COLOR ANT POPULATIONS ALGORITHM FOR DYNAMIC DISTRIBUTION IN SIMULATIONS

Cyrille Bertelle, Antoine Dutot, Frédéric Guinand, Damien Olivier

Laboratoire d'Informatique du Havre
Université du Havre
25 rue Philippe Lebon
76600 Le Havre

email:{Cyrille.Bertelle,Antoine.Dutot,Frederic.Guinand,Damien.Olivier}@univ-lehavre.fr

KEYWORDS

Ant algorithms, dynamic computation, communication graph, clustering, auto-organization

ABSTRACT

Most distributed simulations are applications composed of numerous mobile communicating entities that continuously evolve. Such entities are organized or organize themselves in groups or societies of cooperating agent or processes. To keep these simulations efficient, it may be necessary to migrate entities of highly communicating groups so that they remain close, ideally on the same processing unit, while at the same time, we need to maintain a reasonable load for each processor.

We have proposed (Bertelle et al., 2002b) a method to hint migration of entities making a trade-off between communication and load-balancing based on a variant of ant algorithms. We use a dynamic communication graph to model distributed simulations where vertices represent entities and edges communications. Several colonies of ants, each of a distinct color representing a computing resource, compete to mark vertices of the graph using colored pheromones. A color change on a vertex hints the entity it represents to migrate on the corresponding processor. In some cases, the solution obtained is not satisfying. We try to show in this paper the reasons and to give improvements to solve it.

INTRODUCTION

Simulation are often used to study complex systems (ecosystems, traffic road ...). In such simulations there are a very large number of entities whose behavior and interactions describe the evolution of the system. Such models are implicitly distributed due to their heterogeneous swarm nature. This kind of problem does not allow a static placement, we are in front of a non-predictable aspect.

However, when trying to distribute such systems, communications flows and computing resource load may become problematic for correct execution of the simulation. In this article we propose a way to hint entities on locations that try to reduce communication impact on system execution by migrating entities taking care of load-balancing.

The paper is organized as follows: Section describes the graph model that underlies our system. Section provides some background informations on ant algorithms whereas section describe our specific colored ant system and some preliminary results. Section presents our implementation, some experiments, problems encountered with the initial colored ant system, and enhancements made to it. Finally we conclude with further expected improvements and perspectives.

DYNAMIC COMMUNICATION GRAPH

Model

We model the simulation by a graph $G = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is a set of vertices representing entities of the application and $\mathcal{E} = \mathcal{V} \times \mathcal{V}$ is a set of edges $e = (v_1, v_2)$ representing communications between entities v_1 and v_2. We consider communications as being bidirectional and therefore the graph is undirected. Edges are labeled by communication volumes. Each vertex is assigned to an initial processing resource at start. This initial distribution is fixed by the application and for the model we consider it as random.

We call *actual communications*, communications between entities that are not located on the same processor. We try to limit the number of these actual communications by identifying clusters of highly communicating entities (organisations). However merely trying to avoid actual communications could simply migrate all entities on a single processor. Therefore we also need to do some load balancing. Colored pheromones have been introduced for this purpose.

Proceedings 15th European Simulation Symposium
Alexander Verbraeck, Vlatka Hlupic (Eds.)

ISBN 3-936150-28-1 (book) / 3-936150-29-X (CD)

Dealing with a Dynamic Environment

The graph model we defined is dynamic, communication volumes may vary, vertices and edges may appear or disappear. These changes in both topology and valuation are one of the major motivation for using ant algorithms.

Indeed, a monotonic approach is one way to achieve clustering on a graph. It consists in regularly applying a computation on a frozen copy of the dynamic graph, then trying to use this information, though the real graph is still evolving. This approach is problematic: the graph can have changed during computation and results may not be usable any more, creating discrepancies between the real application state and calculated migration hints. Furthermore, it is not incremental, each time the algorithm is performed.

Another way is to use an anytime algorithm. The dynamic graph is considered as a changing environment for computing entities that travel on the graph, taking into account the changes as they appear, and storing the solution directly in the graph, as a side effect of their evolution. Ant algorithms are well suited for that task as it has been shown in (Dorigo et al., 1996).

ANT ALGORITHMS

Our method is based on ant algorithms. These techniques are part of meta-heuristics approaches that yield near-optimal solutions to hard optimisation problems where algorithms that provide an exact solutions are not an issue. Ant algorithms maintain a population of agents, that exhibit a cooperative behaviour (Langton, 1987), by continuously foraging their territories to find food (Gordon, 1995) using optimal paths, creating bridges, constructing nests, etc.

Such self-organization appears from interactions that can be either direct or indirect. Direct communication is done via antennation, trophallaxis, any sort of contact (mandibular, visual, chemical). Indirect communications arise from individuals changing the environment and other responding to these changes: this is called *stigmergy*. For example, ants deposit signals named *pheromones* in the environment that influence others: the more pheromone on a path, the more ants tend to follow it. As pheromones evaporate, long paths tend to have less pheromone than short ones, and therefore are less used than others.

Such an approach is robust and well supports parameter changes in the problem. Besides, it is intrinsically distributed and scalable. It uses only local informations (required for a continuously changing environment), and find near-optimal solutions. Ant algorithms has been applied successfully to various combinatorial optimization problems like the Travelling Salesman Problem (Dorigo and Gambardella, 1997) or routing in networks(Caro and Dorigo, 1997), (White, 1997), but also to DNA sequencing (Bertelle et al., 2002a), graph partitioning (Kuntz et al., 1997) and clustering (Faieta and Lumer, 1994).

COLORED ANT SYSTEM

As shown above, we model large scale distributed applications by a dynamic graph $G = (\mathcal{V}, \mathcal{E})$ where vertices represent entities of the simulations and edges communication between these entities. Edges are labeled by communication volume. The ant algorithm is used to detect clusters of highly communicating entities. To solve load balancing problems we introduce *colored ants* and *colored pheromones* that correspond to available processors. To support our algorithm we extend our graph definition:

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

- *$\mathcal{C}$ is a set of p colors where p is the number of processors of the distributed system.*
- *$\mathcal{V}$ is the set of vertices. Each vertex has a color belonging to $\mathcal{C}$.*
- *$\mathcal{E}$ is the set of edges. Each edge is labelled with a weight. A weight $w(u, v) \in \mathbb{N}^+$ associated with an edge $(u, v) \in \mathcal{V} \times \mathcal{V}$ corresponds to a volume of communications between the couple of entities corresponding to vertices u and v.*

The figure 1 shows an example of a dynamic communication colored graph. The proposed method changes the color of vertices if this change can improve communications or processor 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.

The Colored Ant Algorithm

Our algorithm is inspired by the Ant System (Dorigo et al., 1996). We used it as a base for further improvements that we detail in the section . We consider a dynamic communication colored graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{C})$.

1. Each processing resource is assigned to a color. Each vertex gets its initial color from the processing resource where it appears. For each processor of color $c \in \mathcal{C}$ a number of ants proportional to the processor power is allocated and are uniformly placed on vertices of their color. $\mathcal{F}$ denotes the set of all ants. An ant of color c deposits pheromones of color c.

2. The algorithm is based on an iterative process. Between steps $t - 1$ and t, each ant crosses one edge an reaches a new vertex. During its move, it drops

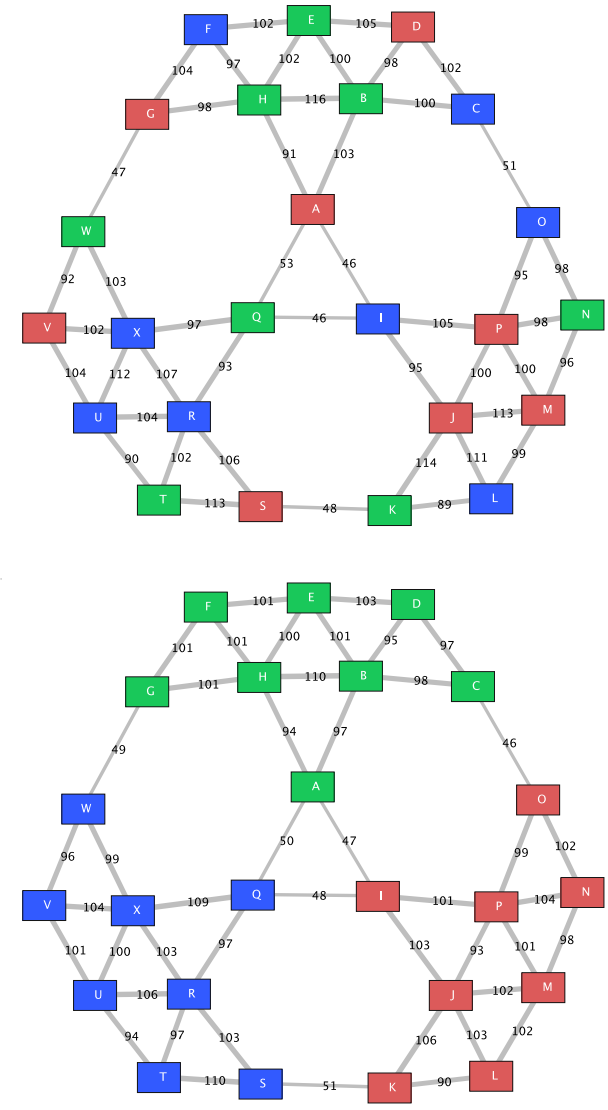

Figure 1: *Example of a dynamic communication graph, at start ($t = 0$), and clustered ($t = 45$).*

pheromone of its color on the crossed edge. Moreover, each ant has the ability to remember the vertex it comes from.

We define the following positive numbers:

- The quantity of pheromone of color c dropped by one ant x on the edge (u, v), between the steps $t-1$ and t is noted $\Delta_x^{(t)}(u, v, c)$.
- The quantity of pheromone of color c dropped by the ants when they cross edge (u, v) between steps $t-1$ and t is noted:

$$\Delta^{(t)}(u, v, c) = \sum_{x \in \mathcal{F}} \Delta_x^{(t)}(u, v, c) \qquad (1)$$

- The total quantity of pheromone of all colors dropped by ant on edge (u, v) between steps $t-1$ and t is noted:

$$\Delta^{(t)}(u, v) = \sum_{c \in \mathcal{C}} \Delta^{(t)}(u, v, c) \qquad (2)$$

- If $\Delta^{(t)}(u, v) \neq 0$, the rate of pheromone of color c on the edge (u, v) between the steps $t-1$ and t is noted

$$K_c^{(t)}(u, v) = \frac{\Delta^{(t)}(u, v, c)}{\Delta^{(t)}(u, v)} \qquad (3)$$

  This rate verifies $K_c^{(t)}(u, v) \in [0, 1]$.

3. The current quantity of pheromone of color c present on the edge (u, v) at step t is denoted by $\tau^{(t)}(u, v, c)$. Its initial value (when $t = 0$) is 0 and then is computed following the recurrent equation:

$$\tau^{(t)}(u, v, c) = \rho \tau^{(t-1)}(u, v, c) + \Delta^{(t)}(u, v, c)$$

   $\rho \in [0, 1]$ represents the pheromone persistence due to its evaporation.

4. At this stage of the algorithm, we have computed the current quantity of pheromone, $\tau^{(t)}(u, v, c)$, in classically, as reinforcement factor for clustering formation based on colored paths. We need now to take into account the load balancing in this auto-organization process. For this purpose, we need to balance this reinforcement factor whith $K_c^{(t)}(u, v)$, the relative importance of considered color with regard to all other colors. This corrected reinforcement factor is noted:

$$\omega^{(t)}(u, v, c) = K_c^{(t)}(u, v)\tau^{(t)}(u, v, c)$$

   Unfortunaly, this corrected reinforcement factor can generate too instable process. So we prefer to use a delay-based relative importance of considered color

with regard to all other colors. For a time range $q \in \mathbb{N}^+$, we define:

$$K_c^{(t,q)}(u,v) = \sum_{s=t-q}^{t} K_c^{(s)}(u,v). \quad (4)$$

According to this definition, we compute the new corrected reinforcement factor :

$$\Omega^{(t)}(u,v,c) = K_c^{(t,q)}(u,v)\tau^{(t)}(u,v,c) \quad (5)$$

5. Let us defines $p(u,v_k,c)$ the transition probability of an edge (u,v_k) incident to vertex u for an ant of color c and whose communication volume is noted $w(u,v_k)$.

   - At the initial step $(t=0)$,

$$p(u,v_k,c) = \frac{w(u,v_k)}{\sum_{v\in\mathcal{V}_u} w(u,v)} \quad (6)$$

   - After the initial step $(t \neq 0)$,

$$p(u,v_k,c) = \frac{(\Omega^{(t)}(u,v_k,c))^\alpha (w(u,v_k))^\beta}{\sum_{v_q\in\mathcal{V}_u} (\Omega^{(t)}(u,v_q,c))^\alpha (w(u,v_q))^\beta} \quad (7)$$

   Where $\mathcal{V}_u$ is the set of vertices adjacent to u.

   The relative values of α and β give the weighting between pheromone factor and communication volumes. We will see later that this weighting is a major factor in the way the algorithm achieves its goals.

   The choice of the next edge crossed by an ant depends on the previous probabilities. However, to avoid the ant moves to oscillate between two vertices, we introduce in the formula a penalisation factor $\eta \in [0,1]$. Given $\overline{v}_x$ the last vertex visited by ant x, the new probability formula is:

$$p_x(u,v_k,c) = \frac{(\Omega^{(t)}(u,v_k,c))^\alpha (w(u,v_k))^\beta \eta_{x,k}}{\sum_{v_q\in\mathcal{V}_u} (\Omega(t)(u,v_q,c))^\alpha (w(u,v_q))^\beta \eta_{x,q}} \quad (8)$$

   Where

$$\eta_{x,q} = \begin{cases} 1 & \text{if } v_q \neq \overline{v}_x \\ \eta & \text{if } v_q = \overline{v}_x \end{cases} \quad (9)$$

6. The color of a vertex u, noted $\xi(u)$ is obtained from the main color of its incident arcs:

$$\xi(u) = \arg\max_{c\in\mathcal{C}} \sum_{u\in\mathcal{V}_u} \tau^{(t)}(u,v,c) \quad (10)$$

Solution Quality

It is necessary to have a measure of the quality of the solution, to know if we must continue to search a solution. They are two aspects to take into account :

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

They are antagonist. So, in order to evaluate our solution we defined two quality criterions r_1 and r_2. The first criterion r_1 allows to know between two solutions which has proportionally less actual communications. Thus we compute global communication costs, noted e, by summing actual communications on the graph (between entities located on distinct processors). Then we compute a ratio r_1 among the total volume of communications, noted s, on the graph and we have:

$$r_1 = e/s$$

The more r_1 is close to 0, the more actual communications are low, as expected. The second criterion r_2 considers the load-balancing. For each color c, we have v_c the number of vertices having color c and p_c the power of processor affected to c. Then we have:

$$r_2 = \frac{min\mathcal{E}}{max\mathcal{E}} \quad \text{where} \quad \mathcal{E} = \left\{ \frac{v_c}{p_c}; c \in C \right\}$$

The more r_2 is close to 1, the more load-balancing is good. For example, we obtain on the two graphs (fig. 1) $r_1 = 0.7$, $r_2 = 1.0$ for $t = 0$ and $r_1 = 0.1$, $r_2 = 1.0$ for $t = 45$.

These criterions are used to compare different solutions obtained during the computation, essentially to verify if we improve the solution during the steps. These criterions, enable us to store the best solution obtained so far.

We use also these criterions to compare communication graphs where clusters are already identified. These graphs are randomly perturbed in term of color allocation to fix an initial configuration. After the algorithm tries to find the initial allocation on the graph, as a solution or a solution where the criterions are closest.

Ant Populations Algorithm

The original algorithm reveals several difficulties due to lack of control over the relations between the numerous parameters. In clusters of high communication, ants tend to follow privileged paths that form *loops*. This is due to the fact communications in such areas are mostly the same and pheromone take a too large importance in ant path choices. Such paths exclude some nodes that could be settled and leads to three problems: *grabs*, *starvation* and *overpopulation* as explained under and shown in figures 2, 3 and 4. In these figures, the graph representation

is as follows. Vertices are rectangles. 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 under 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. These figures are excerpt from the communication graph corresponding to figure 1.

grabs Small ants group of a given color are locked in an area of the graph by a larger colony of another color that surround them. In this case they cannot escape due to repulsion factors and cannot help consolidating correct clusters of their own color elsewhere (figure 2).

starvation Whole parts of the graph get less and less pheromone trails and are left unoccupied by ants. As pheromone evaporate, less and less ants are attracted by them (figure 3).

overpopulation Some parts of the graph get too much ants as the total number of ants on a vertex is not bounded. In the worse case, ants of all colors can appear in such high density populations without the repulsion factors to operate correctly (figure 4).

The base algorithm gives useful results if its parameters are well set, however fixing them is delicate. If they are not correctly configured the algorithm may well fall in local minima as shown in examples above: whether pheromones grow too quickly and overpopulation appears, or they evaporate too fast and starvation comes into play. These two cases leading to grabs.

Control on the ant population as been changed. Before, this control was used only to handle the dynamic graph (vertex and edge appearance or disapearance). Now we add some death and hatching mechanisms. We perturb the ants repartition generating small stable clusters which are the result of local minima. Furthermore this procedure makes senses since our algorithm runs continuously not to find a static solution as the standard Ant System, but to provide anytime solutions to a continuously changing environment.

Including death and hatching ask a question about population: for a given number of vertices is it constant or not? Indeed we want to avoid cases where all ants disappear or too many ants appear. Therefore, we resolved to make one hatch for one death.

Additions to the original Colored And Algorithm are:

1. We define the following positive numbers:

   - $\tau^{(t)}(u, c)$ is the quantity of pheromone of color c dropped on all edges connected to vertex u:

$$\tau^{(t)}(u, c) = \sum_{v_q \in \mathcal{V}_u} \tau^{(t)}(u, v_q, c) \qquad (11)$$

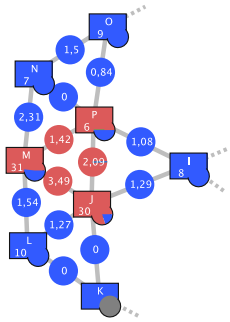

Figure 2: *Grab: red ants are unable to move to the red cluster (not shown here), being hold in a small loop by blue ants. Both groups of ants are responsible: red ants are attracted by their own ever growing loop of pheromone, and blue ants repulse red ones forcing them to remain on the loop.*

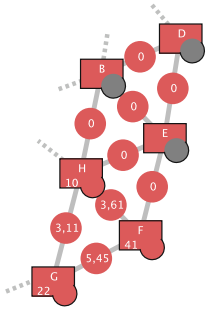

Figure 3: *Starvation: red ants remain on a small loop at the bottom, leaving a whole part of a high communication group unoccupied.*

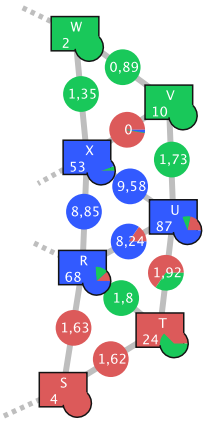

Figure 4: *Overpopulation: the original distribution allocated 20 ants on each vertex. Here there are more than 50 ants on each blue vertex running in a loop. Several other parts (not shown here) of the graph are empty. This problem is due to bad parameters α and β.*

- $\tau^{(t)}(u)$ is the quantity of pheromone of all colors dropped on all edges connected to vertex u:

$$\tau^{(t)}(u) = \sum_{c \in \mathcal{C}} \tau^{(t)}(u, c) \tag{12}$$

- $\varphi_c(u) \in [0, 1]$:

$$\varphi_c(u) = \frac{\tau^{(t)}(u, c)}{\tau^{(t)}(u)} \tag{13}$$

the relative importance of pheromones of color c compared to pheromones of all colors on edges leading to vertex u.

2. Then, at each step, before the ant chooses an arc to cross (equations (6) to (8)), we must choose weither the ant will die or not. We determine this using a thresold parameter $\phi \in [0, 1]$ (preferably small, under 0.1) for an ant of color c on vertex u:

   - if $\varphi_c(u) < \phi$ we make the ant die and create a new ant choosing a new location for it as follows. We select randomly a set $\mathcal{V}_n$ of n vertices. Let $\text{card}(\mathcal{F}(v))$ be the number of ants on vertex v. Then we select a vertex u in $\mathcal{V}_n$ using:

$$u = \arg \min_{v \in \mathcal{V}_n} (\text{card}(\mathcal{F}(v))) \tag{14}$$

   and make the new ant hatch on it.

   - else, we proceed as specified in the original algorithm choosing a new edge using probablities (equation (6) and following).

This procedure eliminates grabs and starvation. Grabbed ants die, and hatch in starvation areas. However it does not eliminate loops, and sometimes loops tend to reappear. In order to break them, we introduce more memory in ants: instead of being able to memorize only one vertex, ants can memorize three or more vertices.

IMPLEMENTATION AND EXPERIMENTATION

In these experiments dynamic valuations of the edges are not shown, we concentrate on the pheromone levels, edge coloration and ant populations.

Results of the Base CAS

We tried the CAS on the graph given in figure 1. This particular graph is problematic. Finding good parameters for it was difficult. The algorithm stagnating in a solution not very closed to the optimal solution (see figure 5 with parameters: $\alpha = 1$, $\beta = 4$, $\eta = 0.0001$, $\rho = 0.8$). This configuration appears after 30 steps and stays identical after 570 more steps.

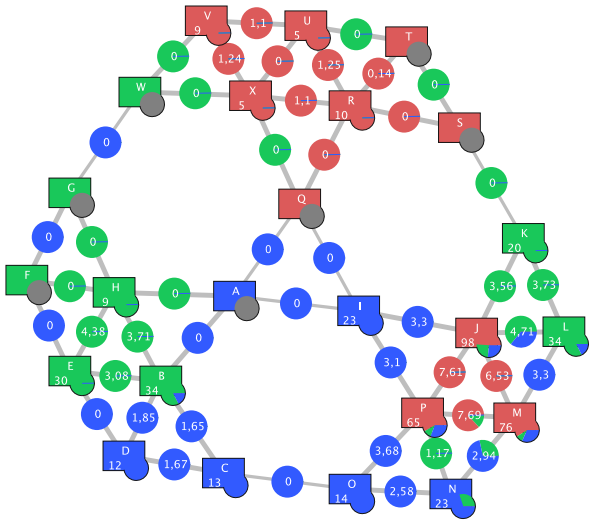

Figure 5: *The original algorithm on a problematic graph after 600 steps*

Results With Ant Population Control

We then used the new algorithm with population control on the same problem with the same parameters plus the two new parameters: $\phi = 0.3$ and the memory set to 4. A correct solution appeared after 23 steps as shown in figure 6. After 200 steps the configuration was stable (figure 7),

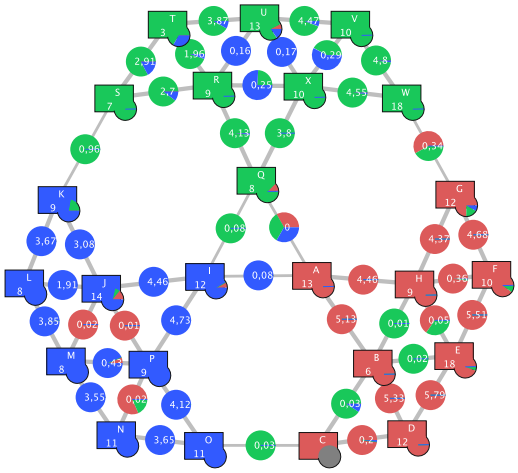

Figure 6: *The algorithm with population control after 23 steps.*

and remained the same at 600 steps (figure 8) where the experiment was stopped.

We also tested the algorithm with grid like graphs like shown in figure 9 and 10. Parts of the grid are subgrids of high communication. A good solution was found after

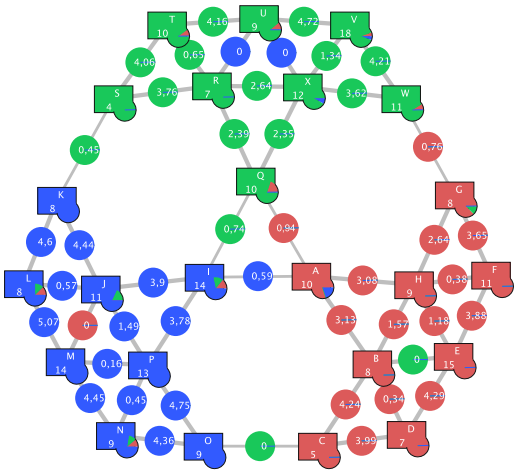

Figure 7: *The algorithm with population control after 200 steps.*

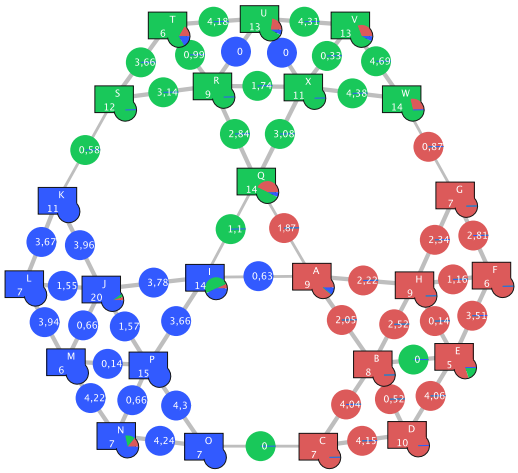

Figure 8: *The algorithm with population control after 600 steps.*

50 steps and stabilized at 100 steps.

We designed a software tool that allows us to edit graphs and modify them as the ant algorithm performs, adding or removing edges and nodes at runtime. The graph shown in figures 11, 12, and 13 is a modification of 1. we also obtain good results, though load balancing is here not optimal. Playing with parameters or adding a new processor (new color) breaks the biggest cluster in two or three (placing some nodes of it in two other colors, or allocating a new color to it following the solution choosen).

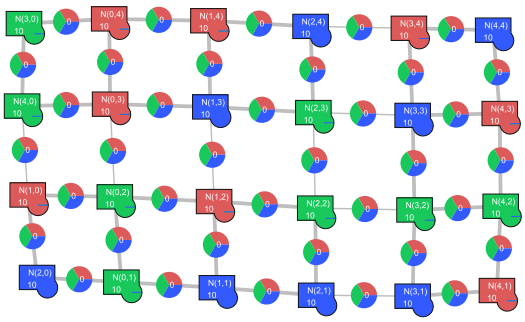

Figure 9: *The initial grid.*

CONCLUSION

In this paper we presented a variant of the Ant System called Colored Ant System that offers advices for entity migration in a distributed system taking care of the load and communication balancing. We described a base colored ant algorithm and then provided several improvements and shown their results.

This works aims at improving repartition of hydrosystem simulations that use a large number of distributed entities. Such simulations form a dynamic communication graph continuously evolving and requires an incremental distribution algorithm that can adapt to their dynamic nature and can provide placement hints at any time.

In the near future, we will consider handling dynamic ant populations where the ant count adapt to the available power and load of associated processors. Later we plan to add more control to our system: indeed, this system is reactive in nature, and it would be desirable to add some processing before giving migration hint to the application (simulation). Therefore, we plan to add an heuristic layer between the colored ant system layer and the application layer that will control and smooth the results given by the CAS. This layer would allow us to take into account constraints tied to the simulation that cannot be directly introduced in the ant system (e.g. non-migrable resources like databases).

REFERENCES

Bertelle, C., Dutot, A., Guinand, F., and Olivier, D. (2002a). Dimants: a distributed multi-castes ant system for dna sequencing by hybridization. In *NETTABS 2002*, AAMAS 2002 Conf, Bologna (Italy).

Bertelle, C., Dutot, A., Guinand, F., and Olivier, D. (2002b). Distribution of agent based simulation with colored ant algorithm. In *ESS2002*, pages 39–43, Dresden (Germany).

Caro, G. D. and Dorigo, M. (1997). Antnet: A mobile agents approach to adaptive routing. Technical report, IRIDIA, Université libre de Bruxelles, Belgium.

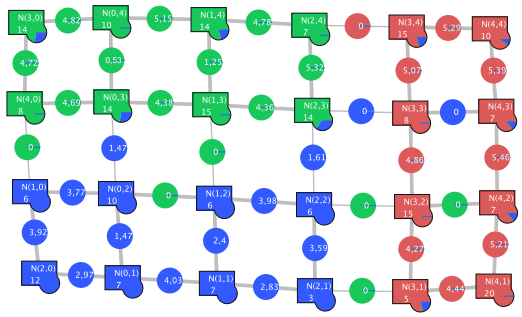

Figure 10: *The algorithm on a grid after 100 steps.*

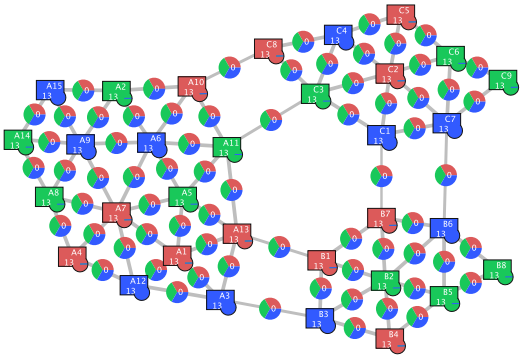

Figure 11: *The original graph.*

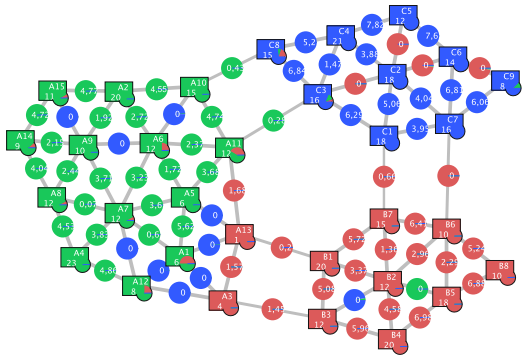

Figure 12: *After 50 steps.*

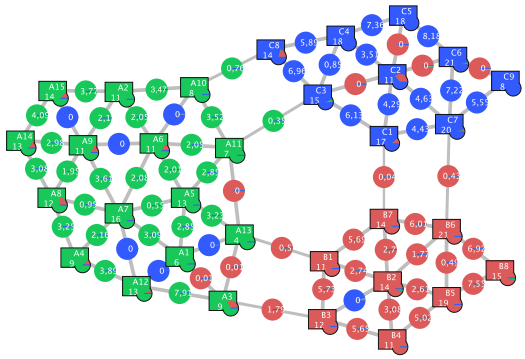

Figure 13: *After 200 steps.*

Dorigo, M. and Gambardella, L. (1997). Ant colony system: A cooperative learning approach to the traveling saleman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66.

Dorigo, M., Maniezzo, V., and Colorni, A. (1996). The ant system: optimization by a colony of cooperating agents. *IEEE Trans. Systems Man Cybernet.*, 26:29–41.

Faieta, B. and Lumer, E. (1994). Diversity and adaptation in populations of clustering ants. In *Conference on Simulation of Adaptive Behaviour*, Brighton.

Gordon, D. (1995). The expandable network of ant exploration. *Animal Behaviour*, 50:995–1007.

Kuntz, P., Layzell, P., and Snyers, D. (1997). 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.

Langton, C., editor (1987). *Artificial Life*. Addison Wesley.

White, T. (1997). Routing with swarm intelligence. Technical Report SCE-97-15.

AUTHOR BIOGRAPHIES

Antoine Dutot is PhD student in computer sciences research laboratory of Le Havre university (LIH). He works on swarm intelligence models for dynamic repartition of distributed simulations.

Cyrille Bertelle is associate professor in computer sciences research laboratory of Le Havre university (LIH). He works on natural complex systems modelling.

Damien Olivier is associate professor in computer sciences research laboratory of Le Havre university (LIH). He works on natural complex systems modelling.

Frédéric Guinand is associate professor in computer sciences research laboratory of Le Havre university (LIH). He works on schedulling for the parallel and distributed applications and on bioinformatic and genomic.