A Model for Agent Mobility and Interaction

Pedro Mariano*, Mário Marques†, Luís Correia*, Rita Ribeiro†,
Vladimir Abramov, Jan Goosenaerts,
Maria Chli, Philippe De Wilde
*Informatics Department, New University of Lisbon, Portugal
Email: plm@di.fct.unl.pt, lc@di.fct.unl.pt
†UNINOVA, New University of Lisbon, Portugal
Email: mjsmarques@netcabo.pt, rar@uninova.pt

***Abstract*— As information infrastructures move towards open systems where agents come and go, new facilities are required so that these agents can take advantage of each others functionalities. We need agent systems that can provide to newcomer agents a place and the right agent to interact with. Such functionality must cope with high rate of agent entrance, with high load of agents, with vanishing agents or nodes in the agent system. Given these requirements, agents are constantly facing a problem of deciding where to go and with whom to work with. These two decisions, pertaining to mobility and interaction, have been singled out as fundamental of every agent system. We present an algorithm targeted at these two decisions while it fulfils the aforementioned requirements.**

I. Introduction

The model that we developed, inspired by natural immune systems [1], is targeted at the problems of mobility and interaction in a distributed multi-agent system. It aims at providing a solution to the problem of deciding with whom to work on a goal and where to go. The model was developed for a particular class of agent systems where agents can be classified as producers and consumers. We assume that: agent systems are composed of agents and network nodes; agents can travel through network nodes; agents interact inside a network node.

II. Definitions

A. Agent System

As we have already said, there are agents and network nodes. Agents have unique identifications, taken from a subject space (S). We will use the word *subject* as agent identity. Network nodes have unique addresses, taken from a network space (N). We will use the word *node* whenever we refer to a network node. Agents' goals are described using terms from a conceptual space (C). We will use the word *concept* as a description of an agent's goal. A concept corresponds to a resource. See table I for a definition of these terms.

subject	s_i
subject space	$S = \{s_1, s_2, \ldots s_i, \ldots\}$
node	n_j
network space	$N = \{n_1, n_2, \ldots n_j, \ldots\}$
concept	r_k
conceptual space	$C = \{r_1, r_2, \ldots r_k, \ldots\}$

TABLE I
SOME GLOSSARY TERMS USED IN THIS PAPER.

goal	$g^y \in G$
specialization	$G = \{r_{k_1}^{x_1}, r_{k_2}^{x_2}, \ldots r_K^{x_K}\},\ K \leq 2\|C\|$
concept-subject map	$CS = \{(r_k^x, s_i), \ldots\}$
concept-node map	$CN = \{(r_k^x, n_j), \ldots\}$
subject-node map	$SN = \{(s_i, n_j), \ldots\}$

TABLE II
SOME AGENT PROPERTIES.

B. Agents

In the model we present, agents are characterised by a set of concepts they can produce and another set of concepts that they can consume. These sets may be empty. The union of the two sets forms the *agent's specialisation*. When required, we will use the superscript ^{P} to identify a concept that one agent produces, and we will use the superscript ^{C} to identify a concept that one agent consumes. On a single instant of time, an agent's goal description is a concept from the specialisation set.

During their lifetime agents collect information about their movements and interactions. For each interaction, an agent records the node where it occurred, the other agent's goal, and the other agent's identity. This information is stored in three maps: concept-subject, concept-node, and subject-node. These three maps form the *agent's experience*. In addition to exchanging the previous information, agents can share part of their experience. See table II for an overview of these agent's characteristics.

The agent behaviour is modelled by a finite state automaton with four states: agent does nothing; agent travels between network nodes; agent wants to talk with somebody else; or agent wants to accomplish its goal by working with another agent. Agent interactions (talk and work) are limited to two agents. One agent needs a second agent to achieve its goals. If the description of their goals match (one agent consumes the resource that the other produces), both accomplish their goals.

Function 1 defines the goal matching function for any two concepts $r_{k_1}^x$ and $r_{k_2}^y$. The ^{x} and ^{y} values are the aforementioned superscripts, ^{P} and ^{C}.

$$match(r_{k_1}^x, r_{k_2}^y) = \begin{cases} 1 & \text{if } r_{k_1} = r_{k_2}, x \neq y \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Agents can have a goal quality measure that could be used to distinguish between good and bad ones. After an agent reached its goal, it may start over and accomplish

another one, or leave the agent system. The description of the new goal is taken from its specialisation set.

III. Immune Systems

The model is targeted at the decisions of selecting a destination network node and an agent to interact with. It is inspired by natural immune systems [1], [2].

The main functions of immune systems are recognition and categorisation [4]. These are some of the functions that components of agent systems must perform. Agents must recognise, categorise and rank partners; they must be able to recognise the concepts they deal with. Agents also travel, therefore they need to select where to go. To sum up, we need to recognise and categorise agents and network nodes. We will now describe how this can be performed.

Given the definitions presented in section II, agents need to single out other agents related to the description of their goal, which implies selecting an agent and selecting a node. These two decisions depend on information stored on the concept-subject map and concept-node map. These maps basically say: choose this agent because of that concept and go to this node because there are agents that know that concept.

The metaphor of self and non-self can now be readily applied as follows: self-cells and non-self cells are concept-dependent; self-cells indicate agents one should choose or nodes one should go to; conversely, non-self cells indicate agents one should not choose or nodes one should not go to. After each agent accomplishes its goal, a measure of the outcome can be used to update the proportion of self and non-self cells. In the next section, we discuss some measures that can be used to control this proportion.

IV. The Model

A. Algorithm Outline

Each agent locally runs the algorithm using as inputs its goal and the three maps. The algorithm ranks nodes and agents according to some criteria. It produces as output the node, $n^\star$, with the highest value, $u(n_j)$, and the agent, $s^\star$, with the highest value, $u(s_i)$.

The best candidate node to travel to is the one known to have more agents related to the first agent's goal. Conversely, the best agent to work with is the one known to have more concepts related to the first agent's goal.

The network node value, $u(n_j)$, is calculated as follows:

$$u(n_j) = \sum_{(r_k^x, n_j) \in CN} match(r_k^x, g^y) \qquad (2)$$

where the superscripts ^{x} and ^{y} can take any of the two values ^{P} and ^{C}, and the function $match$ is defined in 1.

Example – Node choice Suppose that agent *John* has the following concept-node map:

$$CN = \{(drill^C, 1.2.3), (drill^P, 1.2.3), (screwdriver^P, 2.4.6), (drill^C, 3.3.3)\}$$

and its goal is $g^y = drill^P$. Network nodes values are as follows: $u(1.2.3) = 1$, $u(2.4.6) = 0$, and $u(3.3.3) = 1$. Host *2.4.6* has zero value because it is known not to have the concept the agent is looking for. Host *3.3.3* has a value of one because it is known to have one instance of the concept. Host *1.2.3* while known to have two instances of the concept the agent is looking for, only one of them comes from a consumer, therefore it has a value of one. ■

The agent value can be calculated on a network node basis or disregarding the agent's location. Since agents first select a node and then an agent, agent values should be calculated on a node basis. Given a selected node, $n^\star$, the agent value, $u(s_i)$, is calculated as follows:

$$u(s_i) = \sum_{\substack{(r_k^x, s_i) \in CS \\ (s_i, n^\star) \in SN}} match(r_k^x, g^y) \qquad (3)$$

Example – Agent choice Suppose now that agent *John* has selected node $n^\star$ to travel to and the relevant tuples from the concept-subject map, that is to say, the tuples (r_k, s_i) such that $(s_i, n^\star) \in SN$, are:

$$\{(drill^C, Maria), (drill^P, Anna), (hammer^C, Peter)\} \subset CS$$

and its goal is $g^y = drill^P$. From the above we compute agent values are as follows: $u(Maria) = 1$, $u(Anna) = 0$, and $u(Peter) = 0$. Agent *Peter* has zero value because it is known not to have the concept the agent is looking for. Agent *Anna* while known to have the concept the agent is looking for, she is a producer, therefore her value is zero. Agent *Maria* has a value of one because she is a consumer of the product agent *John* produces. ■

B. Biased Algorithm

The previous algorithm does not take into account any goal quality. Agents may measure how satisfied they become after achieving their goal. Generally, producers and consumers will have different measures. In [5], the author considers how agents can rate each others' goals. Three measures were proposed: quality, cost, and duration. We will briefly discuss how they can be used in our framework.

- Quality – How good is the concept produced or consumed (it may have different meanings whether the agent is a producer or a consumer).
- Cost – How much did it cost to achieve the goal. Producers may have to pay a production cost but may receive some money whenever a consumer gets a resource. Likewise, consumers may have to pay to obtain the resource. Other costs, rather than resource price, may be taken into account.
- Duration – How much time did the agent take to achieve the goal.

Independently of the used measure, it is advantageous that a high value (meaning good quality) should increase the corresponding node value and agent value. Since agent and node values depend on the number of tuples (see equations 2 and 3), a goal that results in a high value should increase the number of $(g^y, s^\star)$ and $(g^y, n^\star)$ tuples, while a low value should decrease the number of tuples.

Let us define v_G as the value that an agent gets from accomplishing a goal. The reference value, v_0, is a fair value, in that it maintains the number of tuples. The changes to the number of tuples in the concept-subject map and concept-node map are given by, respectively, expressions 4 and 5.

$$\Delta_{\#(g^y,s^*)} = round(w_s \cdot (v_G - v_0)) \quad (4)$$

$$\Delta_{\#(g^y,n^*)} = round(w_n \cdot (v_G - v_0)) \quad (5)$$

Constants w_s and w_n are weights that control the amplitude of the change: a high value results in a higher change to the number of tuples, while a low value results in a small change to the number of tuples.

The metaphor of immune system can be easily established with this version of the algorithm. The previous equations (4 and 5) provide a way to increase or to decrease the number of tuples which in turn will affect who is selected to travel to and to be talked with. This change (in the number of tuples) is similar to the Clonal Selection algorithm [1].

C. Experience maps management

The algorithms we presented use the information stored in the experience maps. The last algorithm modifies the contents of these maps. The agent's behaviour generates more information to put into these maps. However, these maps cannot grow forever. Each map has a maximum size. Whenever new information must be put into a map, that does not have enough space, random tuples are selected to make up the space for the new tuples.

We could attach time stamps to tuples and remove the oldest. This is subject to future work, but for the time being, random removal is sufficient as the most common tuples are always selected. In conjunction with the second algorithm this mechanism keeps the tuples that result in higher value goals.

Suppose that map CS has 1 tuple $(drill^C, Anna)$ and $n-1$ tuples $(drill^C, Maria)$. The agent's goal is $g^y = drill^P$. Agent *Maria* is chosen to talk to, since it has a higher value. Since the map CS is full, one tuple is chosen to make up space. If it is any of the tuples $(drill^C, Maria)$, agent *Maria* will still be chosen the next time the agent has the same goal. If it is the tuple $(drill^C, Anna)$, again agent *Maria* will be chosen. The probability that the tuple $(drill^C, Anna)$ is present in the CS map after m interactions is $(n-1)^m/n^m$.

V. Results

A. Mobile Agent Model

The results mentioned in this work use a framework with static agents and mobile agents. Static agents dispatch mobile agents that perform the role of information gathering, and then use this information to select a network node to go and an agent to work with. Mobile agents' parameters are composed of a set of nodes to visit and a deadline. During each visit to a network node, mobile agents will try to talk to all agents (at the node) and collect information to store in their experience maps. Whenever these experience maps get full, or all the nodes have been visited or the deadline has been reached, mobile agents will return to their respective static agent and deliver the information they collected. Mobile agents' parameters are generated by an evolutionary algorithm using the sum of expressions 2 and 3 as a fitness measure.

B. Scenarios

The scenario used was a trading system with buyers and sellers of different products. These two roles were performed by the static agents. Buyer agents actively selected adequate seller agents. Both sent mobile agents. Mobile agents could share between themselves information they collected. Since only buyers selected sellers, mobile agents sent by sellers work only as promoting agents. Mobile agents sent by buyers gather information that is used by the respective buyer in the agent decision process. Whenever one seller and one buyer traded on a product, we considered that both have accomplished their goal. After this, they received a new goal. This new goal is taken from their respective specialisations.

To assess our model, we performed a set of control simulations where agent selection (to work on a goal) was done randomly. Static agents still launched mobile agents to collect information, but instead of selecting the node with the highest measure, they selected one node randomly from the set of known network nodes.

Some measures were taken in order to compare the two models:

- solved tasks – this is a counter on how much goals an agent has achieved. We measured it for all agents in the agent system.
- average task solving time – an average of agent lifetime (in simulation steps) divided by number of accomplished tasks.

We ran the simulation for $T = 1000$ time units. We varied the number of different resources in the agent system from 1 to 3. The number of static agents was 100, and the number of nodes was 10. Static agents only traded at most one product per simulation time unit. We also tested our algorithm in face of some types of failures. In one configuration there was no accident, while in a second configuration at $T/2$ one third of the static agent population was killed (along with their mobile agents). The purpose of the first configuration is to compare the node and agent selection model against a random choice, while the second is to assess how our model behaves when an accident occurs. Each simulation configuration was run 10 times. The following table shows the different configurations tested:

node agent selection	our model random
number resources	1, 2, 3
accident	A – no accident B – kill 1/3 agents at $T/2$

C. Comments

In the configuration A (no accident), agents with our model accomplished more goals than agents with the random choice model. However, the difference is not significant. Whereas in the configuration $accident = B$ there is a clear difference between the two types of agents.

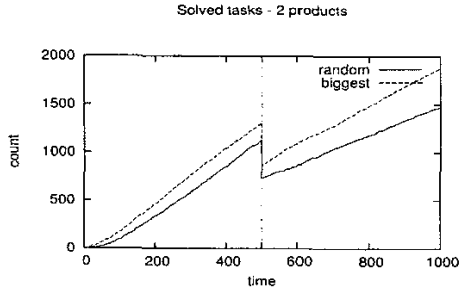

Fig. 1. This plot shows the number of accomplished goals versus time with 2 products. Each line is an average of 10 simulation runs (all of them with the same setting). At $t = 500$ one third of all static agents was killed.

Agents that make random choices, regarding the node where they will go to, complete less tasks than agents that go to the node with highest rating. In most cases, there is no difference. For instance, in the case of 2 products, the two plots match each other. The behaviour is the same in both scenarios.

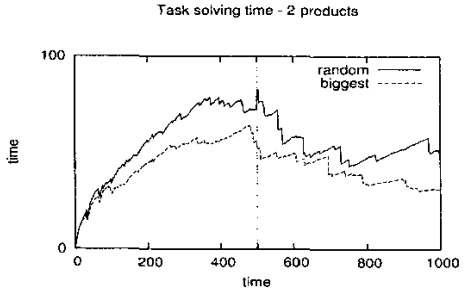

Fig. 2. This plot shows the goal accomplishment time versus time with 2 products. Each line is an average of 10 simulation runs (all of them with the same setting). At $t = 500$ one third of all static agents was killed.

In order to assess the behaviour of our model, in the configuration B (accident), we plotted the average goal accomplishment time. Figure 2 shows typical results. Again, each line is an average of 10 simulation runs. After the accident, agents with our model seemed to behave well. They accomplished their goals in less time than agents with random choices.

An agent with our model always goes to a network node known to have agents that can help him achieve its goals. There is always the possibility that there is no agent in the network node, which in the case of accident B has a higher probability of occurring. Compared to a second agent with the random node selection, which can go to a network node with no agent to work with, the first agent will spend less time looking for an agent to work with, and thus solves more tasks in less time.

VI. Conclusions

We have presented an algorithm targeted at the fundamental decisions of network node selection and agent selection, and we have applied it in a scenario where agents must interact and work together to trade different resources. Such agents live in an agent system composed of network nodes where agents move around. Results have shown that our model improved agents' accomplishing capabilities, while providing robustness to different types of accidents and conditions.

We have used a trading model that is discussed elsewhere [6]. In order to assess our model for agent mobility and interaction, we could have abstract goals. Each agent would have a goal solving probability and a goal quality measure drawn from a normal distribution. Whenever two agents would met together to work, we would compute the probability of solving a goal. If one agent could solve a goal, we would then compute its quality from the normal distribution.

Scenarios characterised by agents that must interact within a distributed information infrastructure, are potential candidates for our model. In addition to the trading model used, others include information retrieval, and transportation. In the first, agents have to search for information that is scattered throughout the network. If agents are constantly updating the information they own, other agents must know who are the agents that hold the information they are looking for. The second scenario, is similar to the trading scenario, however agents must find out other agents that are able to deliver the resources they hold.

We need to test other types of accidents such as a node that disappears along with any agent there located. Such assessment aims at testing the adaptability of the agent system: if agents are still able to find other partners and work together on their tasks. The model agents use to select where to go and with whom to work with must react to changes in the agent system: new agents and their locations, agents leaving the system, or accidents such as disappearing agents and nodes. While expressions 4 and 5 were defined when an agent accomplished its goal, they can be also used when the agent chose a no longer existing node or agent. In this case we could set v_G to some minimum value that would guarantee a fast reaction of our model.

Acknowledgments

This work has been carried in the framework of the research project Ecology and Evolution of Interacting Infohabitants, No. IST-1999-10304 supported by the European Union.

Pedro Mariano is financed by grant no. SFRH/BD/ 1219/2000 supported by FCT/MCT of Portugal.

References

[1] L. N. de Castro and F. J. V. Zuben, "Learning and optimization using the clonal selection principle," *IEEE Transaction on Evolutionary Computation, Special Issue on Artificial Immune Systems*, 2001.

[2] S. A. Hofmeyr and S. Forrest, "Architecture for an artificial immune system," *Evolutionary Computation*, vol. 8, no. 4, pp. 443–473, 2001.

[3] H. Davies, *Introductory Immunobiology*. Chapman & Hall, 1997.

[4] D. Dasgupta, "Artificial neural networks and artificial immune systems; similarities and differences," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, 1997.

[5] V. Lesser, "Reflections on the nature of multi-agent coordination and its implications for an agent architecture," *Autonomous Agents and Multi-Agent Systems*, vol. 1, pp. 89–111, 1998.

[6] M. Marques, P. Mariano, R. Ribeiro, L. Correia, M. Chli, P. D. Wilde, V. Abramov, and J. B. Goossenaerts, "Contributions to adaptable agent societies," project Ecology and Evolution of Interacting Infohabitants, Tech. Rep., 2003, submitted to 9th IEEE International Conference on Emerging Technologies and Factory Automation.