Competing Ants for Organization Detection
Application to Dynamic Distribution

Alain Cardon[1], Antoine Dutot[2], Frédéric Guinand[2], and Damien Olivier[2]

[1] Laboratoire d'Informatique de Paris VI
Université Paris VI
8, rue du Capitaine Scott
75015 Paris, France
Alain.Cardon@lip6.fr
[2] Laboratoire d'Informatique du Havre
Université du Havre
25 rue Philippe Lebon
76600 Le Havre, France
Antoine.Dutot@univ-lehavre.fr

Summary. A simulation application may be modeled as a set of interacting entities within an environment. Such applications can be represented as a graph with a one-to-one mapping between vertices and entities and between edges and communications. As for classical applications, performances depend directly on a good load balancing of the entities between available computing devices and on the minimization of the impact of the communications between them. However, both objectives are contradictory and good performances may be achieved if and only if a good trade off is found. Our method for finding such a trade off leans on a bio-inspired method. We use competitive colonies of numerical ants, each one depositing colored pheromones, to find organizations of highly communicating entities.

Key words: Complex Systems, Self-organization, Artificial Ants, Dynamic Graph, Community Structures, Dynamic Load Balancing

1 Introduction

We present a new heuristic for the dynamic load balancing problem, when both the application and the computing environment change during the execution. From the computing environment point of view, the generalization of wireless technologies together with the emergence of grid-like and peer-to-peer environments allows us to imagine next HPC (High Performance Computing) environments as collections of heterogeneous computing devices appearing

and disappearing during the computation. Within such a context the availability of dynamic load balancing strategies able to manage both application dynamics and environment changes might be of primary importance .

There exist many application classes whose structural as well as numerical characteristics may change during their execution. Simulation is one of these classes. Simulation covers a wide range of domains: from life and earth sciences to social sciences. Many applications rest on a mathematical analysis and formulation of the global process to be simulated with systems of differential equations. An alternative or a complementary way of modeling problems for simulation purposes consists in modeling the environment, each entity moving and evolving within this environment, and the mechanisms needed for computing entities evolution, their activities and their interactions. Entities may be living beings, particles, or may represent non-material elements (vortices, winds, streams...). Individual-based models fall into this category. Within such models, interactions between the environment and the individuals are explicit as well as interactions between individuals themselves. During the simulation, some individuals may appear and/or disappear without prior notice, and interactions may change, leading to unpredictable sets of highly communicating entities. Due to the dynamics of such applications, the distribution of the entities has to be continuously examined and some migration choices may be proposed in order to improve both performances and emergent organization detections.

In this paper, a bio-inspired algorithm (based on ants) named $AntCO^2$ is described that advises on a possible better location of some entities according to the trade off between load balancing and minimization of communication overhead. The paper is organised as follows: the next section described in detail the considered models of applications and of computing environments. In section 3 the adaptation of the ant approach to our problem is presented, and some experiments are reported in section 4, and we conclude in the last section.

2 Problem Formulation

We are interested in the problem of the allocation and migration of the tasks/-entities/agents/elements of applications on parallel and distributed computing environments. Our method was primarily designed for managing dynamic characteristics of both the application and the execution environment.

2.1 Application Model

Ecological simulations are especially interesting from our point of view since very few is known about the development and the evolution of elements and about their interactions before the execution. Every piece of the simulation needing special processing or interactions is called an entity. Thus, we are in

front of complex systems, in which there are numerous interacting entities. As the system evolves, communications between entities change. Entities and their interactions appear and disappear creating stable or unstable organizations. Many individual-based models (IBMs) are often considered, either as an alternative or as a complement to state variable models [22]. Each entity, in IBMs, is simulated at the level of the individual organism rather than the population level, and the model allows individual variation to be simulated [3]. Thus, during the simulation process some entities may appear or disappear, interactions may meet, leading to an important increase of the number of communications between them (for instance a shoal of sardines passing a group of tuna). While highly dynamic, the whole system may be modeled using a classical non oriented graph whose structural and numerical characteristics may change during the simulation. Within such a graph, the vertex u is associated to an element of the simulation that may be a biological entity or any environmental element, and an edge $e = (u, v)$ materializes a communication or an interaction between two elements of the simulation.

AntCO^2 aims to provide some kind of load-balancing for dynamic communication-based graphs and the simulation is coupled with it, in such way that both applications remain independent up to a certain degree. In one side we have the simulation and in the other side *AntCO*^{2} whereas it goal is to provide a dynamic suggestion on how and where to migrate entities. In other words, *AntCO*^{2} offers a service to the simulation.

2.2 Execution Environment Model

In addition to supercomputers and network of workstations, the last decade have seen the emergence of a new generation of high-performance computing environments made of many computing devices characterized by a high degree of heterogeneity and by frequent temporary slowdowns. These environments - computational grids, peer to peer environments, and networks of mobile computing devices - work almost always in a semi-degraded mode. The sudden failure of a computing device may entail problems about the recovery of elements executed on that device. However, research of solutions for that problem is out of the scope of our work. We assume that fault tolerance during the execution is managed either by the platform used for the implementation or by the application itself. If such a failure occur, we only focus on the reallocation of elements that have lost the computing device they were allocated to.

We also suppose that some events, occurring during the execution, may be automatically recorded and applied to the graph. This can be achieved using functionalities proposed by some programming environments [9]. Relevant events are: arrival and disappearance of elements, communications (source and destination elements, time and/or volume and/or frequency of exchanged data), hardware failures (links, computing devices). An interface is defined between simulation and *AntCO*^{2} (see [15]).

2.3 Previous Works on Dynamic Load Balancing

The problem may be formulated as the on-line allocation, reallocation and migration of tasks to computing resources composing a distributed environment that may be subject to failure. Another way of formulating this problem needs the description of the different communications that may occur between elements. On the one hand, communications between elements located on the same computing resource are supposed negligible. This hypothesis refers to data locality in parallel processing. On the other hand, communications between elements located on distinct computing resources constitute the source of the communication overhead. The latter are called *actual communications* in the sequel. Then, our objective may be expressed as the following trade-off: to reduce the impact of actual communications while preserving a good load-balancing between computing resources.

There exist many variants of this problem which has been extensively studied, and two main classical approaches may be distinguished. The static one consists in computing at compilation time an allocation of the tasks using the knowledge - assumed to be known prior to the execution - about the application and about the computing environment. This problem is known as the mapping problem [6]. A global optimization function, representing both load-balancing and communication overhead minimization, should be minimized. Strategies for achieving this objective often leans on graph partitioning techniques using optimization heuristics like simulated annealing, tabu search or evolutionary computing methods. However this approach is not suitable as-is in our context since we do not have information at compilation time.

The second approach is dynamic load-balancing. Many works have been dedicated to the special case of independent tasks [11, 16, 31, 34]. When communications are considered, the problem is often constrained by some information about the precedence relation between tasks. In [24], for instance, the proposed heuristic, named K1, for scheduling a set of tasks with some dynamic characteristics takes into consideration precedence tasks graphs with inter-tasks communication delays, but, the execution of the task set is supposed to be repeated in successive cycles and the structure of the precedence task graph remains fixed, and only numerical values (execution times and communication delays) may change during the execution. When no information is available about the precedence relation between tasks, Heiss and Schmitz have proposed a decentralized approach based on a physical analogy by considering tasks as particles subject to forces [23]. Their application model is similar to ours, but they do not consider changes that may happen to the execution environment. Moreover, their approach is based on local negotiations between neighbors resources and may spread out, depending of the load.

The key point for achieving a good trade-off lies in the capacity of combining local decisions (communicating tasks forming clusters) with global ones (load balancing). Partitioning of the application graph seems to be a very suitable method for that goal, since clustering may be driven by the objective

of minimizing the communications, and the load balancing may be achieved by fixing the number of clusters. In general, finding an exact solution to k-partitioning problem of this kind is believed to be NP-hard [19]. A wide variety of heuristic has been used to generally provide acceptably good solutions, for example the Kernighan-Lin algorithm [26]. In the last few years, alternatives using natural principles such as simulated annealing, tabu search, genetic algorithms, ant algorithms ([2,25,27,29]) have shown great potential. Partitioning a dynamic application graph seems to be out of reach of traditional clustering approaches. Moreover, when local changes occur in the application graph, it seems more profitable to modify/improve the current solution than to perform a brand new clustering. This implies to consider the global solution as emergent from the set of local solutions rather than as the result of a driven process. All these considerations have led us to consider an artificial collective intelligence approach for our problem.

2.4 Problem Definition

The considered simulations are constituted by a number n at time t of intercommunicating entities, which we wish to distribute over a number of p processing resources. Entities do not necessarily communicate with all the others and their numbers and the communications vary during time. The pattern of communications can be represented by a dynamic graph (network) in which entities are mapped one-to-one with vertices and communications with edges.

Definition 1 (Dynamic graph). *A dynamic graph is a weighted undirected graph* $G(t) = (\mathcal{V}(t), \mathcal{E}(t))$ *such that:*

- $\mathcal{V}(t)$ *is the set of vertices at time* t*.*
- $\mathcal{E}(t)$ *is the set of edges at time* t*. Each edge* e *is characterized by:*
  - *a weight* $w^{(t)}(e) \in \mathbb{N}^+$*.*

The problem is to distribute at anytime in such a way to balance the load on each processing resources and at the same time minimizing the actual communications (see 2.3). To solve this problem, we search a partition of the graph $G(t)$.

Definition 2 (Graph partition). *Let* $G(t) = (\mathcal{V}(t), \mathcal{E}(t))$ *a dynamic graph, a partition* $\mathcal{D}(t)$ *of* $G(t)$ *is composed by* k *disjointed subsets* $\mathcal{D}_i(t)$ *of* $\mathcal{V}(t)$ *called domains with :*

$$k > 0, \bigcup_{i=1..k} \mathcal{D}_i(t) = \mathcal{V}(t)$$

The set of edges connecting the domains of a partition (i.e. edges cut by the partition) $\mathcal{D}(t)$ *is called an edge-cut denoted by* $\mathcal{B}(t)$*.*

Our objective is to find a k-partition, at anytime, which evenly balances the vertex weight among the processing resources whilst minimizing the total weight of $\mathcal{B}(t)$. The number of domains k must be greater or equal to p the number of processing resources (for example the partition found with three colors ($p = 3$ and four domains ($k = 4$) on the graph of Figure 1).

Our method uses organizations inside the communication graph to minimize communications. To each organization corresponds a domain $\mathcal{D}_i(t)$. Detected organizations in the communication graph may, but do not necessarily correspond to entity organizations in the application. These detected organizations appear, evolve and disappear along time.

3 Colored Ant System

Algorithms using numerical ants are a class of meta-heuristics based on a population of agents exhibiting a cooperative behaviour [30]. Ants are social insects which manifest a collective problem-solving ability [12]. They continuously forage their territories to find food [20] visiting paths, creating bridges, constructing nests, etc. This form of self-organization appears from interactions that can be either direct (e.g. mandibular, visual) or indirect. Indirect communications arise from individuals changing the environment and other responding to these changes: this is called *stigmergy*[3]. There are two forms of stigmergy, sematectonic stigmergy produces changes in the physical environment - building the nest for example - and stigmergy based on signal which uses environment as support. Thus, for example, ants perform such indirect communications using chemical signals called *pheromones*. The larger the quantity of pheromones on a path, the larger the number of ants visiting this path. As pheromone evaporates, long paths tend to have less pheromone than short ones, and therefore are less used than others (binary bridge experiment [21]). Such an approach is robust and well supports parameter changes in the problem. Besides, it is intrinsically distributed and scalable. It uses only local information (required for a continuously changing environment), and find nearly optimal solutions. Ant algorithms have been applied successfully to various combinatorial optimization problems like the Travelling Salesman Problem [13], routing in networks [8, 33], clustering [18], coloring [10], graph partitioning [28], and more recently DNA sequencing [4], Dynamic load balancing falls into the category of optimization problems. As reported in [7], Ant Colony Optimisation (ACO) approaches may be applied to a given problem if it is possible to define: the problem representation as a graph, the heuristic desirability of edges, a constraint satisfaction, a pheromone updating rule and, a probabilistic transition rule. Our approach adds to the load balancing

[3] PP. Grassé, "La théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs", in Insectes Sociaux, 6, (1959), p. 41-80, introduced this notion to describe termite building activity.

problem, organization detection and placement of these to reduce network communication. Even if we can find similarities with ACO, we will see in the following that our algorithm, having no satisfaction constraint, depart from this.

Let us first motivate our choice for coloring ants and pheromones.

3.1 Colored Ants

The method proposed by Kuntz et al. [28] for graph partitioning is able to detect clusters within a graph and it also has the ability of gathering vertices such that:

1. if they belong to the same cluster they are gathered in the same place,
2. the number of inter-cluster edges is minimized and,
3. distinct clusters are located in different places in the space.

Points 1. and 2. are relevant for our application, however, additional issues have to be considered:

1. the number of clusters may be different of the number of processing resources,
2. the sum of the sizes of the clusters allocated to each processing resource has to be similar,
3. and their number as well as the graph structure may change.

For the first issue, in [7], the authors mentioned the possibility of setting parameters of the KLS algorithm in order to choose the number of clusters to build. Unfortunately, for our application, we do not know in advance what should be the best number of clusters for achieving the best load balancing, as shown in Figure 1, where elements of the two opposite and not directly linked clusters should be allocated to the same processing resource.

As the number of processing resources available at a given moment is known, this problem may be identified as a competition between processing resources for computing as many elements as possible. This may be directly included in the ant approach by considering competing ant colonies. In our work, a distinct color is associated to each computing resource, and an ant colony is attached to each resource. Each ant is also colored, and drops pheromones of that color.

So, in addition to the classical collaborative nature of ant colonies we have added a competition aspect to the method. In fact, this reflects exactly the trade-off previously discussed about dynamic load-balancing. On the one hand, the collaborative aspect of ants allows the minimization of communications by gathering into clusters elements that communicate a lot, while, on the other hand, the competition aspect of colored ants allows the balancing of the load between computing resources.

The technical issues about the management of ants, colors and pheromones are described in detail in the next sections.

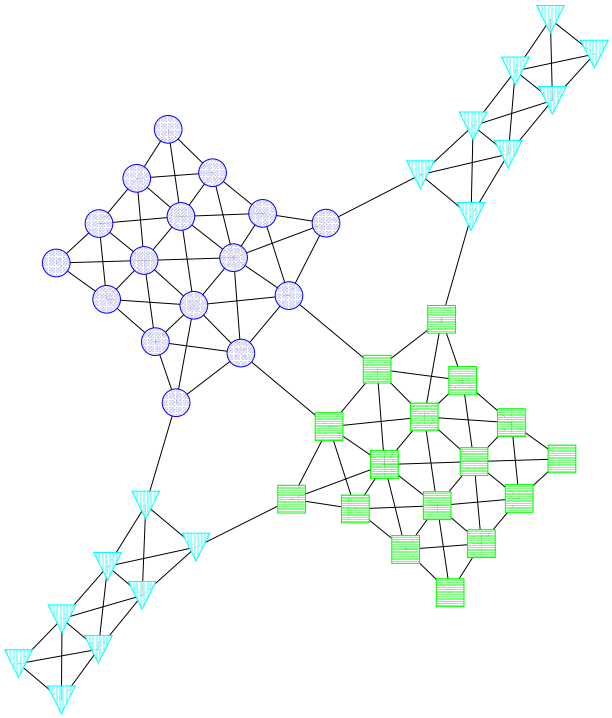


Fig. 1. *Example for which the best number of clusters is different of the number of processing resources (left to right).*

3.2 Graph Description and Notations

As previously mentioned, we consider an undirected graph whose structural as well as numerical characteristics may be subject to changes during the execution. Colored ants walk within this graph, crossing edges and dropping colored pheromones on them.

Definition 3 (Dynamic Communication Colored Graph). *A dynamic colored communication graph is a dynamic graph* $G(t) = (\mathcal{V}(t), \mathcal{E}(t), \mathcal{C}(t))$ *such that:*

- $\mathcal{V}(t)$ *is the set of vertices at time* t*. Each vertex* v *is characterized by:*
  - *a color* $c \in \mathcal{C}(t)$*,*
- $\mathcal{E}(t)$ *is the set of edges at time* t*. Each edge* e *is characterized by:*
  - *a weight* $w^{(t)}(e) \in \mathbb{N}^+$ *that corresponds to the volume and/or the frequency and/or the delay of communications between the elements at each end of edge* e*.*
  - *a quantity of pheromones of each color.*

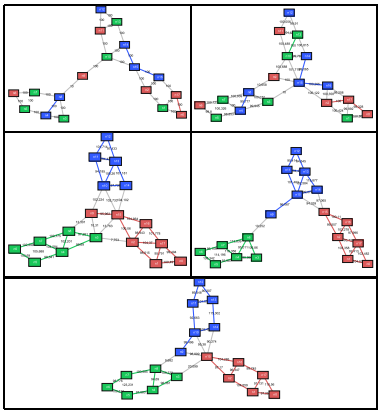


Fig. 2. *Example of a dynamic colored communication graph at five stages of its evolution.*

- $\mathcal{C}(t)$ *is a set of* p *colors where* p *is the number of available processing resources of the distributed system at time* t*.*

The Figure 2 shows an example of a dynamic colored communication graph at several steps of its evolution, where the proposed method described in the following, changes the color of vertices if this improve communications or processing resource load. The algorithm tries to color vertices of highly communicating clusters with the same colors. Therefore a vertex may change color several times, depending on the variations of data exchange between entities.

3.3 Pheromones Management

We denote by $\mathcal{F}(t)$ the population of ants at time t, and $\mathcal{F}_c(t)$ the set of ants of color c at time t. Pheromones are dropped on edges by ants crossing them. Pheromones are colored. An ant x of color c crossing an edge e between steps $t-1$ and t will drop a given quantity of pheromone of color c. This quantity is denoted by $\Delta_x^{(t)}(e,c)$, and the quantity of pheromones of color c dropped by ants when they crossed edge e during time interval $]t-1,t]$ is equal to:

$$\Delta^{(t)}(e,c) = \sum_{x \in \mathcal{F}_c(t)} \Delta_x^{(t)}(e,c) \tag{1}$$

The total quantity of pheromones of all colors dropped by ants on edge e during $]t-1,t]$ is equal to

$$\Delta^{(t)}(e) = \sum_{c \in \mathcal{C}(t)} \Delta^{(t)}(e,c) \tag{2}$$

If $\Delta^{(t)}(e) \neq 0$, the rate of dropped pheromones of color c on e during $]t-1,t]$ is equal to:

$$K_c^{(t)}(e) = \frac{\Delta^{(t)}(e,c)}{\Delta^{(t)}(e)} \text{ with } K_c^{(t)}(e) \in [0,1] \tag{3}$$

The quantity of pheromone of color c on the edge (e) at time t is denoted by $\tau^{(t)}(e,c)$. At the beginning $\tau^{(0)}(e) = 0$ and this value changes according to the following recurrent equation:

$$\tau^{(t)}(e,c) = \rho\tau^{(t-1)}(e,c) + \Delta^{(t)}(e,c) \tag{4}$$

Where $\rho \in]0,1]$ is the proportion of pheromones which has not been removed by the evaporation phenomenon. This denotes the persistence of the pheromones on the edges .

$\tau^{(t)}(e,c)$ may be considered as a reinforcement factor for clustering vertices based on colored paths. However, due to the presence of several colors, this reinforcement factor is corrected according to $K_c^{(t)}(e)$ that represents the relative importance of the considered color with respect to all colors. This corrected reinforcement factor is noted:

$$\Omega^{(t)}(e,c) = K_c^{(t)}(e)\tau^{(t)}(e,c)$$

Then, if we denote by $\mathcal{E}_u(t)$ the set of edges connected to vertex u at time t, the color $\xi^{(t)}(u)$ of vertex u is obtained from the main color of its incident edges:

$$\xi^{(t)}(u) = \arg\max_{c \in \mathcal{C}(t)} \sum_{e \in \mathcal{E}_u(t)} \tau^{(t)}(e,c) \tag{5}$$

3.4 Ants Moving and Population Management

Ants move according to local information present in the graph. Each processing resource is assigned to a color. Each vertex gets its initial color from the processing resource it was allocated to. Initially the number of ants of a given color is proportional to the processing resource power that they represent.

Population Management

The process is iterative, between two steps, each ant crosses one edge and reaches a vertex. When there are too few ants, evaporation makes pheromones disappear and the method behaves as a greedy algorithm. If there are too many ants, pheromones play a predominant role and the system efficiency may decreases. Furthermore, the population is regulated with respect to the number of processing resources and to the number of entities.

Initially, our algorithm creates a fixed number of ants per vertex, which depends on processing resources power. Vertices are, in the same way, proportionally associated to processing resources. Then, during the execution, our

method tries to keep the population density constant in the graph. When some new vertices are created, new ants are created in order to maintain the population density, and some ants are removed from the population when some vertices disappear. When one new processing resource becomes available, the number of colors increases by one. However, the population has to be modified in order to take into account this new color. This is done by changing the color of an equal number of ants of each current color into the new color. Symmetrically, when a processing resource, associated to color c, disappears from the environment, either because of a failure or because this resource is out of reach in case of wireless environments, all ants of color c change their color into the remaining ones. The creation and the removing of edges have no effect on the population.

Ants Moves

The moving decision of one ant located on a vertex u is made according to its color and to the concentration of the corresponding colored pheromones on adjacent edges of u. Let us define $p^{(t)}(e,c)$ the probability for one arbitrary ant of color c, located on the vertex u, to cross edge $e=(u,v)$ during the next time interval $]t,t+1]$. If we denote $w^{(t)}(e)$ the weight associated to this edge at time t, then:

$$\begin{cases} p^{(t)}(e,c) = \dfrac{w^{(0)}(e)}{\sum\limits_{e_i \in \mathcal{E}_u(t)} w^{(0)}(e_i)} & \text{if } t=0 \\ p^{(t)}(e,c) = \dfrac{(\Omega^{(t)}(e,c))^{\alpha}(w^{(t)}(e))^{\beta}}{\sum\limits_{e_i \in \mathcal{E}_u(t)} (\Omega^{(t)}(e_i,c))^{\alpha}(w^{(t)}(e_i))^{\beta}} & \text{if } t \neq 0 \end{cases} \tag{6}$$

The parameters α and β (both >0) allow the weighting of the relative importance of pheromones and respectively weights. However, if ants choices were only driven by this probability formula, there would be no way for avoiding oscillatory moves. So, we introduce a handicap factor in equation (6) $\eta \in]0,1]$ aiming at preventing ants from using previously crossed edges. The idea is very similar to the tabu list. We introduce this constraint directly into the probability formula. Each ant has the ability to remember the k last edges it has crossed. These edges are stored into a list: $\mathcal{W}_x$ with $\text{card}(\mathcal{W}_x) < M$ (M constant threshold). Then, the value of η for an ant x considering an edge (u,v) is equal to:

$$\eta_x(v) = \begin{cases} 1 \text{ if } v \notin \mathcal{W}_x \\ \eta \text{ if } v \in \mathcal{W}_x \end{cases} \tag{7}$$

For the ant x, the probability of choosing edge $e=(u,v_i)$ during time interval $]t,t+1]$ is equal to:

$$p_x^{(t)}(e,c) = \frac{(\Omega^{(t)}(e,c))^{\alpha}(w^{(t)}(e))^{\beta}\eta_x(u)}{\sum_{e_i \in \mathcal{E}_u(t)} (\Omega^{(t)}(e_i,c))^{\alpha}(w^{(t)}(e_i))^{\beta}\eta_x(v_i)} \tag{8}$$

To complete this, we introduce a demographic pressure to avoid vertices that already contain too many ants, so that they are better spread in the graph. It is modeled by another handicap factor $\gamma(v)$ on vertices that have a population greater than a given threshold. This factor is introduced in the formula (8). Given $N(v)$ the ant count on the vertex v and N^* the threshold.

$$\gamma(v) = \begin{cases} 1 & \text{if } N(v) \leq N^* \\ \gamma \in]0,1] & \text{else} \end{cases} \tag{9}$$

The formula (8) becomes :

$$p_x^{(t)}(e,c) = \frac{(\Omega^{(t)}(e,c))^{\alpha}(w^{(t)}(e))^{\beta}\eta_x(u)\gamma(u)}{\sum_{e_i \in \mathcal{E}_u(t)} (\Omega^{(t)}(e_i,c))^{\alpha}(w^{(t)}(e_i))^{\beta}\eta_x(v_i)\gamma(v_i)} \tag{10}$$

Ants Way of Life

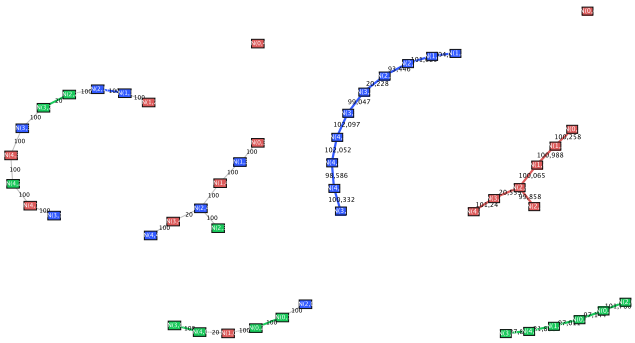
The algorithm is based on an iterative process. During the time interval $]t, t+1]$, each ant may either, hatch, move, or die.

An ant of color c, located on a vertex u dies if the proportion of color c on adjacent edges is under a threshold. If $\phi \in [0,1]$ is the threshold, then, the ant of color c located on u dies if:

$$\frac{\sum_{e \in \mathcal{E}_u(t)} \tau^{(t)}(e,c)}{\sum_{c_i \in \mathcal{C}(t)} \left(\sum_{e \in \mathcal{E}_u(t)} \tau^{(t)}(e,c_i) \right)} < \phi \tag{11}$$

A new ant is then created in another location.

This "jumping" mechanism improves the global algorithm behavior. For instance, when the graph becomes unconnected, some ants may be prisoners of isolated clusters, and the die-and-hatch sequence allows them to escape, as illustrated on Figures 3(a) and 3(b). The mechanism, while keeping population constant, also avoids locked situations (grabs, overpopulation, starvation) (see Fig. 4) occurring when the system meets local minima. Moreover, this improves the reactivity of our algorithm that runs continuously, not to find the best solution for a static graph but, for providing anytime solutions to a continuously changing environment.



(a) *At start.* (b) *Later on, after some ants jumped.*

Fig. 3. *Example of a disconnected dynamic graph.*

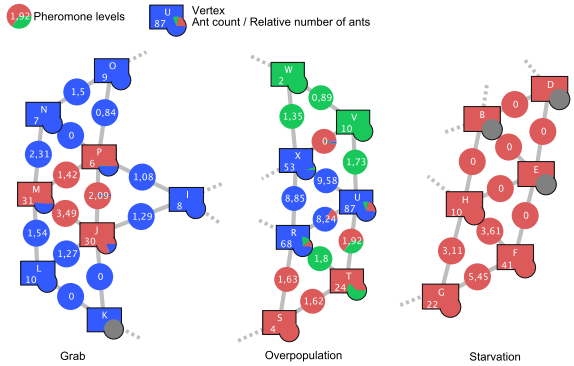


Fig. 4. *Problems solved by jumping mechanism.*

4 Experiments and Results

Dynamic load balancing falls into the category of distributed time-varying problems. It seems difficult to perform a comparison to optimal solutions on dynamic graphs. This is why the first part of the performance analysis is dedicated to the comparison of allocations computed by our method for some classes of static graphs. The second part of the analysis focuses on the reactivity and on the adaptability of our algorithm for some relevant dynamic graphs.

Before entering into details, some performance measures are defined in the next section.

4.1 Quality Analysis

Two measures are relevant for defining the quality of a dynamic load balancing:

- The global costs of communications;
- The load-balancing of the application.

They are antagonist. So, in order to evaluate our solution we first define two quality criteria r_1 and r_2.

The first criterion $r_1 \geq 0$ corresponds to the ratio of actual communications (see section 2.3) over the total number of communications occurring in the graph. Solutions are better when r_1 is close to zero.

$$r_1 = \frac{\text{actual communications}}{\text{all the communications}}$$

The second criterion r_2 measures how the load is balanced among the available processing resources, independently of the communications. For each color c, we have $\mathcal{V}_c(t)$ the set of vertices having color c at time t. Then we have:

$$r_2 = \frac{\min(card(\mathcal{V}_c(t)))}{\max(card(\mathcal{V}_c(t)))}$$

The load-balancing is better when r_2 is close to 1.

In case of static graphs, these criteria, enable us to store the best solutions obtained so far.

Since we seek to find organizations, r_1 is considered to be a more important criterion. Indeed, it is explicitly defined in ant behavior, unlike r_2 that is implicitly optimized by competition mechanisms.

4.2 Static and Dynamic Analysis

The main objective of our approach is not to compute load-balancing for static graphs, however, several tests have been made on different kinds of static graphs to check the validity of our approach. We assume that the computation time required by every element, during one time step, is the same, and we do not take into account migration costs. Each vertex is randomly assigned to an initial processing resource at start. No assumption is made about this initial mapping. For each presented class, figures show both the colored graph solutions, and the evolution of r_1 and r_2 criteria. The abscissa is the number of iterations. For most graphs, parameter used where $\alpha = 1$, $\beta = 3$, $\rho = 0.86$, $N^* = 10$, $\phi = 0.3$, $\eta = 0.0001$, $M = 4$. Changes in these parameters are indicated when needed.

Random Graphs

We tested two kinds of random graphs [17], one is purely random in its topology with uniform weights, and another with the same topology that in addition defines "communications structures". The first graph was created by connecting a given number of vertices randomly with the degree distribution following a Poisson law, the second was made by creating a spanning tree on the first graph, cutting some edges in the tree to create a forest, and giving to each remaining edges of this tree larger communication weights.

Figure 5 shows coloration and criteria evolution for the first graph. Though sometimes nodes with large degree perturb the algorithm, a good distribution is found (r_2 almost always greater than 0.8). The second criterion r_1 is not close to 0 since there are no organizations in the graph.

The Figure 6 uses the same graph topology, but contains organizations of all sizes under the form of communications (Figure 6(a) shows the colored spanning tree with high weights communications only for the random graph of Figure 6(b)). The r_1 criterion is better than for Figure 5 due to the presence of organizations. The slight diminution of r_2 compared to the previous graph comes from the fact organizations introduced by natural clusters do not always define groups of equal size.

Some slight coloration problems appear. They are due to the high number of low communication edges that perturb the algorithm.

Complete Graphs and Graphs with Small Degree Vertices

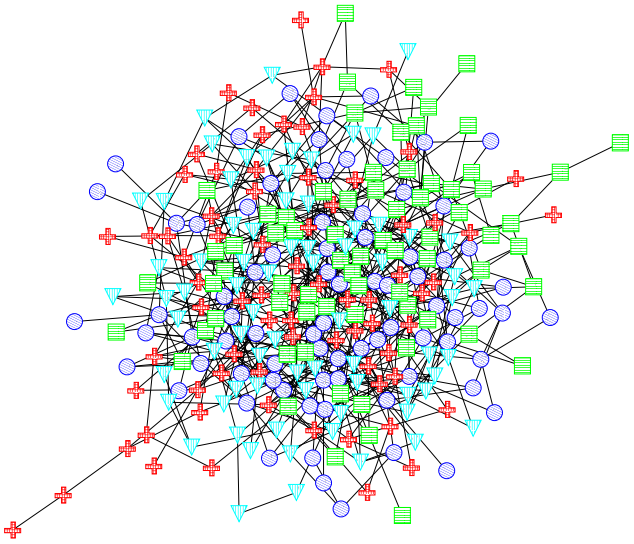
We also consider complete graphs in order to measure how our algorithm balances the load between resources.

The first graph whose r_1 and r_2 evolution is shown on Figure 7, is a complete one with 100 nodes. This graph is unweighted and it therefore does not present organizations. We see that we have a good r_2 criterion, though the algorithm is quite perturbed by the degree of each node. Naturally, r_1 cannot be made small. Indeed the value of r_1 depends on the distribution, and only on it, as all nodes play the same role (same degree and same weight). Therefore it is not really meaningful. As an information on the diagram 7, a dashed blue line indicates the theoretical best r_1 value if all clusters had had the same size (this optimal is crossed by our r_1 value on the graph since all clusters do not have the same size, the plotted "best" r_1 is only informative). This value is:

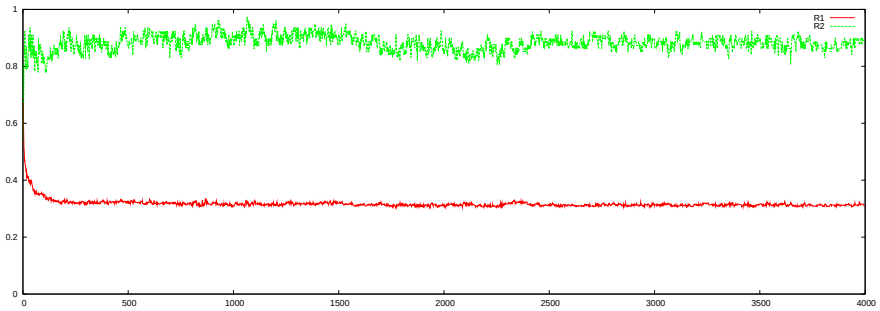
$$\frac{n(p-1)}{p(n-1)}$$

Else, if clusters are not all the same size, r_1 is the optimal and is equal to:

$$\frac{\sum_{j=1}^{p}\left(card(\mathcal{V}_j(t))\left(\sum_{i=1,i\neq j}^{p} card(\mathcal{V}_i(t))\right)\right)}{n(n-1)}$$



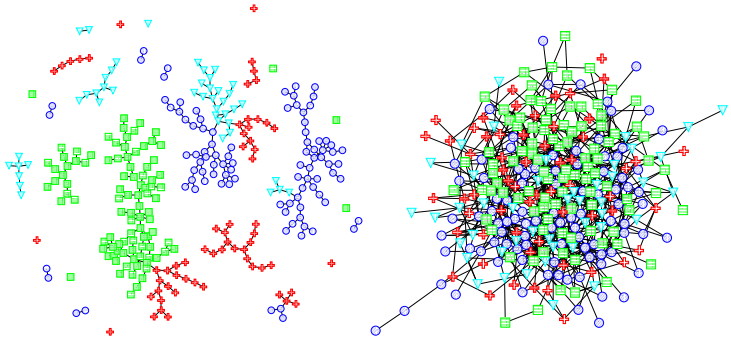
(a) *Colored random graph*



(b) *Evolution of* r_1 *and* r_2

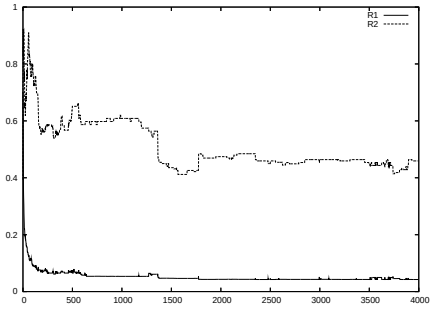
Fig. 5. *A random graph with uniform weights and 4 colors (300 vertices, 602 edges).*

The second graph whose r_1 and r_2 evolution is shown on Figure 8 is almost the same, but we introduced, as for random graphs, a communication structure under the form of weights following a spanning tree in the graph. The graph therefore contains high weights and low weights with a large interval between the two. The r_1 criterion is better than for the first graph. As an indication, the theoretical best r_1 value if all clusters had had the same size is indicated as a dashed blue line. This number is computed as follows. Let $\bar{h}$ and $\bar{l}$ be



(a) *Spanning tree*

(b) *Colored weighted random graph*



(c) *Evolution of r_1 and r_2 on the random graph*

Fig. 6. *A random graph with "communication structures" following a spanning tree, and 4 colors (300 vertices, 602 edges).*

the average high and low weights respectively. Let $\mathcal{E}_h(t)$ and $\mathcal{E}_l(t)$ be the set of edges at time t having high and low weights respectively. Let $\mathcal{A}_h(t)$ and $\mathcal{A}_l(t)$ be the set of actual communication edges at time t having high and low weights respectively. Then the optimal assumes that $\mathcal{A}_l(t)$ has the smallest possible number of elements (for the graph we use, it is zero):

$$\frac{card(\mathcal{A}_h(t))\bar{h} + card(\mathcal{A}_l(t))\bar{l}}{card(\mathcal{E}_h(t))\bar{h} + card(\mathcal{E}_l(t))\bar{l}}$$

Notice that no particular structure has been given to the spanning tree, notably, communication cluster are not all the same size. This impacts on r_2.

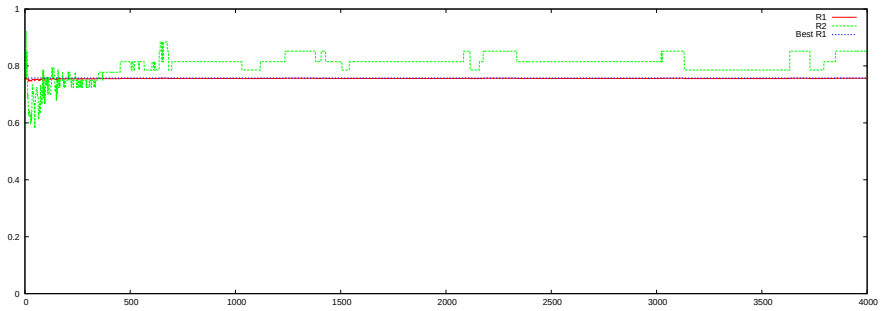


Fig. 7. *A complete graph with uniform weights and 4 colors (100 vertices, 4950 edges).*

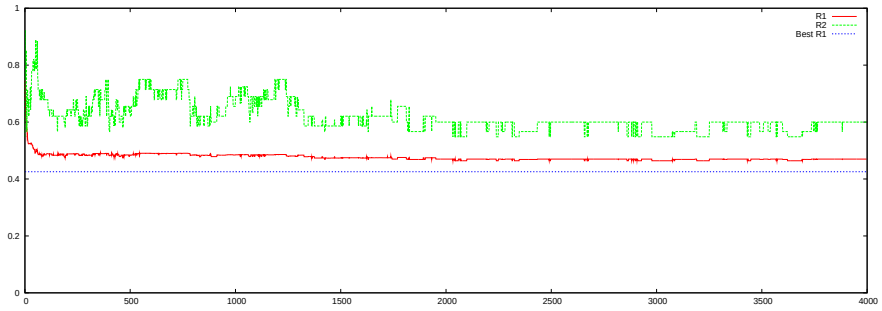
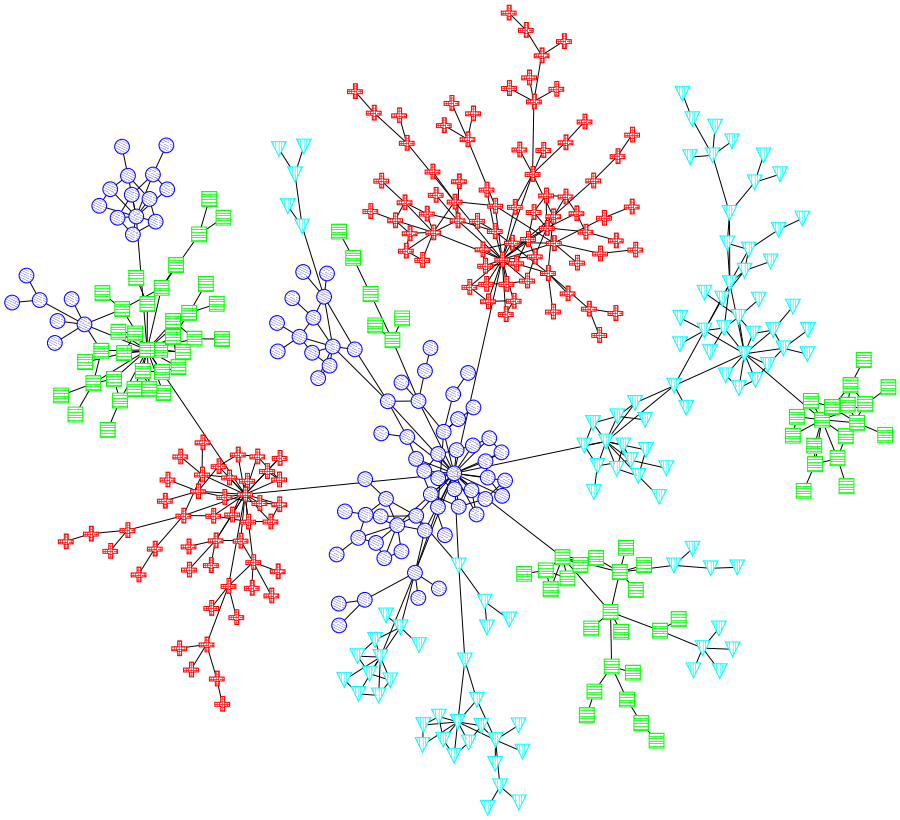


Fig. 8. *A complete graph with "communication structures" following a spanning tree, and 4 colors (100 vertices, 4950 edges).*

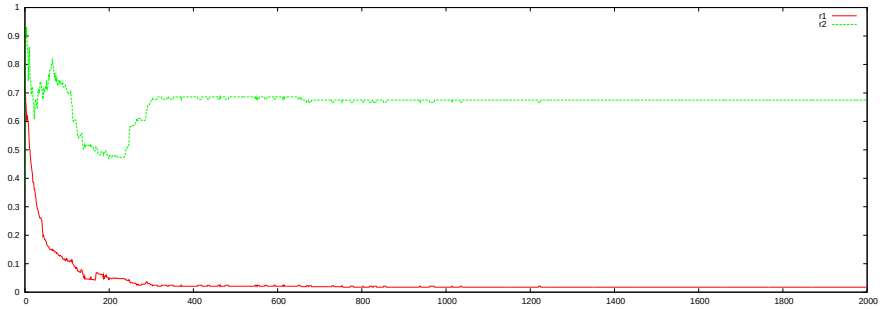
Scale-Free Graphs

These graphs are characterized by the fact that a restrictive number of vertices have many connections with weakly connected vertices. The degree of each vertex follows a power law [1]. These networks are interesting because of their omnipresence in nature. Human interactions in society as well as computer networks or protein interaction networks are some examples of such scale-free graphs [32].

Figure 9 shows the algorithm ability to find organizations of communicating entities. In this example the r_2 value is due to the specific conformation of the graph. The graph forms *natural* clusters which influence how ants travel through the graph: they have far less chances to cross edges between natural clusters except if pushed-of them by competition (as ants tend to favor the r_1 criterion explicitly in their behavior whereas r_2 is only implicitly optimized by the fact several colonies are in competition, as explained above).



(a) *Colored scale-free graph*



(b) *Evolution of* r_1 *and* r_2

Fig. 9. *A weighted scale-free graph with 4 colors (391 vertices, 591 edges).*

Dynamic Graphs

Our algorithm is very well-suited for dynamic graphs processing. The relevance of our method lies in the distributed nature of solutions computations. A change in the input (a change in the dynamic graph, the number of computing resources) occurring at any time during the computation, will entail only local changes, changes taken into account by the method that still continue its task of organizations detection. Indeed, at the base component level, the dynamic graph is constantly being reconfigured. On the contrary, taken as a whole, long lasting organizations appear. They are the image of organizations appearing in the distributed application the graph is a representation of. Such organizations often have a longer lifetime than the average duration of edges or vertices of the graph. Inside these organizations, communication is higher both in terms of volume and connectivity. These two last points are criteria used by ants to form clusters.

Following are several experiments we made with dynamic graphs. For some tests, we used program that simulate the application by creating a graph and then applying events to it. Events are the appearance or disappearance of an edge, a vertex or a processing resource, but also functions that change weights on edges. For others theses we used an ecosystem application where entities have a boid-like behaviour.

In figures 10 and 11, the graph representation is as follows: vertices are rectangles, and edges are shown with a pie chart in the middle that indicates relative levels of pheromones with the maximum pheromone level numbered. Vertices are labeled by their name at the top with underneath at the left the total number of ants they host and at the right a pie chart indicating the relative number of ants of each color present on this vertex.

The first experiment, already shown in Figure 2 and detailed in Figure 10 is a small graph (18 vertices), where three main communication clusters appear. These clusters are linked at the center by low communication edges that appear and disappear. Inside the clusters some edges also appear and disappear. For this experiment we used parameters $\alpha = 1.0$, $\beta = 4.0$, $\rho = 0.8$ $\phi = 0.3$, $\eta = 0.0001$, $N^* = 10$ and $M = 4$ vertices. These parameters will be the same for all other experiments.

The second experiment used a bigger graph (32 vertices) that continuously switch between three configurations. Six snapshots of the graph are presented and show that clusters remains stable across reconfigurations (Figure 11).

The third experiment uses a grid like graph. A small graph travels through the first one. It represents a stable organisation which interacts with the grid. It is continuously connected to the grid but these connection change, so that the small graph moves along the grid (see figure 12).

The small graph keeps the same color along the experiment while it crosses different domains of the grid having a distinct color because communications in the small graph are stronger. Therefore, the organization formed by the

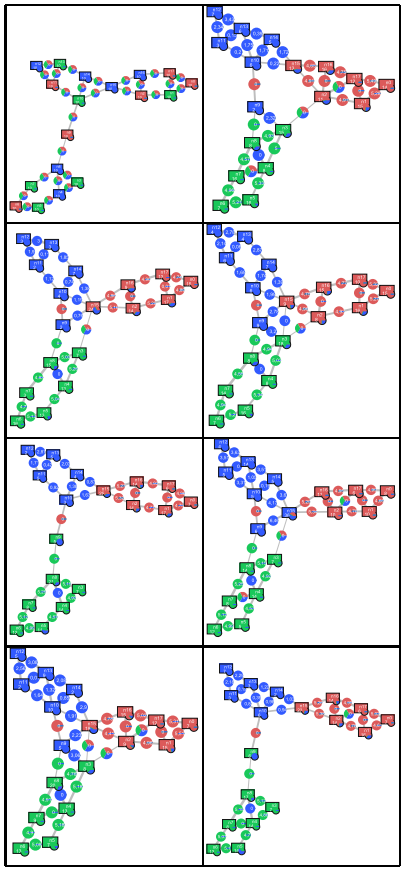


Fig. 10. *Experiment 1.*

small graph is not perturbed by its interactions with the grid. This graph could represent an aquatic simulation application where a fish school passes in an environment (the grid being the environment and the small graph being the fish school). Interactions in the fish school are based on the fish vision area and are more important, and durable, inside the school than with the environment.

The fourth experiment is made using an ecosystem simulation where entities have a boid-like behavior, made of three rules:

- avoidance: they try to stay at a small distance of perceived boids,
- cohesion: they try to fly toward the average position of all perceived boids,
- alignment: they try to match velocity with perceived boids.

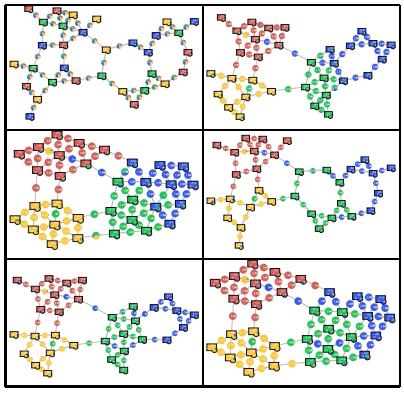


Fig. 11. *Experiment 2.*

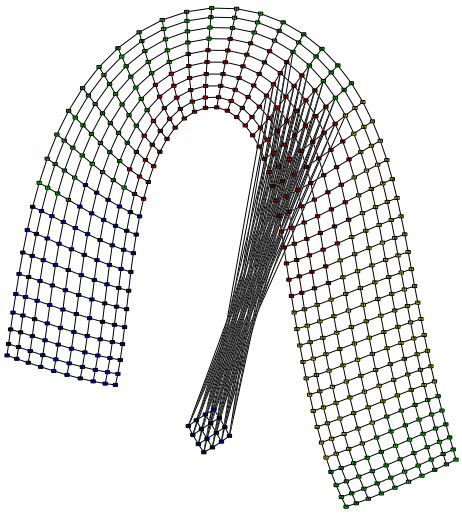


Fig. 12. *Experiment 3.*

These rules create one or several groups of boids. Furthermore, boids try to avoid predators introduced in the simulation. Predators cut boids groups in sub-groups.

Each boid is modeled by a vertex in the dynamic graph. When a boid enters in the field of view of another this creates an interaction (boids reacting to others in their field of view) and therefore an edge in the graph. Figures 15 and 16 show both the boids simulation and the corresponding colored dynamic graph. On Figure 16 boids group formed, and the graph shows the corresponding clusters, detected by AntCO^2 as shown by colors.

We compared $AntCO^2$ with two other distribution strategies: random and grid. The first assigns a color to a boid randomly following an uniform law. As the number of boids stays the same during the experiment, the load balancing is perfect and stays the same. A contrario interactions between boids on distinct computing ressources are much more probable.

In the other strategy, we divide the environment in cells as a grid. Grid cells are mapped to processing ressources. When a boid is in a cell it is running on the processing ressource of this cell. In the same way, interactions between boids which are on two different sub-grids generate actual communications.

Figures 13 and 14 show the r_1 and r_2 criteria evolution respectively for the three strategies on a boid simulation using 200 boids during 5000 steps.

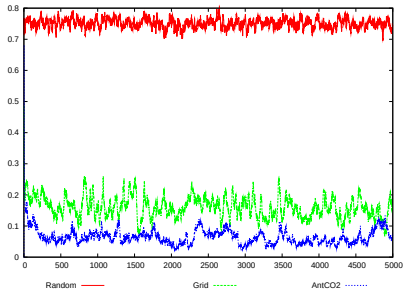


Fig. 13. *R_1 evolution on a* 200 *boids application using three distribution strategies: random, grid and $AntCO^2$.*

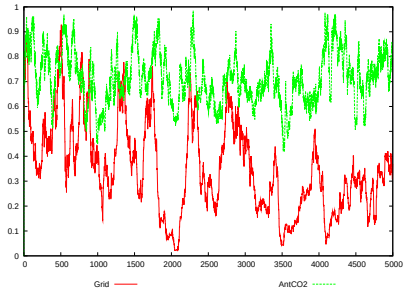
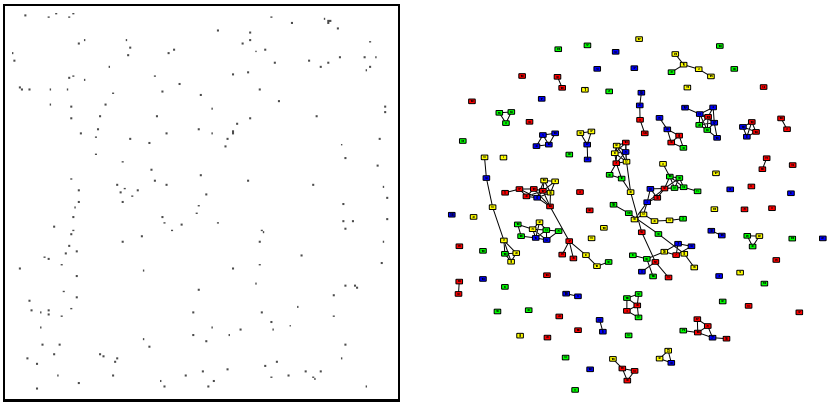


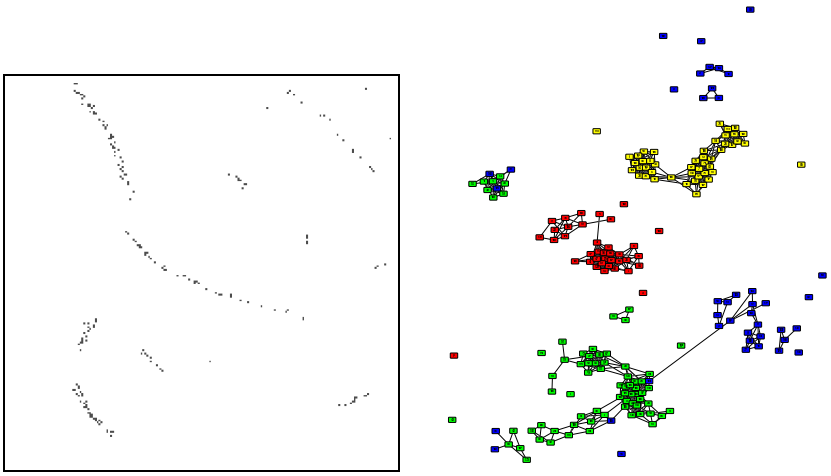
Fig. 14. *R_2 evolution on a* 200 *boids application using two distribution strategies: grid and $AntCO^2$ (random always give $r_2 = 1$).*



(a) Simulation

(b) Colored relation graph between entities of the simulation

Fig. 15. *At start of the simulation.*



(a) Simulation

(b) Graph

Fig. 16. *Later on in the simulation.*

5 Conclusion

In this paper we presented a colored ant algorithm allowing to detect and distribute dynamic organizations. The algorithm offers advices for entity migration in a distributed system taking care of the load and communication balancing. We described a base colored algorithm, observed its behaviour with static and dynamic graphs and provided methods to manage them.

Our algorithm handles dynamic graphs. Two properties of the algorithm allow this: *positive feedback* maintain paths in the graph between highly correlated vertices, *negative feedback* isolate these communities. The first is controlled by ants (pheromone drops), while the other is completely driven by the environment (evaporation, edge deletion, weights). Negative feedback is what makes our algorithm truly adaptive to dynamic graphs, allowing to forget bad communities introduced by the dynamics.

Organizations emerge from the ant behavior which is not explicitly implemented. These organizations correspond to solutions, which is the reason why we don't need an objective function at the contrary of traditional ant systems [14].

Since the algorithm searches for organizations, it favours the r_1 criterion, communication minimisation, above the r_2 criterion, load balancing. The r_1 criterion is explicitly defined in ant behavior whereas r_2 is only implicitly expressed by colored ant competition. This can be seen in our example 9 (scale-free graph), organizations in the graph are sometimes antagonist to an optimal load balancing, merely because organization size is not predictable.

Experimental results show that when there are no organizations (random graphs, complete graphs...), the algorithm indeed finds no artifacts, and favours the implicit r_2 criterion providing good load distributions.

We started development on a heuristic layer allowing to handle some constraints tied to the application, like entities that cannot migrate (e.g. bound to a database), but also information peculiar to the application.

This work takes place within the context of aquatic ecosystem models [5], where we are faced to a very large number of heterogeneous auto-organizing entities, from fluids representatives to living creatures with their specific behaviour.

References

1. R. Albert and A.L. Barabaśi. Statistical mechanics of complex networks. *Reviews of modern physics*, 74:47–97, 2002.
2. R. Banos, C. Gil, J. Ortega, and F.G. Montoya. Multilevel heuristic algorithm for graph partitioning. In G.R. Raidl et al, editor, *Applications of Evolutionary Computing*, volume 2611, pages 143–153. Lecture Notes in Computer Science, Springer, 2003.

3. D. J. Barnes and T. R. Hopkins. The impact of programming paradigms on the efficiency of an individual-based simulation model. *Simulation Modelling Practice and Theory*, 11(7-8):557–569, 2003.
4. C. Bertelle, A. Dutot, F. Guinand, and D. Olivier. Dimants: a distributed multi-castes ant system for DNA sequencing by hybridization. In *NETTAB 2002*, pages 1–7, AAMAS 2002 Conf, Bologna (Italy), July 15th 2002.
5. C. Bertelle, V. Jay, and D. Olivier. Distributed multi-agents systems used for dynamic aquatic simulations. In D.P.F. Müller, editor, *ESS'2000 Congress*, pages 504–508, Hambourg, September 2000.
6. S. H. Bokhari. On the Mapping Problem. *IEEE Transactions on Computers*, 30:207–214, March 1981.
7. E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence - From natural to Artificial Systems*. Oxford University Press, 1999.
8. G. Di Caro and M. Dorigo. Antnet: A mobile agents approach to adaptive routing. Technical report, IRIDIA, Université libre de Bruxelles, Belgium, 1997.
9. D. Caromel, W. Klauser, and J. Vayssiere. Towards seamless computing and metacomputing in java. In Geoffrey C. Fox, editor, *Concurrency Practice and Experience*, volume 10, pages 1043–1061. Wiley & Sons, Ltd., September-November 1998. http://www-sop.inria.fr/oasis/proactive/.
10. D. Costa and A. Hertz. Ant can colour graphs. *Journal of Operation Research Society*, (48):105–128, 1997.
11. S. K. Das, D. J. Harvey, and R. Biswas. Adaptative load-balancing algorihtms using symmetric broadcast networks. *Journal of Parallel and Distributed Computing (JPDC)*, 62:1042–1068, 2002.
12. J.-L. Deneubourg and S. Goss. Collective patterns and decision making. *Ethology Ecology and Evolution*, 1(4):295–311, 1989.
13. M. Dorigo and L.M. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.
14. M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: optimization by a colony of cooperating agents. *IEEE Trans. Systems Man Cybernet.*, 26:29–41, 1996.
15. A. Dutot, R. Fisch, D. Olivier, and Y. Pigné. Dynamic distribution of an entity-based simulation. In *JICCSE 04 - Jordan International Conference on Computer Science and Engineering*, Alt-Salt, Jordan, 4-7 October 2004.
16. D. L. Eager, E. D. Lazowska, and J. Zahorjan. A comparison of receiver-initiated and sender-initiated adaptive load sharing. *Performance evaluation*, 6:53–68, 1986.
17. P. Erdös and A. Rényi. On random graphs. *Pubiones Mathematicaelicat*, 6:290–297, 1959.
18. B. Faieta and E. Lumer. Diversity and adaptation in populations of clustering ants. In *Conference on Simulation of Adaptive Behaviour*, Brighton, 1994.
19. M.R. Garey and D.S. Johnson. *Computers and Intractability, a Guide to the Theory of NP-Completeness*. W.H. Freeman and Compagny, 1979.
20. D.M. Gordon. The expandable network of ant exploration. *Animal Behaviour*, 50:995–1007, 1995.
21. S. Goss, S. Aron, J.-L. Deneubourg, and J. M. Pasteels. Self-organized shortcuts in the argentine ant. *Naturwissenchaften*, 76:579–581, 1989.

22. V. Grimm. Ten years of individual-based modelling in ecology: what have we learned and what could we lear in the future ? *Ecological Modelling*, 115(2-3):129–148, 1999.
23. H.-U. Heiss and M. Schmitz. Decentralized dynamic load balancing: The particles approach. *Information Sciences*, 84:115–128, 1995.
24. Z. Jovanovic and S. Maric. Heuristic algorithm for dynamic task scheduling in highly parallel computing systems. *Future Generation Computer Systems*, 17:721–732, 2001.
25. P. Kadluzczka and K. Wala. Tabu search and genetic algorithms for the generalized graph partitioning problem. *Control and Cybernetics*, 24(4):459–476, 1995.
26. B.W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graph. *The Bell System Technical Journal*, 49(2):192–307, 19970.
27. P. Korošec, J. Šilc, and B. Robič. Solving the mesh-partitioning problem with an ant-colony algorithm. *Parallel Computing, Elsevier*, 30:785–801, 2004.
28. P. Kuntz, P. Layzell, and D. Snyers. A colony of ant-like agents for partitioning in vlsi technology. In *Fourth European Conference on Artificial Life*, pages 417–424, Cambridge, MA:MIT Press, 1997.
29. A. E. Langham and P.W. Grant. Using competing ant colonies to solve k-way partitioning problems with foraging and raiding strategies. In D. Floreano et al., editor, *Advances in Artificial Life*, volume 1674, pages 621–625. Lecture Notes in Computer Sciences, Springer, 1999.
30. C.G. Langton, editor. *Artificial Life*. Addison Wesley, 1987.
31. F. C. H. Lin and R. M. Keller. The gradient model load balancing method. *IEEE TOSE*, 13:32–38, 1987.
32. M. E. J. Newman. The structure and function of complex networks. *SIAM Review 45*, pages 167–256, 2003.
33. T. White. Routing with swarm intelligence. Technical Report SCE-97-15, 1997.
34. M. H. Willebeek-LeMair and 1. P. Reeves. Strategies for dynamic load balancing on highly parallel computers. *IEEE Transactions on parallel and distributed systems*, 4(9):979–993, 1993.

Index