that agents can get from the environment.

Sequential Exploration with One Termite

Sequential exploration with one termite is modeled as point of reference given the definition of the world as a matrix. Sequential exploration is a good strategy to explore the world because it implies exploring each location in the world only one time. The termite i knows its current location as $loc_i = (x, y)$ where x corresponds to $rows$ and y corresponds to $columns$. Given this location a simple movement program of the termite is the depicted in Algorithm 3 and Fig 1.

One termite is set to get $data$ in the world of $Props^{50\times50}$ (size of the world equal to 2500). The termite is located in

```
while (status != Action.DIE) {
        ...
        if (x != width-1) {
                action = right;
        }else{
                action = downright;
        }
        ...
}
```

Algorithm 3: Sequential exploration for termite i

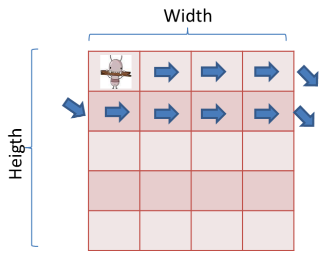

Figure 1: Sequential Exploration Algorithm.

a random location and the world is explored in a sequential way. Given this size of the world, a maximum of 2500 movements is enough for completing information. Sequential exploration experiments were performed 30 times for several values of p_f and results were averaged with the aim of determining how much information a termite can collect in each case.

Table 1 presents the summary of experiments for sequential exploration. Column *Data Collected* displays the average and standard deviation of data collected over the 30 experiments performed. As expected, a greater value in the p_f parameter implies a lesser probability of exploring the entire world. For instance, for a $p_f \geq 10^{-3}$ exploration of the world was never achieved. For the experiments performed

p_f	Data Collected	Successful Experiments
10^{-1}	6.8 ± 6.17	0/30
10^{-2}	113.27 ± 90.36	0/30
10^{-3}	792.2 ± 560.57	0/30
4×10^{-4}	1532.93 ± 906.9	12/30
10^{-4}	2047 ± 825	22/30
10^{-5}	2496.2 ± 14.85	28/30
10^{-6}	2500 ± 0	30/30
0	2500 ± 0	30/30

Table 1: Summary of experiments for crash sequential exploration ($Population = 1, width = 50, height = 50$. A $p_f = 4 \times 10^{-4}$ is added because $1/(width \times height) = 4 \times 10^{-4}$).

with a $p_f = 4 \times 10^{-4}$ a full exploration of the world was achieved only in 12 of the 30 executions. For $p_f = 10^{-5}$ this was achieved in 28 of 30 experiments. The single termite could explore the entire world in a reliable way only if $p_f \leq 10^{-6}$.

Random Exploration with Communication

In these experiments, more than one termite is added to explore the world in a random fashion starting in random locations in the same environment ($Props^{50\times50}$). Termites move in a random way in the environment but have the same p_f. For the same world, experiments were performed with populations of 10, 30 and 50 termites. A maximum of 3000 iterations were defined to get the complete information and each experiment was performed 30 times. Termites can communicate and exchange information with neighbors as specified in the last section *(Termites Definition)*.

Levy Walk Exploration With Communication

A Levy walk is a movement process in which a particle makes a sequence of movements in the same random direction during a time length that depends of another uniform random variable. Foraging mechanisms present in some animals appears to obey Levy walks (Benhamou, 2007). Levy Walks have been used for solving the networking coverage problem in robots by moving them until find a location with an acceptable number of neighbors and make connections (Beal, 2013). In this paper, we adapt the motion mechanism of (Beal, 2013) in the following way (Alg. 4): $randomDir()$ returns a random direction $dir \in \{down, left, right, up, upleft, upright, downright, downleft\}$, α is a uniform random number that represents the increment rate of $acumulator$, T defines a threshold of accumulator for generate a new direction dir and Δt defines a increase of α in terms of time. Several experiments using Levy walks were performed using $T = 1$ and $\Delta t = 1$ and varying the other parameters in the same way it was done with random exploration.

```
while $status \neq Action.DIE$ do
    $dir \leftarrow randomDir()$;
    $\alpha \leftarrow U[0,1]$ //uniform random number;
    repeat
        $move(dir)$;
        $acumulator \leftarrow acumulator + \alpha \cdot \Delta t$;
    until $accumulator \geq T \vee neighbor\_sensor()$;
end
```

Algorithm 4: Reactive Levy walk for termite i (Beal, 2013)

Pheromone-based Exploration

In this approach, each termite determines its movements by using stigmergy inspired by swarms and the Ant Colony

System algorithm (Dorigo and Gambardella, 1997). As a main difference, in this proposal termites are looking for new information (present in other nestmates) instead of looking for food. In this way, termites will have a status determined by the amount of local information that each one has: `SEEKERS` which are termites that look for others for getting new information and to explore locations with more pheromone, and `CARRIERS` that are termites believing they have more information than others and explore world locations with less pheromone. Pheromone value τ_w defined in an environment is used. At the beginning all world locations have a pheromone value of 0.5.

A termite i chooses a direction dir that corresponds to the application of a biased exploration or an exploitation rule depending of a random variable $q \in [0, 1]$ by applying Eq. 1 (Dorigo and Gambardella, 1997):

$$dir = \begin{cases} \text{exploitation rule} & \text{if } q \leq 0.9 \\ \text{biased exploration} & \text{otherwise} \end{cases} \tag{1}$$

Exploitation rule generates a direction depending of a termite status:

- `SEEKERS`: If a termite is `SEEKER`, this termite will choose the direction with the *maximum* amount of pheromone in its vicinity looking for `CARRIERS`. If all values in the vicinity are the same, a random direction is chosen.

- `CARRIERS`: If a termite is `CARRIER`, this termite will choose the direction with the *minimum* amount of pheromone in its vicinity. If all values in the vicinity are the same, a random direction is chosen.

Biased exploration is a random-proportional rule (Dorigo and Gambardella, 1997) which gives to a termite i a probability of choosing a direction $p_d(x, y)$ depending the amount of pheromone τ_w in its vicinity $neighborhood(i)$ (Eq. 2). $neighborhood(i)$ represent the locations in the Moore neighborhood of i with $r = 1$):

$$p_d(x, y) = \left\{ \frac{\tau_w(x,y)}{\sum_{(k,l)\in \text{neighborhood(i)}} \tau_w(k,l)} \right. \tag{2}$$

Each time a termite i performs a movement, it updates its local amount of pheromone $\tau_t(i)$ (local update rule of Eq 3) and updates the pheromone in this world location $\tau_w(x, y)$ (global update rule of Eq 4):

$$\tau_t(i) = (\tau_t(i) + 0.01 * (0.5 - \tau_t(i))) \tag{3}$$

$$\tau_w(x, y) = \tau_w(x, y) + 0.01 * (\tau_t(i) - \tau_w(x, y)) \tag{4}$$

If a termite i turns into a `SEEKER` or finds new information, its pheromone value is updated to 0 ($\tau_t(i) = 0$). Equation 3 is based on the local update rule of ACS (Dorigo and Gambardella, 1997) and represents a local update rule that makes possible that termite pheromone value increases with the time until a certain point reducing the amount of pheromone of the world (global update rule of equation 4). This influence is reduced with time making pheromone in the world converge to its default value of 0.5. A termite in `SEEKER` state can reach a maximum value of pheromone of 0.5.

If a termite i becomes a `CARRIER` or finds new information, the pheromone of the termite gets a value of 1 ($\tau_t(i) = 1$). Local update rule of Equation 3, produces a decrease of the local amount of pheromone until reaching a minimum value of 0.5. At the same time, a `CARRIER` increases the amount of pheromone in the world (global update rule Eq. 4). This influence is reduced with the time making pheromone in the world converge to its default value (0.5). A termite in a `CARRIER` state can reach a minimum value of pheromone of 0.5.

If one termite has more information than the other termite then it turns into a `CARRIER`; or a `SEEKER` in the other case (Fig 2). In this way and just like in nature, termites are donors to other neighbors in a cascade scheme called *Trophallactic cascade*. Each time that a termite s senses some another neighbor r, s sends its information to r, r merge its information ($I_r = I_r \cup I_s$) and additionally r calculates the difference between information $dif = I_s \setminus I_r$. In this way two following scenarios are possible (Fig 2):

- $dif = \emptyset$ means (in a local way) that r had at least the same information as s, so termite r turns into a `CARRIER` and sets its pheromone value in one $\tau_t(r) = 1$.

- Otherwise (if $dif \neq \emptyset$), r turns into a `SEEKER` and sets its pheromone in zero $\tau_t(r) = 0$.

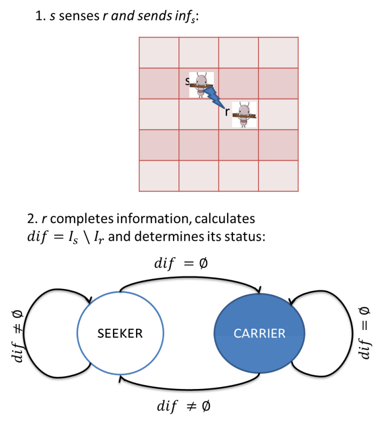
1. s senses r and sends inf_s:
s
r
2. r completes information, calculates $dif = I_s \setminus I_r$ and determines its status:
$dif = \emptyset$
$dif \neq \emptyset$
SEEKER
CARRIER
$dif = \emptyset$
$dif \neq \emptyset$

Figure 2: Communication and Status determination of termite r.

Experiments were performed with 10, 30 and 50 termites. All agents at the beginning are SEEKERS, so they start exploring the world and when they make contact with another nestmate information interchange starts. Figure 3 shows how termites start as SEEKERS (white points) explore the world and turn into CARRIERS (blue points) over time. Red locations in the world represent the variation in the amount of pheromone with the time and how agents explore the world. Squares 3, 4, 5 and 6 of Figure 3 depict cases where communication occurs (green circles).

Fig 4 shows how exploration influences the states of the termites between SEEKERS and CARRIERS. The *Termites* axis represents the individuals in the simulation and the *Iteration* axis represents the average round number. After some iterations all the population becomes CARRIERS for all the experiments performed. Bigger populations makes termites turn into CARRIERS in a fast way.

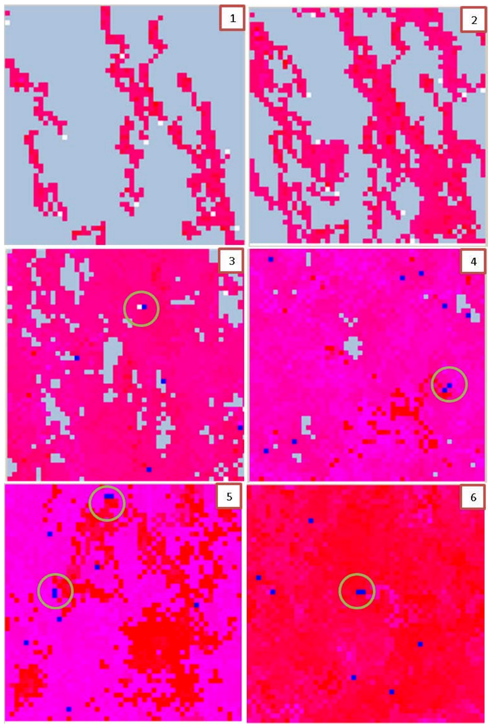

Figure 3: Pheromone exploration with 10 termites, white points are SEEKERS and blue points are CARRIERS.

Results Analysis

Tables 2 and 3 present the results for the experiments of Random Exploration (column Random Expl.). The column *Inf. Col.* shows the average and standard deviation of $data$ collected for agents in the 30 executions. Column *Ag. Compl.* is the average and standard deviation of agents with complete $data$ for the 30 executions of each experiment, Table 3 presents the average round number of the agents that completed information at first place in each experiment.

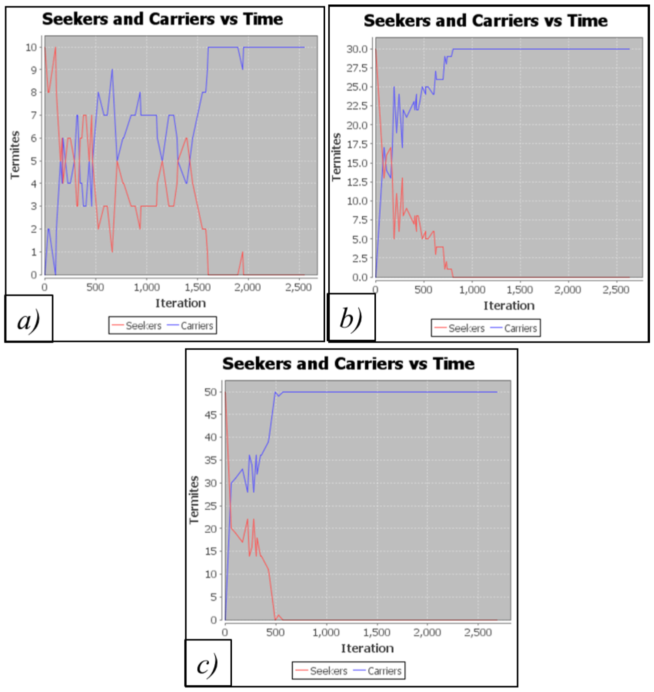

Figure 4: SEEKERS (red line) vs CARRIERS (blue line), $p_f = 0$ $a)$ $Pop = 10$, $b)$ $Pop = 30$, $c)$ $Pop = 50$.

In the experiments performed, a smaller value of p_f means more data obtained and more termites with complete information. However with 10 termites ($Pop = 10$) it was impossible to get the complete $data$ for random exploration. For the given size, random movements and small populations do not warrant exploration of the whole data in the 3000 iterations specified because is difficult for termites to meet and communicate and termites tend to repeat paths. For 30 and 50 termites, for a $p_f \geq 4 \times 10^{-4}$ it is observed more than one agent with all the $data$ information (Agents Complete).

Columns *Levy Walk Expl.* and *Ph. Expl.* of Table 2 presents the results for Levy walks and pheromone exploration respectively. Results show that Levy walks and pheromone exploration work even with 10 termites exploring the world with a $p_f \leq 10^{-4}$ for Levy walks and a $p_f \leq 4 \times 10^{-4}$ in some experiments of pheromone exploration. For 30 and 50 termites the entire information is obtained even with a $p_f = 10^{-3}$. More $data$ is collected as p_f decreases. Bigger populations produce more exploration and a fast information dissemination.

Levy walks are a good technique for exploring new information because the average of information collected is higher compared to pheromone exploration (Table 2). However, the number of agents with complete information before 3000 iterations is bigger for pheromone exploration in 10 and 30 agents and a $pf = 10^{-3}$. In table 3, it is observed

		Random Expl.		Levy Walk Expl.		Ph. Expl.	
Pop	p_f	Inf. Col.	Ag. Compl.	Inf. Col.	Ag. Compl.	Inf. Col	Ag. Compl.
10	10^{-2}	76.68 ± 33.01	0.00 ± 0.00	108.61 ± 52.53	0.00 ± 0.00	122.03 ± 53.08	0.00 ± 0.00
	10^{-3}	897.60 ± 356.59	0.00 ± 0.00	1563.14 ± 335.87	0.00 ± 0.00	1259.91 ± 306.18	0.00 ± 0.00
	4×10^{-4}	1517.67 ± 361.70	0.00 ± 0.00	2073.50 ± 302.45	0.00 ± 0.00	2008.09 ± 314.01	1.57 ± 2.54
	10^{-4}	2268.12 ± 172.21	0.00 ± 0.00	2415.21 ± 114.37	3.70 ± 4.04	2327.36 ± 184.36	7.47 ± 2.50
	10^{-5}	2442.36 ± 75.39	0.00 ± 0.00	2489.92 ± 28.70	6.53 ± 4.46	2489.10 ± 46.04	9.80 ± 0.41
	0	2486.75 ± 6.84	0.00 ± 0.00	2499.66 ± 0.54	7.29 ± 3.98	2500.00 ± 0.00	10.00 ± 0.00
30	10^{-2}	98.69 ± 30.75	0.00 ± 0.00	235.04 ± 79.92	0.00 ± 0.00	174.59 ± 48.76	0.00 ± 0.00
	10^{-3}	1650.54 ± 211.53	0.00 ± 0.00	1991.28 ± 164.40	2.47 ± 4.43	1963.42 ± 190.37	9.03 ± 5.31
	4×10^{-4}	2169.66 ± 132.03	10.07 ± 6.90	2338.12 ± 95.95	20.50 ± 3.15	2218.35 ± 124.45	20.33 ± 2.94
	10^{-4}	2405.94 ± 72.88	25.23 ± 2.53	2451.25 ± 47.00	27.47 ± 1.70	2448.38 ± 51.92	27.83 ± 1.53
	10^{-5}	2491.70 ± 19.91	29.63 ± 0.76	2492.56 ± 22.91	29.67 ± 0.80	2490.70 ± 26.38	29.67 ± 0.61
	0	2500.00 ± 0.00	30.00 ± 0.00	2500.00 ± 0.00	30.00 ± 0.00	2500.00 ± 0.00	30.00 ± 0.00
50	10^{-2}	128.28 ± 30.35	0.00 ± 0.00	428.46 ± 141.93	0.00 ± 0.00	284.50 ± 72.68	0.00 ± 0.00
	10^{-3}	1926.88 ± 156.06	5.90 ± 6.83	2228.73 ± 86.68	22.60 ± 6.42	2051.49 ± 148.74	23.93 ± 4.87
	4×10^{-4}	2230.28 ± 91.13	31.40 ± 4.85	2370.01 ± 57.04	38.07 ± 3.86	2336.00 ± 67.95	40.43 ± 2.80
	10^{-4}	2436.82 ± 41.48	45.17 ± 2.09	2478.72 ± 27.30	47.27 ± 1.87	2459.20 ± 47.88	47.67 ± 1.58
	10^{-5}	2485.10 ± 23.48	49.23 ± 0.90	2496.95 ± 10.79	49.80 ± 0.41	2493.52 ± 16.37	49.63 ± 0.72
	0	2500.00 ± 0.00	50.00 ± 0.00	2500.00 ± 0.00	50.00 ± 0.00	2500.00 ± 0.00	50.00 ± 0.00

Table 2: Summary of average of information collected and number of agents with complete information ($pop = 10, 30, 50$, $width = 50$, $height = 50$, $maxiter = 3000$).

		Average Best Round Number		
Pop	pf	Random Expl.	Levy Walk Expl.	Ph. Expl.
10	10^{-2}	–	–	–
	10^{-3}	–	–	–
	4×10^{-4}	–	–	2162.80 ± 257.22
	10^{-4}	–	2500.81 ± 282.07	1994.39 ± 279.98
	10^{-5}	–	2581.50 ± 219.28	1834.50 ± 224.54
	0	–	2525.78 ± 290.92	1764.35 ± 173.78
30	10^{-2}	–	–	–
	10^{-3}	–	1751.55 ± 516.21	1180.42 ± 332.79
	4×10^{-4}	2038.16 ± 460.05	1074.10 ± 253.87	818.30 ± 90.19
	10^{-4}	1571.40 ± 269.89	925.03 ± 125.36	702.80 ± 54.02
	10^{-5}	1401.70 ± 233.84	871.00 ± 148.31	688.87 ± 48.18
	0	1404.00 ± 231.14	918.57 ± 129.50	691.07 ± 37.52
50	10^{-2}	–	–	–
	10^{-3}	1782.59 ± 570.89	750.70 ± 190.13	626.57 ± 76.69
	4×10^{-4}	1008.00 ± 181.80	584.83 ± 77.98	483.90 ± 35.62
	10^{-4}	874.00 ± 134.37	550.90 ± 133.62	467.87 ± 33.84
	10^{-5}	863.50 ± 144.88	509.03 ± 76.87	458.53 ± 25.03
	0	823.57 ± 96.36	548.77 ± 106.91	451.13 ± 34.85

Table 3: Summary of averages of the number of rounds required for the best agents to collect all information ($pop = 10, 30, 50$, $width = 50$, $height = 50$, $maxiter = 3000$).

that pheromone exploration allows some individuals collect information in a faster way than the other two methods, because the best agents require a small number of rounds in collect all the information.

In the experiments performed, communication is important to reduce the time necessary for getting and disseminating information. Even communication in random exploration tends to reduce the number of rounds necessary for a termite to get all the information from 2500 of sequential exploration with one termite to 823.57 for 50 termites (Table 3). Results with pheromone exploration are even better getting an average number of 451.13 rounds needed for the best termite to collect the complete data without failures ($p_f = 0$).

As expected, bigger populations provide a better performance for exploration and for sharing information in a decentralized, scalable and simple way even with unreliable termites. Additionally, Table 2 shows how Levy walks and pheromone-based exploration imply more resistance to failures compared to random exploration and how a fast data synchronization implies more resistance to failures.

Conclusions and Future Work

In this paper we proposed a solution for obtaining global information via a set of unreliable termites (agents), which explore, get local data and share collected information. The communication mechanism allows neighbors to interchange information, enabling agents to acquire global data as the result of local interactions and cooperation.

It is possible to see how pheromone exploration and Levy walks improve world exploration compared to random exploration. Figure 5 shows how a termite explores the world (a trace is added on visited locations). As expected, pheromone exploration tends to avoid path repetition during exploration vs random exploration where a termite can explore a determined location more than once. Additionally, Levy walks cover an area in a better way than random by maintaining the same direction for some time.

Changes in the status of a termite between `SEEKER` and `CARRIER`, restarts the amount of local pheromone that a termite has, reinforcing the trace and rewarding communication. Future work is intended to test other methods to relate local information with the local update rule of pheromone (e.g. in the way that more local information could represent a stronger trace) and study how a termite's status influences data synchronization and what would happen if perceptions were wrong or if there were changes in the environment and a consensus were required.

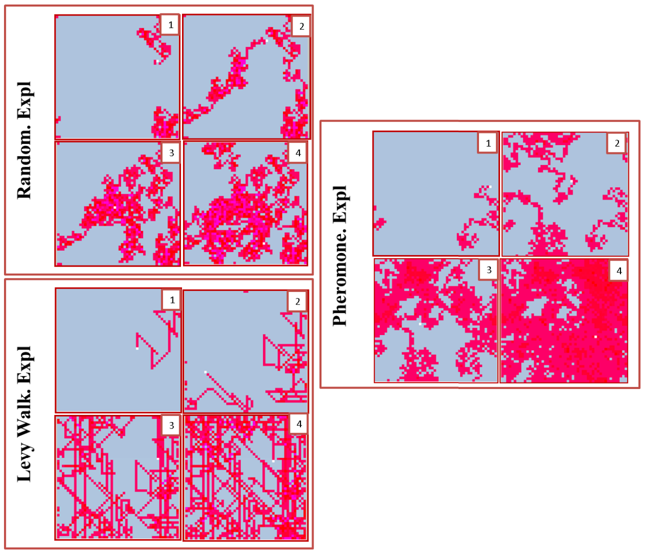

Figure 5: Random, Levy Walk and Pheromone Exploration.

If the same world size is maintained and the population size is increased, a greater amount of information is obtained and a reduction in the average rounds necessary for gathering the whole world information is achieved. Possible next steps include studying the influence of each of these algorithms in terms of number of messages and local information obtained by each termite maintaining the same density of agents and performing experiments with different world sizes.

Acknowledgements

This work was supported by ”Programa de becas para estudiantes sobresalientes de posgrado”, the Alife research group - Universidad Nacional de Colombia, and “Convocatoria del Programa Nacional de Proyectos para el Fortalecimiento de la Investigación, la Creación y la Innovación en Posgrados de La Universidad Nacional De Colombia 2013-2015” code 23418.

References

Aguilera, M. K. (2010). Stumbling over consensus research: Misunderstandings and issues. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 5959 LNCS, pages 59–72.

Balaji, P. G. and Srinivasan, D. (2010). An introduction to multi-agent systems. *Studies in Computational Intelligence*, 310:1–27.

Beal, J. (2013). Superdiffusive dispersion and mixing of swarms with reactive levy walks. *International Conference on Self-Adaptive and Self-Organizing Systems, SASO*, pages 141–148.

Beal, J., Correll, N., Urbina, L., and Bachrach, J. (2009). Behavior modes for randomized robotic coverage. *2009 Second International Conference on Robot Communication and Coordination*.

Benhamou, S. (2007). How Many Animals Really Do the Lévy Walk? *Ecology*, 88(8):1962–1969.

Chandra, T. and Toueg, S. (1996). Unreliable failure detectors for reliable distributed systems. *Journal of the ACM (JACM)*, 43(2).

Dorigo, M. and Gambardella, L. M. (1997). Ant Colony System: A cooperative learning approach to the traveling salesman problem. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*.

Doursat, R., Sayama, H., and Michel, O. (2012). Morphogenetic Engineering: Reconciling Self-Organization and Architecture. pages 1–24.

Gray, L. (2002). at Wolfram s New Kind of Science. *Notices of the AMS*, 50:200–211.

Kephart, J. O. and Chess, D. M. (2003). The vision of autonomic computing. *Computer*, 36(1):41–50.

Kshemkalyani, A. and Singhal, M. (2008). *Distributed computing: principles, algorithms, and systems*. Cambridge University Press.

Lalanda, P., Mccann, J. A., and Diaconescu, A. (2013). *Autonomic Computing: Principles, Design and Implementation*. Springer.

Lamport, L. (1998). The part-time parliament. *ACM Transactions on Computer Systems*, 16(May 1998):133–169.

Nedic, A., Ozdaglar, A., and Parrilo, P. a. (2010). Constrained consensus and optimization in multi-agent networks. *IEEE Transactions on Automatic Control*, 55(4):922–938.

Ongaro, D. and Ousterhout, J. (2013). In Search of an Understandable Consensus Algorithm. *Ramcloud.Stanford.Edu*.

Ozdaglar, A. and Nédic, A. (2007). Consensus Problem in Multi-Agent Systems. (July).

Raynal, M. (2013). *Distributed Algorithms for Message-Passing Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg.

Rodriguez, A. and Gomez, J. (2011). Programs self-healing over a termites simulator based on language games and evolutionary computing. In Tom Lenaerts, Mario Giacobini, H. B. P. B. M. D. and Doursat, R., editors, *Advances in Artificial Life, ECAL 2011 Proceedings of the Eleventh European Conference on the Synthesis and Simulation of Living Systems*, pages 664–671, Paris. MIT Press.

Russell, S. and Norvig, P. (2004). *Inteligencia Artificial. Un enfoque moderno. 2da Edición*.

Satzger, B. (2008). *Self-healing Distributed Systems*. PhD thesis, Augsburg University.

Suárez, M. E. and Thorne, B. L. (2000). Rate, Amount, and Distribution Pattern of Alimentary Fluid Transfer via Trophallaxis in Three Species of Termites (Isoptera: Rhinotermitidae, Termopsidae). *Annals of the Entomological Society of America*, 93(Shapiro 1990):145–155.

Tanenbaum, A. and Steen, M. V. (2006). *Distributed systems: principles and paradigms*. Prentice-Hall.