Description of the project (phase1)

Introduction

Ant Colony Optimization (ACO), [1], is a method used for solving optimization problems inspired by the foraging behavior of some ant species. These ants deposit pheromone s on the ground in order to mark some favorable path that should be followed by other members of the colony.

The impact of the implicit interaction mechanism through the environm ents (by means of the pheromone trails) on the optimization problem of finding the shortest path between the ant hill and a location where there is some food is to be studied as a first phase of the project. However, better approaches can be used and an explicit coordination strategy may improve the performance of the system. Thus, the second phase of the project is to propose and use a strategy of coordination between the ants so that we expect some better results in terms of the number of iterations needed, the number of ants used, etc. Thus, the new ants created, named smart ants, are expected to behave better.

The study should end up with a comparative analysis of the two approaches.

ACO Ant System (AS) Algorithm

As described in [1], for the AS algorithm at each iteration, the pheromone values are updated by *all* the m ants that have built a solution in the iteration itself. A simple Java implementation of the system is offered, together with a testing configuration, for the same travelling salesman problem (TSP) as the one described by the specified work.

We would like to modify this problem. First, we would like to distribute this approach, to a certain degree, for the coordination phase to come and then we would like to imagine a new scenario for this new approach. The method together with the new scenario and with the requirements are described in the subsequent sections.

The Problem

A colony of ants has to find the shortest path in a network of locations to a place where they can find food. The topology of the network is given at the beginning of the simulation. The graphs used are undirected and each simulation is described by a number of iterations in which the ants find their way to the food, based on the intensity of a trail of pheromones left on the edges of the graph. After each iteration the trail of pheromones is updated based on the solutions found by the ants.

The ants are at the location of the ant hill at the beginning of the simulation and at the beginning of each subsequent iteration. A trace of pheromones is left on each path between two locations by all the visiting ants, at their return trip to the ant hill, according to a modified version the “Ant System” ACO algorithm presented in [1]. The modification consists in the fact that the ants have some information of the solution found when they reach the food location (or if they don't reach it at all) so they can modify the trail of pheromones, on the way back, by themselves. The constant Q can be replaced by the quantity of food transported (usually 10, representing 10 times the weight of an ant, but can be less if the quantity of food available at the location found is smaller than that). Thus, if the ants don't find a location where food is stored (if, for example, they reach a node from which there is no unvisited edge), their contribution to the pheromone trail during that iteration will be zero.

After a number of trips to a place where food is found and back to the ant hill the path chosen by the ants should be optimal, thanks to the pheromone trails left and to an evaporation rate of the pheromone trace that causes the longest paths to be abandoned.

A solution to this problem is offered by a mobile agent-oriented approach inspired from the S-CLAIM language, [2].

The Agent-Oriented Approach

We consider a hierarchical MAS (multi-agent system), similar to the ones described by the S-CLAIM language, representing the system we desire to develop. Each separate entity in the system is represented by an agent. Thus, the actions of sensing or of acting upon the environment will be made by means of communication between the agents and also by modifying the internal state of some key agents. Thus, the agents will be described by at least the followings :

- an internal state;
- a communication component; (agents can communicate with other agents they already know, with their parents and they can also answer to the messages they receive. the parents know their children and each agent knows its parent too. additionally, the names of other agents can be stored)
- a reasoning & decision making component.

We use the *in* and *out* mobility primitives inspired by the S-CLAIM language and described as follows:

- *in* entering (moving to) the subhierarchy of an agent;
- *out* getting out of the subhierarchy of the current parent and entering the hierarchy of its parent (i.e. grandparent) if any.

Agentification of the Described Scenario

A proposal of the agent classes to be used is the following one :

- **Manager** (1) – responsible with the preparation and execution of the simulation and offering a global visualization interface. It is also the root of the hierarchy of agents;
- **Location** (many) – Children of the Manager agent, they are associated with the locations (nodes) from the graph imagined at the beginning of the simulation. They are aware of the edges starting from and coming to the node they represent.

- **AntHill** (1) – Child of a Location agent, it is responsible with the creation of the ants and with storing the food brought by the ants;
- **FoodDeposit** (1 or more) – Child(ren) of a Location agent, it manages the food that can be found at a specific location;
- **Ant** (many) – they travel, seek food and transport it to the ant hill and leave pheromones.

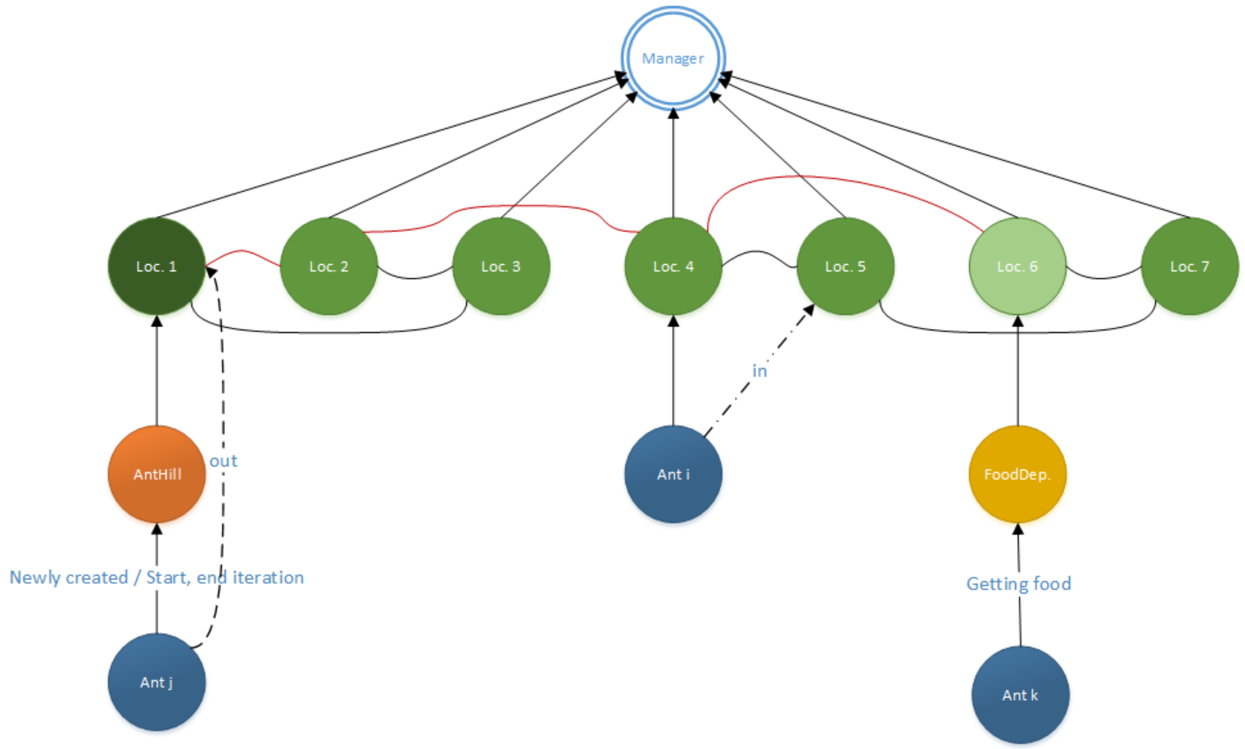

Fig. 1 The agent hierarchy used. The agents are represented by circles. The continuous arrows are used to illustrate the parents. The dashed arrow represents a following *out* primitive. The dash-dotted arrow represents a subsequent *in* primitive. The continuous lines represent the edges of the graph and the red ones are part of the preferred path.

In Fig. 1 we can see an agentification of the ACO scenario described above.

An Example

We consider the ants as software agents travelling through a network of computers (each computer being represented by a Location agent). After the first few iterations the trail of pheromones will look like in Fig. 2, where the color of the location is the sum of the intensity of the pheromone values of all the edges containing it over the mean intensity of these sums, with red denoting a high intensity, green a low one and yellow (the mean) being between them.

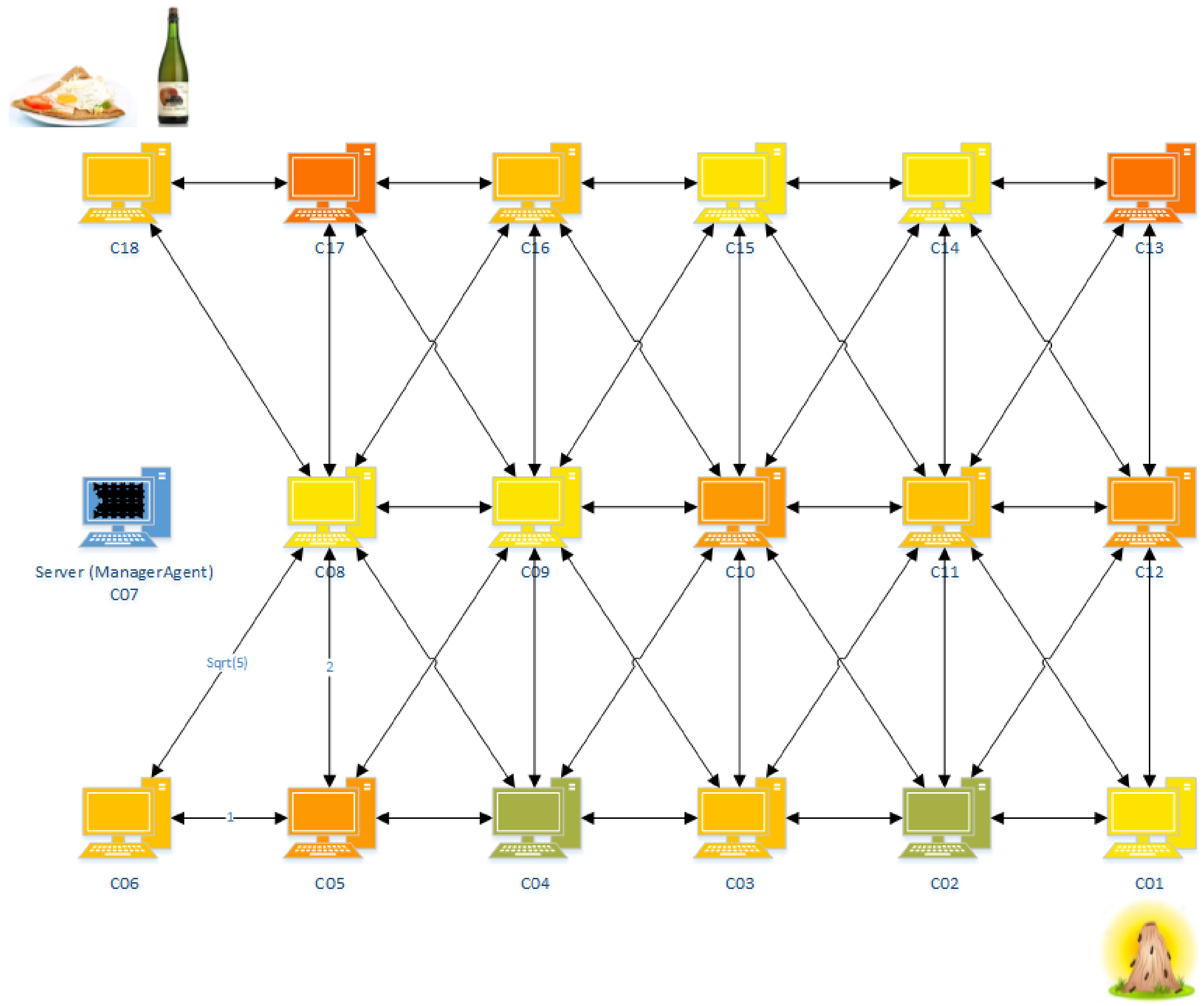

Fig. 2 Configuration of the system after a few iterations

After many iterations the ants will prefer a specific, short, path to reach the food, as illustrated in Fig. 3.

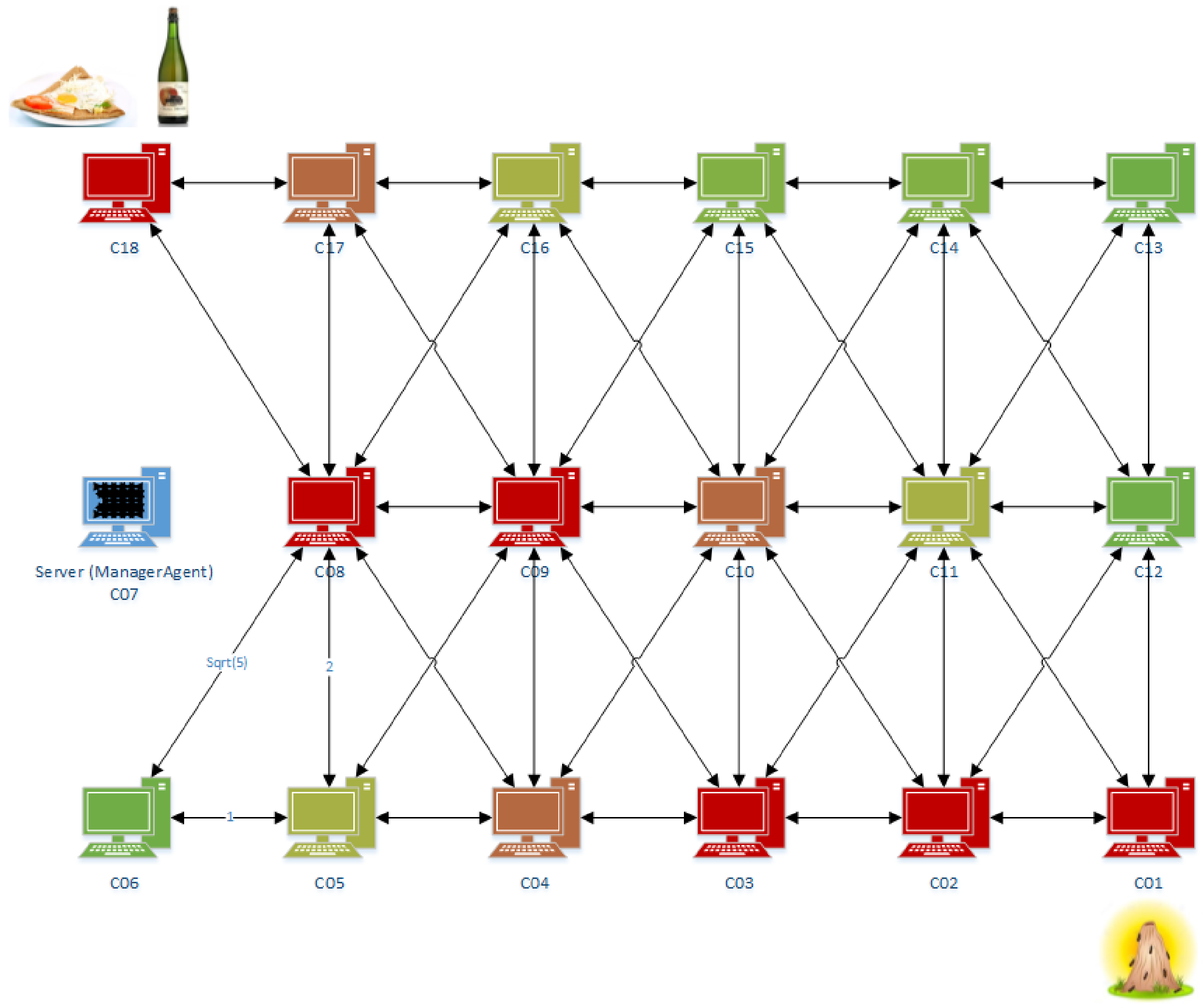

Fig. 3 Configuration of the system after many iterations

Details about the Implementation

The implementation language and / or framework used is the choice of each student (the preference of the assistant is Java but the student can use whatever he considers to be in his advantage). However, it should be clear from the way the system acts and from the results what happens and how it was built and the student should be able to justify specific behaviors.

A set of files describing the example from the previous section is also offered to the students.

Homework (Phase 1 – a total of 10p)

- implement the scenario presented in this document using the suggested approach; (6p)
- analyze the behavior of the system with respect to the :
    1. number of ants ; (1p)

2. number of iterations (to transport all the available food); (1p)
3. <u>quantity of food transported. (1p)</u>

- Offered: (1p)

References

[1] Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization.*Computational Intelligence Magazine, IEEE*, *1*(4), 28-39.

[2] Baljak, V., Benea, M. T., Seghrouchni, A. E. F., Herpson, C., Honiden, S., Nguyen, T. T. N., ... & Toriumi, S. (2012). S-claim: An agent-based programming language for ami, a smart-room case study. *Procedia Computer Science*, *10*, 30-37.