EXPLORING UNKOWN NETWORKS USING A COOPERATIVE MAS-BASED APPROACH

Pedro Simeão Carvalho, Rosaldo J. F. Rossetti, Ana Paula Rocha, Eugénio C. Oliveira

Informatics Engineering Department, Artificial Intelligence and Computer Science Lab
Faculty of Engineering, University of Porto
Rua Dr. Roberto Frias, S/N, 4200-465 Porto, Portugal

{pedro.simeao, rossetti, arocha, eco}@fe.up.pt

ABSTRACT
This paper reports on a novel method to explore and map an entirely unknown network using a cooperative Multi-Agent System (MAS) to extract knowledge or information from nodes and connections. We consider the likely presence of obstacles, eventually making the network disconnected. The MAS architecture is applicable to a vast range of scenarios. Our main goal is to discover the entire network as quickly as possible, characterizing its nodes' meta-structures. In this paper, we propose a novel method that relies on agents that can communicate to each other through simple messages, ensuring that there is no resource sharing. The proposed method is compared to other two non-cooperative methods through simulation, in order to establish a basis for comparison. Preliminary results show that our cooperative approach produces better results than the other two implemented and guarantees that the entire network is explored at the end.

Keywords: network exploration, multi-agent systems, cooperation in MAS.

1. INTRODUCTION

The motivation for this work is the need to completely explore an *a priori* unknown network, defined as an abstract set of nodes connected to each other through edges. There is also the need to consider the presence of "dark nodes" on the network, which are nodes that can be neither analyzed by agents nor even crossed by them while moving over the network (e.g. physical or abstract obstacles, unreadable nodes, etc.)

This problem specification considers no specific application domain and is applicable to a vast range of scenarios, provided they can be represented as a network. This approach might be used, for instance, in space exploration vehicles or robots, data block processing, navigation systems, social networks, and generic network discovery. Thus, our main goal is to discover the entire network as quickly as possible, characterizing its nodes' meta-structures and/or its connections.

In the past years, the widely adoption of Multi-Agent Systems (MASs) has increased in problem solving across many areas, such as informatics, intelligent systems and even the industrial sector. The MAS concept allows the use of distributed computation that can be either virtual or physical. These agents are autonomous, share an environment through communication and interactions, and make decisions according to the situation (Parker 2003). This approach is useful when processing a considerable amount of data such as in large networks, because this exploration and processing can be divided into small pieces and performed by individual agents. We believe that this idea can be used when exploring networks as well, so they can increase the overall performance (Tan 1993).

Although there are many search algorithms used in graphs or networks (Knuth 1997, Knuth 1998), such as Breadth-First Search (BFS) (Bader 2006, Yoo 2005), Depth-First Search (DFS), Dijkstra and Kruskal, they are all thought to search for values on networks, not to explore them. These algorithms were not thought to be used either for very large and highly connected networks, for the presence of obstacles or for being used by multiple agents simultaneously. Moreover, some algorithms do not consider the distance between nodes or even a cost to travel through them. Thus, they cannot be used in our scenario. There are still other algorithms based on the Ant Colony approach (Weyns 2007, Claes 2011, Colorni 1991, Dorigo 1992) that might be interesting to this scenario but they do not fully meet our constraints.

We propose a novel method using a MAS architecture to explore a network, where agents can only communicate between them using simple messages, so as to share information and to accomplish the entire discovery. This method guarantees almost full isolation of the agents and no resource sharing. These agents do not need to negotiate, so they are naturally fault safe considering the overall process.

To prove the usefulness of this method, we had developed a simulation with two other methods as a basis for comparison. The application domain for this simulation was the discovery of a new planet by space vehicles of different types. We implemented three types of agents, each of them with a different method (random, non-cooperative and cooperative). The comparison was made using the full exploration mean time for each agent type.

The remainder of this paper is structured as follows. Section 2 discusses on the proposed approach, where we present the problem formalization and our solution method. In Section 3, preliminary results are presented and analyzed. Related work on network exploration are presented in Section 4, whereas in Section 5 we draw conclusions about the proposed method.

2. PROPOSTED APPROACH

The basis for the cooperative discovery approach proposed in this paper is the use of altruistic agents that work together to discover all the network.

The core for this agents is the A* algorithm, whose premises rely on agents featuring two sets of node identifiers. These sets contain the nodes that the agent wishes to explore (SN_{wish}) and those it has already explored ($SN_{explored}$). At the beginning, both SN_{wish} and $SN_{explored}$ are empty. This is the only information that an agent needs to store in its memory. Each agent has memory and only keeps knowledge of its experience and history; thus, despite what it actually knows, an agent is unaware of the remaining network and of what other agents know. Therefore, each agent is independent from other agents, which guarantees that there is no memory/resource sharing and no breaking points all over its activity.

Since an agent is isolated, as mentioned before, it can only send and receive messages to and from other agents. These messages can be directed to one agent in particular, or be broadcast across the network to all other agents. Thus, there must exist a messaging service to provide this requirement. There are only three types of messages that the agents can send:

1. Inform all other agents that it is visiting one node ($Message_{inform}$); this is a broadcast message;
2. Ask all agents what is the content of their SN_{wish} set ($Message_{ask\text{-}help}$); this is a broadcast message;
3. Reply to a specific agent that previously has sent $Message_{ask\text{-}help}$ the content of the SN_{wish} set ($Message_{response\text{-}help}$).

These messages will be used on specific situations and will be explained with the algorithm. Each agent has its own mail box.

The visibility of each agent on each node is limited to the node itself and to the node's adjacencies, which means that the sole information it can have are nodes' identifier and whether all connected nodes are dark nodes or not. That means that an agent can "see" which connected nodes can be explored prior to moving to them. This is particularly useful, e.g. when an agent is exploring a map and there is a wall on his path.

Figure 1 shows the state diagram for this method, representing an algorithm overview. More specifically, at each step, an agent does the following (considering that the agent is inside a node):

1. Read mail box and process all messages (explained ahead);
2. Extracts the needed node information;
3. Sends a broadcast message to all other agents ($Message_{inform}$) saying that the node being analyzed is already explored, so other agents can remove this node from theirs SN_{wish}, if it exists, and add it to $SN_{explored}$;
4. Adds that node identifier to $SN_{explored}$;
5. From the neighbor nodes, chooses the ones that are not dark nodes, and adds them to SN_{wish}, if not present in $SN_{explored}$;
6. If SN_{wish} is not empty, chooses the nearest node from SN_{wish} (if there are more than one, use random choice) and goes to it (then, return to step 2);
7. If SN_{wish} is empty, sends a broadcast message to all other agents ($Message_{ask\text{-}help}$), asking them to send their SN_{wish} contents, so that it can have new nodes to explore.

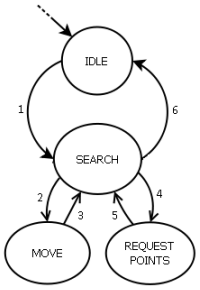

Figure 1: Agent State Diagram

There is no defined algorithm to move across nodes; one can use any heuristics, depending on the problem. When moving between nodes, it executes steps 2, 3 and 4. Thus, the agent maintains its goal and builds on the work. Step 6 can be performed using a heuristic utility that tries to maximize any goal. However, this is optional.

The mail box reading process uses the following rules, according to the message received:

- If $Message_{inform}$: reads the node identifier from the message. Removes that node from SN_{wish}, if exists. Add that node to $SN_{explored}$;
- If $Message_{ask\text{-}help}$: sends the content of $Message_{wish}$ to the agent that sent this message;
- If $Message_{response\text{-}help}$: copy the received nodes identifiers to SN_{wish}.

When SN_{wish} is empty and there is no response for $Message_{ask\text{-}help}$ ($Message_{response\text{-}help}$ messages), the work is done. That means there is no known nodes to further explore, and the job is complete.

This method guarantees that the entire network is discovered, because each agent contributes, at each step, to this process. Despite these agents are isolated and do not explicitly share information about their knowledge,

they can guarantee, as a whole, that the network is fully explored. At an earlier stage, each agent tries to explore all the nodes that are in their SN_{wish} set; later on, when their job queue is empty, they try to collaborate with other agents and do their work.

Each agent is independent and acts by itself, so there is no critical point in the whole process (i.e., a coordinator base) - the exploration is just cooperative, and there is no need to have a centralized coordinator. That means there are no breaking points in the exploration, which makes them naturally fault safe and act as reactive agents. In other words, if one agent fails, the full exploration is not compromised. According to Stan Franklin and Art Graesser, these agents are autonomous, goal-oriented, temporally continuous and communicative (Graesser 1996).

3. PRELIMINARY RESULTS AND DISCUSSION

To test the proposed method, we developed a simulated case study, using the *REPAST*[1] simulation framework. The chosen scenario was the exploration of an unknown planet by some space rovers (space exploration vehicles/robots).

3.1. Simulation setup

The planet surface was represented in a 2D perspective, using the torus concept. The surface had many obstacles, representing the "dark nodes". Each node (position), represented by the coordinate pair (X, Y), was connected to the closest eight near cells (degree equals 8) using the Moore's neighborhood concept. There was a starting point, called "base", where all the robots were at the beginning of the simulation, and 250 obstacle points (representing walls or rocks that the robot cannot cross), distributed in different forms, in order to provide a complete experience. Figure 2 depicts the map (size was 50x50) used for this experiment.

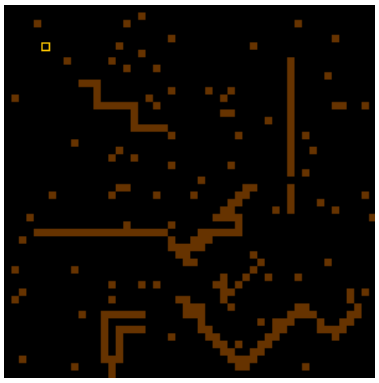

Figure 2: map with obstacles (brown points) and one base (yellow square)

In order to establish a basis for comparison of the proposed method, we developed three types of agents: random agents (A_R), non-cooperative (selfish) agents (A_{NC}) and cooperative (altruistic) agents (A_C).

[1] http://repast.sourceforge.net

A_R agents, at each iteration, choose the next node randomly from the connected nodes. They have no knowledge or memory. This is the simplest agent.

A_{NC} agents use pre-determined blocks to explore (areas), previously calculated, and they follow it, avoiding obstacles when found. Although the division of this scenario (network) in blocks is based on the map size, this division can be done by other heuristic that do not need that information. Moreover, the block-division algorithm does not know the map content, either. The reason for choosing this algorithm was its simplicity to be implemented; nonetheless, it could be any other algorithm as long as it is non-cooperative-based.

The third type of agent, A_C, uses the methodology presented in this paper, whose performance we want to compare with the former two.

At each simulation step ('tick' in *REPAST* framework), the order of execution of each agent was random. In order to compare all agents, we measured the number of explored positions on the map over time (ticks) for each agent type.

We did seven tests, with 10, 15, 20, 25, 30, 35 and 40 agents of each type. For statistical significance, we run each experiment 160 times; the average of the ending time was calculated, and the less (lower values) the better.

3.2. Results and Discussion

In some of our tests, our A_R (random) got results over 7000 ticks in all rounds, so it will not be considered in the analysis. This happens because this map size is too big for this number of agents, so they need more time to explore everything.

Table 1 and Table 2 show the results obtained in the experiments.

Table 1: Execution ticks averages

Nr. of Agents	A_R	A_{NC}	A_C
10	N/A	878.98	484.16
15	N/A	665.64	408.56
20	5733.76	537.66	412.18
25	4132.44	440.61	413.36
30	3948.95	479.74	371.99
35	3306.01	425.40	388.91
40	3038.31	425.30	386.26

In Table 1 we present the ticks average of complete exploration for all the algorithms. As expected, A_R gets better as there are more agents to explore this network. However, those results are much higher than those corresponding to A_{NC} and A_C. A_C got in all tests always a better result than A_{NC}.

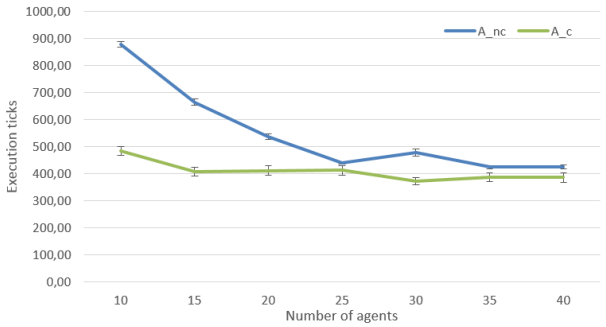

Figure 3: Execution ticks average for A_{NC} (blue line) and A_C (green line), with confidence range level of 95%.

Table 1 also shows that the performance obtained in the A_C method has a low variance. In other words, the difference between the best and worst value for A_{NC} was 453.68 ticks and for A_C was just 112.6 ticks.

Table 2 shows the precision of the mean obtained in Table 1, with a confidence level of 95%, calculated with equation 1 for experimental results (JCGM 2008, Carvalho, et al. 2012, Simões 2008). Figure 3 shows the execution tick average for A_{NC} and A_C, for all experiments, with precision range indicated on Table 2.

$$\sigma_{m(95\%)} = 2\sigma_m = 2\frac{\sigma}{\sqrt{N}} \approx 2\sqrt{\frac{\sum(x_i-\bar{x})^2}{n(n-1)}} \quad (1)$$

Table 2: Standard Deviation of the Mean (confidence level of 95%) in ticks and corresponding percentages.

Nr. of Agents	A_R	A_{NC}	A_C
10	N/A	9.92 (1.1%)	15.97 (3.3%)
15	N/A	12.61 (1.9%)	16.76 (4.1%)
20	604.01 (10.5%)	10.01 (1.9%)	18.07 (4.4%)
25	304.80 (7.4%)	5.23 (1.2%)	17.87 (4.3%)
30	382.79 (9.7%)	13.07 (2.7%)	13.55 (3.6%)
35	289.04 (8.7%)	5.61 (1.3%)	15.96 (4.1%)
40	364.30 (12.0%)	7.12 (1.7%)	17.09 (4.4%)

As shown in Table 2, the confidence level for the mean of the A_C method is around 4%, which represents a good precision value. Applying the range of $\bar{x} \pm \sigma_{m(95\%)}$ for all results, we can prove that A_C is still always better (lower) than A_{NC}.

Figure 4 and Figure 5 show two simulation executions, indicating the number of explored positions *vs.* tick time, for each type of agent. We can see that the evolution of the methods are different. A_C proves to be linear through time, dealing well with the obstacles found.

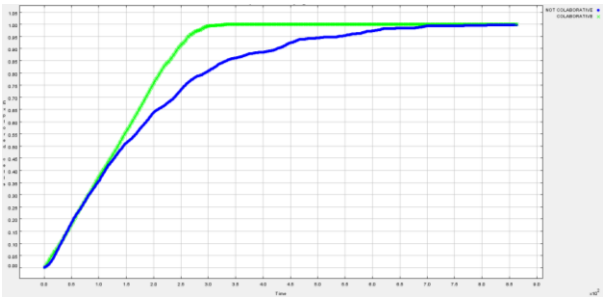

Figure 4: Execution example with 10 agents. Explored nodes *vs.* ticks. Blue line - A_{NC}; green line - A_C.

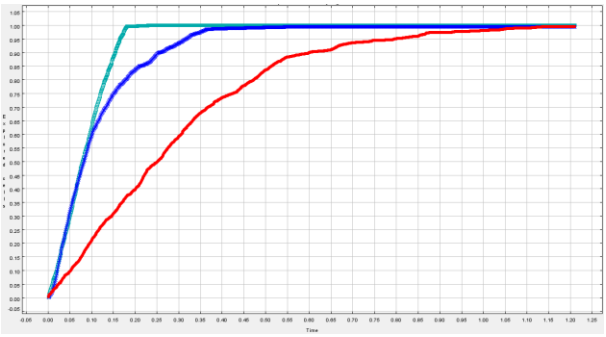

Figure 5: Execution example for 20 agents. Explored nodes *vs.* ticks. Red line - A_R; blue line - A_{NC}; green line - A_C.

These results shows that our proposed method is always better than the other two used for comparison. This happens in result for the existence of dark nodes inside the network, which oblige agents to cope with them. In these experiments, our network has a lot of combinations of dark nodes (horizontal, vertical and diagonal walls, "S-shapped" walls and isolated obstacles). Although we can use some simple and reactive obstacle avoidance algorithm such as Bug1 or Bug2 (Ribeiro 2005, Stepanov 1990, V. a. Lumelsky 1990, Choset 2005), they still have to cope with them, so agents will have to overcome these obstacles. In our experiment, A_{NC} had predefined areas to explore; sometimes, agents needed to abandon their "working area" to avoid some obstacles, which proves to be a high cost to the overall exploration process.

Figure 6 shows the states evolution of our A_C in a simulation with 40 agents, as defined in Figure 1. In the initial phase there are more agents exploring than moving. In the middle of the execution, search and moving states are equal and then, when the network starts to be fully explored (approximately 90% at 100 ticks), all agents start to move and request points. That means only approximately 10% of the network is still not explored and it consumes almost the same time to explore it as in the first 90%. This difference is a result of the traveling time of agents to explore some missing points across the network. Figure 7 can be used to check the evolution of this experience.

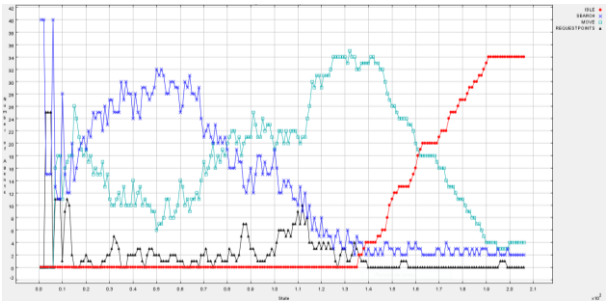

Figure 6: Collaborative agents' state for an execution example with 40 agents. IDLE - red line; SEARCH - blue line; MOVE - green line; REQUEST POINTS - dark line.

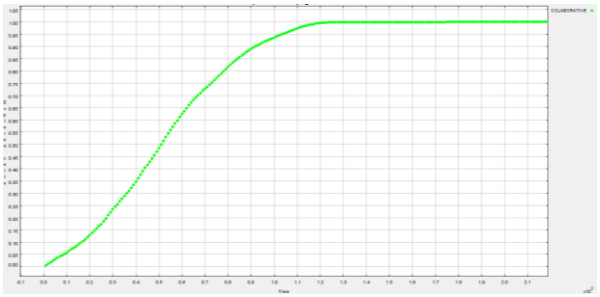

Figure 7: Execution time with 40 A_C (green line)

The fact that agents A_C do not need to negotiate between them has the consequence that there is no need for a synchronizing phase. Even without sharing information about the network, all agents proved effective when coping (on their own only) with unpredictable obstacles during exploration.

4. RELATED WORK

Network exploration and network search in computer science and mathematics is not a recent topic. There are many algorithms that try to optimize the searching mechanism on a network, to find the best path between two nodes or even to explore its metadata (Leiserson 2001). There are also other approaches based on the Ant Colony concept that use the environment as a communication medium, to perform a cooperative exploration. Furthermore, some authors have used parallel processing in their algorithms, as well as MAS architectures to perform their tasks using different approaches such as decentralized search (Zhang 2005).

However, to the best of the authors' knowledge there are no such methods or algorithms that meet all constraints for the type of networks here presented. Consequently, this work was not based on any previously published works.

5. CONCLUSIONS

In this paper, we presented a novel method that performs a cooperative exploration of an unknown network with obstacles using a MAS-based approach, in order to explore it as quickly as possible. Our concept relied on isolated and altruist agents that communicate through simple messages in order to exchange some information.

We performed a basis for comparison with other two non-cooperative methods through simulation. Our method always outperformed the non-cooperative ones whenever dark nodes were present on the network. The precision of our results was about 4%, for a confidence level of 95%. Thus, results show that our agents have a higher performance when there are a few agents running the exploration algorithm for a given network, compared with the other methods. That means cooperative agents can produce better results when dealing with large networks. This is also true even though a dark node is an articulation point in the network, for instance, when the associated graph representing the network will be disconnected.

As our agents will explore the whole bunch of nodes within range, we only need to ensure that the entering points are normally distributed over the network to guarantee total coverage. Moreover, our agents are kept isolated, with no resource sharing, and there is no need of negotiation, so there is no fault point in the whole process. Also, they do not try to foresee the future, so they are very reactive according to the environment. Their behavior relies on the information that they have at that very moment.

Further improvements may include, for instance, developing better communication and path calculation algorithms, to optimize and minimize the length of the path traveled by the agent when it finds a dark node or needs to travel between nodes. Thus, the future goal is to minimize the wasting time in any journey and use it for discovery purposes.

Depending on the scenarios, other improvements or modifications can be made, such as the existence of a "refresh" time of the nodes or the need for a specific agent to analyze a nodes' set. Our method is flexible in many aspects in order to adapt to existing constraints.

The knowledge of each agent is relatively exclusive to its execution but, as a whole, agents learn and behave in an implicit way. This method guarantees that the entire network is explored to the end, despite the network size or characteristics. Thus, this method can be applied either to extract nodes metadata or to analyze their connections using a cooperative MAS approach.

ACKNOWLEDGMENTS

The authors are indebted to José Pedro Silva for helpful discussions about some of the ideas presented in this work at its earlier stages.

REFERENCES

Bader, David A and Madduri, Kamesh. "Designing multithreaded algorithms for breadth-first search and st-connectivity on the Cray MTA-2." *Parallel Processing, 2006. ICPP 2006. International Conference on.* IEEE, 2006. 523--530.

Carvalho, Paulo Simeão, Adriano Sampaio Sousa, João Paiva, e António José Ferreira. *Ensino Experimental*

das Ciências. Um guia para professores do ensino secundário. Física e Química. U.Porto Editorial, 2012.

Choset, H. and Lynch, K.M. and Hutchinson, S. and Kantor, G. and Burgard, W. and Kavraki, L.E. and Thrun, S. *Principles of robot motion: theory, algorithms, and implementations.* MIT press, 2005.

Claes, Rutger and Holvoet, Tom and Weyns, Danny. "A decentralized approach for anticipatory vehicle routing using delegate multiagent systems." *Intelligent Transportation Systems, IEEE Transactions on.* IEEE, 2011. 364--373.

Colorni, Alberto and Dorigo, Marco and Maniezzo, Vittorio and others. "Distributed optimization by ant colonies." *Proceedings of the First European Conference on Artificial Life.* Paris, France, 1991. 134--142.

Dorigo, Marco. "Optimization, learning and natural algorithms." *Ph. D. Thesis, Politecnico di Milano*, 1992.

Graesser, Stan Franklin and Art. "Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents." *Third International Workshop on Agent Theories, Architectures, and Languages.* Springer-Verlag, 1996. 21--35.

JCGM. *Evaluation of measurement data - Guide to the expression of uncertainty in measurement.* Sèvres: BIPM, 2008.

Knuth, Donald E. *The Art of Computer Programming.* Vol. 3. Harlow: Addison-Wesley, 1998.

—. *The Art Of Computer Programming.* 3. Vol. 1. Boston: Addison-Wesley, 1997.

Leiserson, Charles E., Rivest, Ronald L., Stein, Clifford and Cormen, Thomas H. *Introduction to Algorithms.* The MIT press, 2001.

Lumelsky, V.J. and Skewis, T. "Incorporating range sensing in the robot navigation function." Editado por IEEE. *Systems, Man and Cybernetics, IEEE Transactions on* 20, nº 5 (1990): 1058-1069.

Parker, Dawn C and Manson, Steven M and Janssen, Marco A and Hoffmann, Matthew J and Deadman, Peter. "Multi-agent systems for the simulation of land-use and land-cover change: a review." *Annals of the Association of American Geographers* (Taylor & Francis) 93 (2003): 314--337.

Ribeiro, Maria Isabel. "Obstacle avoidance." *Instituto de Sistemas e Robótica, Instituto Superio Técnico* (Citeseer), 2005: 1.

Simões, JAM, Castanho, MARB, Lampreia, IMS, Santos, FJV, Castro, CAN, Norberto, MF, Pamplona, MT, Mira, L, Meireles, MM. *Guia do Laboratório de Química e Bioquímica.* Lisboa: Lidel, 2008.

Stepanov, V. Lumelsky and. "Path-planning srategies for a point mobile automaton amidst unknown obstacles of arbitrary shape." *Autonomous Robots Vehicles* (Springer), 1990: 1058-1068.

Tan, M. *Multi-agent reinforcement learning: Independent vs. cooperative agents.* Vol. 337, em *Proceedings of the tenth international conference on machine learning*. Amherst, MA, 1993.

Weyns, Danny and Holvoet, Tom and Helleboogh, Alexander. "Anticipatory vehicle routing using delegate multi-agent systems." *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*. IEEE, 2007. 87--93.

Yoo, Andy and Chow, Edmond and Henderson, Keith and McLendon, William and Hendrickson, Bruce and Catalyurek, Umit. "A scalable distributed parallel breadth-first search algorithm on BlueGene/L." *Supercomputing, 2005. Proceedings of the ACM/IEEE SC 2005 Conference.* IEEE, 2005. 25--25.

Zhang, Jun and Ackerman, Mark S. "Searching for expertise in social networks: a simulation of potential strategies." *Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work.* ACM, 2005. 71--80.

AUTHORS BIOGRAPHY

Pedro Simeão Carvalho is graduated in Informatics and Computing Engineering where he took his master degree; he is now working at software developer TLANTIC SI enterprise, in mobile systems department.

Rosaldo J. F. Rossetti is an assistant professor at Faculty of Engineering, University of Porto, and member of the Artificial Intelligence and Computer Science Laboratory.

Ana Paula Rocha is an assistant professor at Faculty of Engineering, University of Porto, and member of the Artificial Intelligence and Computer Science Laboratory.

Eugénio C. Oliveira is a full professor at Faculty of Engineering, University of Porto. Prof. Oliveira is the head of the Artificial Intelligence and Computer Science Laboratory, and his research interests are mainly in the field of Multi-Agent Systems and their applications