COMPACT NODE COUNTING EXPLORATION ALGORITHM

Bogdan-Florin FLOREA[1], Ovidiu GRIGORE[2], Mihai DATCU[3]

In this paper, we propose an exploration algorithm based on a modification of the original node counting algorithm which provides compact spatial exploration capabilities for reflex agents and it is capable of multi-agent operation by using a pheromone map as information storage and exchange medium. The algorithm proposed in this paper outperforms in terms of cumulative path length all other popular exploration algorithms based on reflex agents that we included in the comparison.

Keywords: autonomous agents, cooperative systems, intelligent agents, mobile agents, reflex agents, spatial exploration

1. Introduction

There has been interest in the scientific community for efficient self-healing and self-organizing spatial exploration techniques that can be used for spatial exploration, with applications both on Earth and in extraterrestrial environments. The exploration algorithm that we propose in this paper builds on the NCA (Node Counting Algorithm) [1] by using a different cost structure for changing its behavior at the exploration frontier in order to obtain a compact exploration pattern, which favors the exploration of unexplored cells which are adjacent to the already explored cells.

The compact exploration approach is interesting for the spatial exploration of terrains of unknown and potentially very large size, which are typically encountered in extraterrestrial exploration. By using an exploration algorithm which produces a compact explored area, it is possible to study the explored area more thoroughly and to get relevant information about the explored environment.

The exploration algorithm proposed in this paper can be used for building a resilient self-organizing and self-healing multi-agent exploration system.

[1] University POLITEHNICA of Bucharest, Bucharest, Romania, e-mail: bogdan.florea@ai.pub.ro
[2] University POLITEHNICA of Bucharest, Bucharest, Romania, e-mail: ovidiu.grigore@ai.pub.ro
[3] University POLITEHNICA of Bucharest, Bucharest, Romania, and Deutsches Zentrum für Luft und Raumfahrt, Oberpfaffenhofen, Weßling 82234, Germany, e-mail: mihai.datcu@dlr.de

2. Theory

2.1 The collaborative exploration problem

For the purpose of our research we have modelled the terrain as an 8-connected discrete grid (*V, E*) as shown in Fig. 1. For each intelligent agent, we have considered a visibility horizon (sensor range) of one cell as shown in Fig. 2.

We have used a pheromone map as an information storage and communication medium for the agents. For the purpose of this research, we have considered that a communication and localization system is already available for the agents and that the pheromone map and the discovered obstacle map is shared between the agents in the multi-agent exploration scenario.

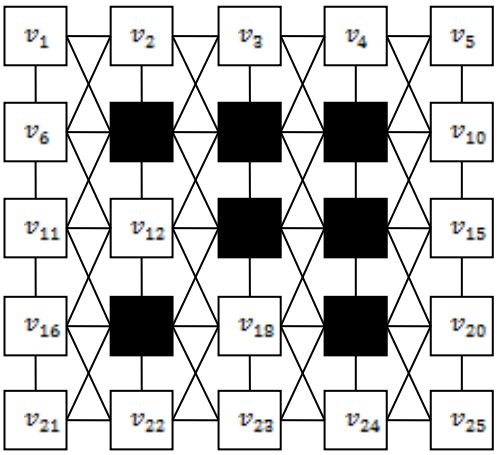

Fig. 1. A 5x5 terrain modelled as an 8-connected discrete grid. The vertices corresponding to obstacles are coloured in black

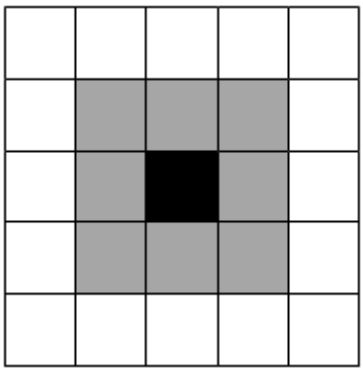

Fig. 2. The visibility horizon of an agent (the agent is marked with black colour)

The single agent exploration problem for a finite size terrain consists into finding a path that visits all the accessible cells with a cost as low as possible:

$$P = \underset{\substack{(v_0, v_1, v_2 \ldots v_{n-1}) \\ v_i \downarrow v_{i-1}, i=\overline{1,n-1} \\ f_a(v_i)=1, i=\overline{1,n-1}}}{\operatorname{argmin}} \sum_{i=1}^{n-1} f_c\left(e_{i-1,i}\right), \tag{1}$$

where:

$P = (v_0, v_1, v_2 \cdots v_{n-1}) \in V^n$ is the path

$v \downarrow u$ means that v is adjacent to u

$e_{i-1,i} = \{v_{i-1}, v_i\}$ is the edge connecting the vertex v_{i-1} to v_i

$f_c\left(e_{i-1,i}\right)$ is the cost function

$f_a(v) = \begin{cases} 1, & \neg f_o(v) \wedge (v_0 \text{ connected to } v) \\ 0, & \text{otherwise} \end{cases}$

$f_o(v) = \begin{cases} 1, & \text{if } v \text{ is an obstacle} \\ 0, & \text{otherwise} \end{cases}$.

The collaborative exploration problem for finite terrains consists into finding a set of path for the intelligent agents so that the cumulative cost of the paths is as low as possible:

$$\left(P_1, P_2 \ldots P_N\right) = \underset{\substack{(P_1, P_2 \cdots P_N) \\ \{v \in V \mid f_a(v)=1\} \subseteq Vertices(P_1, P_2 \cdots P_N)}}{\operatorname{argmin}} \sum_{i=1}^{N} f_{pc}(P_i), \tag{2}$$

where:

$f_{pc}(P) = \sum_{i=1}^{n-1} f_c\left(e_{i-1,i}\right)$ is the cumulative cost of the path P

P_i is the path followed by the i^{th} agent from the team

$Vertices(P_1, P_2 \cdots P_N) = \bigcup_{i=1}^{N} Vertices(P_i)$

$Vertices(P_i)$ is the set of vertices visited by the path P_i.

Since the algorithm presented in this paper is not optimal, for our formalism we have considered the "argmin" operator as a best-effort search for a solution, which returns a solution, but not necessarily the optimal one.

Each agent is capable to perform computations and to take actions by taking into account the information from the sensors over the visibility horizon of one cell and being able to sense the obstacles (inaccessible cells) and the values from the pheromone map.

When exploring unknown environments, the obstacle information acquired from the sensors is used to build a map of the environment, by sensing and recording the obstacle cells.

2.2 The proposed algorithm

2.2.1 Using a pheromone map to avoid revisiting cells

Similarly to the Node Counting Algorithm [1], we have used a pheromone map, in which the entry corresponding to each cell from the grid is incremented each time the agent visits that location.

We have used the following cost in order to penalize repeated visits of the same cells:

$$c_1(e_{ij}) = pheromoneMap(v_j), \tag{3}$$

where:

e_{ij} is the edge connecting the vertex v_i to the vertex v_j

$pheromoneMap(v_j)$ is the pheromone value corresponding to the vertex v_j.

2.2.2 Modeling the compactness constraints as costs

For the purpose of this algorithm, we have modelled the compactness cost as a local cost which penalizes the actions that lead to a lower compactness of the explored area.

We have defined the compactness cost as follows:

$$c_2(e_{ij}) = \sum_{\substack{u \in V \\ v_j \downarrow u}} (2 - f_e(u) - 2 f_o(u)), \tag{4}$$

where:

$$f_e(v) = \begin{cases} 1, & \text{if vertex } v \text{ is already explored} \\ 0, & \text{otherwise} \end{cases}.$$

This cost also penalizes the agent for exploring areas that are not adjacent to obstacles, creating therefore an affinity for expanding the exploration frontier towards the areas close to obstacles.

2.2.3 The exploring algorithm

The exploration algorithm is based on reflex agents, each agent choosing the successor cell that it is going to visit according to the following rule:

$$succ(v_i) = \underset{v_j}{\operatorname{argmin}} \; f_c(e_{ij}), \tag{5}$$

where:

$$f_c(e_{ij}) = \begin{cases} c_1(e_{ij}), & f_e(v_j) = 1 \\ c_2(e_{ij}), & f_e(v_j) = 0 \end{cases}.$$

Each reflex agent works according to the following algorithm:

Initialize *pheromoneMap* with zeros

Initialize *successor* with the starting position

While exploration not complete **do**:

$$successor := succ(successor)$$

Mark *successor* as visited

Increment *pheromoneMap*(*successor*)

End

3. Results and discussion

For the comparative analysis, we have used the shape factor as a global measure of the compactness of the explored area:

$$sf = \frac{4 \cdot \pi \cdot A}{P^2}, \tag{6}$$

where:

A is the area of the explored region (considering the holes that surround obstacles to be filled)

P is the perimeter of the explored region (the frontier).

The value of the shape factor indicates the compactness of the explored region. For a circular region it has a value of one and it decreases as the shape deviates from circular form.

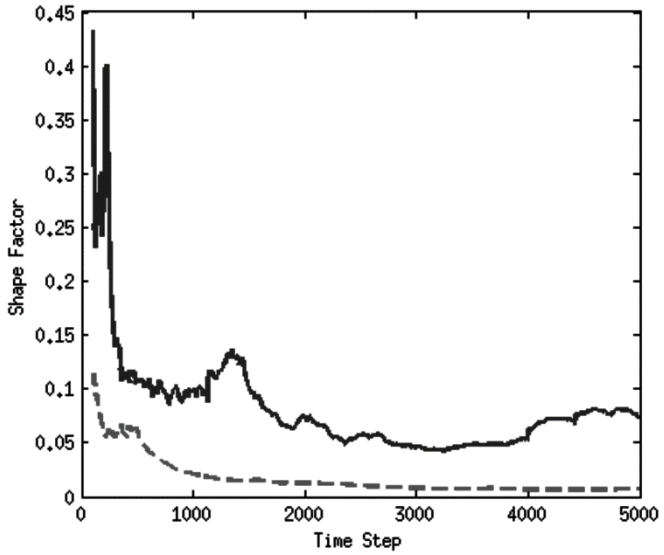

Fig. 3. Shape factor comparison. (The shape factor for the compact node counting algorithm is represented with continuous line and the shape factor for the original node counting algorithm is represented with dashed line)

From Fig. 3, it can be seen that in contrast to the original node counting algorithm, the shape factor of the explored area obtained using the algorithm proposed in this paper deteriorates significantly less as the exploration continues over time. In the following tables we have also shown that our approach oriented towards compactness brings a speed benefit.

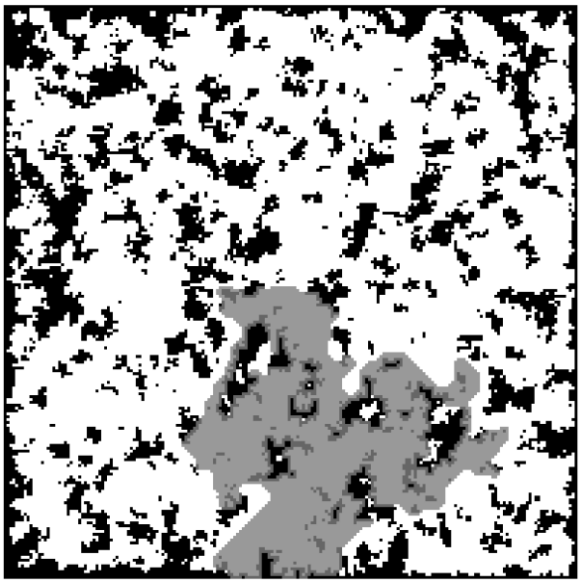

Fig. 4. Typical exploration pattern of the compact node counting exploration algorithm

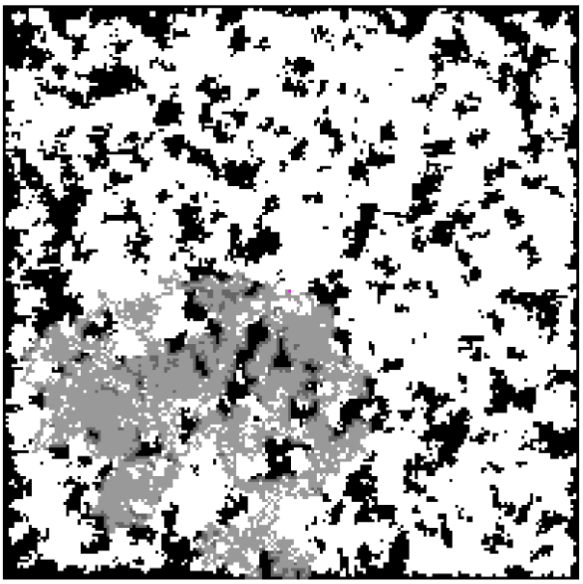

Fig. 5. Typical exploration pattern of the original node counting exploration algorithm

In Fig. 4 and 5, it can be observed qualitatively that there is a significant difference in terms of the compactness of the explored area between the algorithm proposed by us and the original node counting algorithm, resulting into a lower number of "holes" in the explored area pattern. Our investigation has shown that

this behavior, along with a slight affinity for expanding the exploration frontier around the obstacles as imposed by the costs defined at the exploration frontier, also brings an improvement in terms of the exploration speed (measures as the cumulative path length).

Besides the shape factor, we have also investigated the exploration speed of the algorithm that we have proposed in this paper, comparing it with several exploration algorithms from the literature. We have compared it with the original node counting algorithm [1], with the exploration algorithm based on Thrun's rule [7], with vertex ant walk [6] and with learning real-time A* algorithm with a look-ahead of one cell.

Table 1

Exploration speed comparison (10000 runs with 1 agent on different 30x30 maps)

Algorithm	Step count (average)	Step count (standard deviation)
Current algorithm	544.99	309.87
Node counting algorithm	674.20	351.72
Thrun's rule	679.89	354.73
Vertex ant walk	749.14	410.93
Learning real-time A*	675.52	354.00

Table 2

Exploration speed comparison (10000 runs with 3 agents on different 30x30 maps)

Algorithm	Step count (average)	Step count (standard deviation)
Current algorithm	597.05	314.40
Node counting algorithm	718.35	361.64
Thrun's rule	721.12	362.35
Vertex ant walk	821.77	436.14
Learning real-time A*	720.15	364.00

Table 3

Exploration speed comparison (10000 runs with 6 agents on different 30x30 maps)

Algorithm	Step count (average)	Step count (standard deviation)
Current algorithm	698.69	352.94
Node counting algorithm	824.44	426.74
Thrun's rule	812.27	409.16
Vertex ant walk	966.90	503.45
Learning real-time A*	811.71	411.04

From the table 1, 2 and 3 it can be observed that the proposed algorithm is faster than the original node counting algorithm and that it outperforms all other

algorithms included in this comparison in terms of speed. These results were obtained by running the exploration algorithms on 10000 randomly generated maps of size 30x30.

In order to have a more through comparison, we have also run the comparison on a set of 1000 randomly generated maps of size 100x100 and we have found that the algorithm proposed in this paper consistently outperforms the other algorithms.

Table 4

Exploration speed comparison (1000 runs with 1 agent on different 100x100 maps)

Algorithm	Step count (average)	Step count (standard deviation)
Current algorithm	13397.70	8381.30
Node counting algorithm	15652.30	8991.10
Thrun's rule	16009.70	9154.40
Vertex ant walk	23649.10	14808.80
Learning real-time A*	15502.40	9414.80

Table 5

Exploration speed comparison (1000 runs with 3 agents on different 100x100 maps)

Algorithm	Step count (average)	Step count (standard deviation)
Current algorithm	15765.60	10206.70
Node counting algorithm	17829.30	10942.50
Thrun's rule	18175.90	12040.80
Vertex ant walk	28781.20	18977.30
Learning real-time A*	18554.80	11835.60

Table 6

Exploration speed comparison (1000 runs with 6 agents on different 100x100 maps)

Algorithm	Step count (average)	Step count (standard deviation)
Current algorithm	19217.90	15631.80
Node counting algorithm	21037.70	15010.00
Thrun's rule	22478.60	17254.10
Vertex ant walk	35717.70	24838.60
Learning real-time A*	21736.80	16201.60

From table 4, 5 and 6, it can be observed that the algorithm proposed in this paper outperforms the other algorithms included in the comparison on the big maps data set, in both the single agent and multi-agent exploration scenarios. This shows that the performance advantage over the other algorithms is consistent and not limited only to particular scenarios.

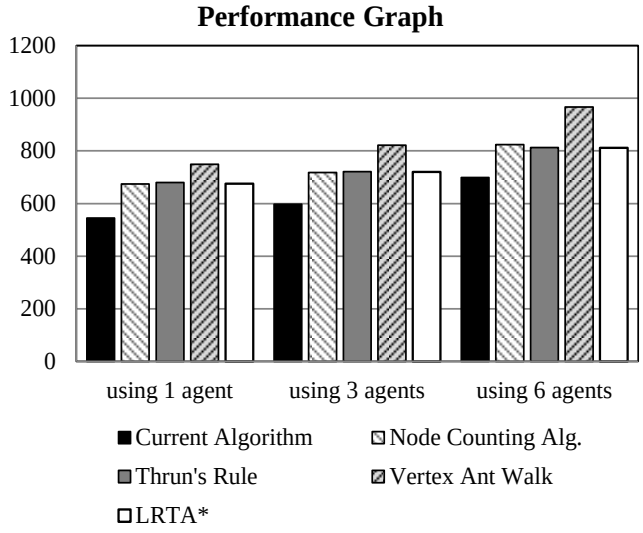

Fig. 6. Performance comparison graph plotted using the data presented in table 1, 2 and 3 (lower is better)

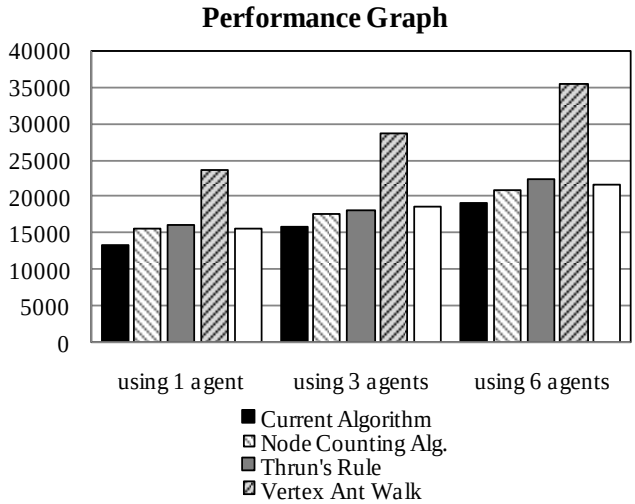

Fig. 7. Performance comparison graph plotted using the data presented in table 4, 5 and 6 (lower is better)

In Fig. 6 and 7 we have presented an overview of the exploration performance of the algorithm proposed in this paper, compared with several other exploration algorithms from the literature. From these figures it can be observed that the algorithm presented in this paper outperformed the other algorithms in all scenarios that we have analysed.Since the analysis has been performed on a large number of maps and on maps of different dimensions, we conclude that our results are statistically relevant. Although there is some overhead in the multi-agent scenarios for parallel exploration, this parallel overhead can be observed for all of the algorithms included in the comparison and it is not specific to our algorithm.

These results show that the affinity for expanding the exploration frontier towards areas occupied by obstacles imposed by the structure of the costs that we designed for this algorithm is beneficial in terms of exploration speed.

This approach, combined with the compactness avoids leaving many unexplored gaps in the explored area. The other algorithms that don't have any compactness affinity are prone to leaving "holes" in the explored area, which need to be revisited at a later time, therefore decreasing the efficiency of that exploration algorithms.

4. Conclusions

In this paper, we have introduced an algorithm for compact spatial exploration, based on reflex agents and with low computational requirements which outperformed all other four exploration algorithms from the literature that we included in our comparison.

The compact exploration algorithm introduced by us is capable to keep the compactness of the explored area by using only local costs, which makes it computationally efficient.

We have shown that our exploration approach based on compactness has multiple benefits, including an increase in exploration speed, outperforming all other algorithms included in the comparison, in both single agent and multi-agent exploration scenarios, with a performance gain between 8% and 46%.

In this paper, we have shown that it is possible to obtain a compact exploration pattern while also benefiting from a significant performance gain in terms of cumulative path length. This contribution is important for the field of artificial intelligence and robotics, because it can be incorporated in autonomous robots capable of intelligent spatial exploration with low computational requirements. This is important for a wide range of applications, ranging from intelligent extraterrestrial spatial exploration to commercial applications like autonomous vacuum cleaners.

Acknowledgements

The work has been funded by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Ministry of European Funds through the Financial Agreement POSDRU/159/1.5/S/132397.

R E F E R E N C E S

[1]. *A. Pirzadeh, W. Snyder*, "A unified solution to coverage and search in explored and unexplored terrains using indirect control", in Proceedings of the International Conference on Robotics and Automation, 1990, pp. 2113–2119.

[2]. *Sven Koenig, Yaxin Liu,* Terrain Coverage with Ant Robots: A Simulation Study, AGENTS'01, Montreal, 2011.

[3]. *Sven Koenig, Reid G. Simmons*, Easy and Hard Testbeds for Real-Time Search Algorithms, AAAI-96 Proceedings, 1996.

[4]. *Richard E. Korf*, "Real-Time Heuristic Search", in Artificial Intelligence, **vol. 42**, 1990, pp. 189–211.

[5]. *Sven Koenig*, "Agent Centered Search", in Artificial Intelligence, **vol. 22**, no. 4, 2001, pp. 109–131.

[6]. *Israel A. Wagner, Michael Lindenbaum, Alfred M. Bruckstein*, "On-Line Graph Searching by a Smell-Oriented Vertex Process", in AAAI Technical Report WS-97-10, 1997.

[7]. *S. Thrun*, Efficient Exploration In Reinforcement Learning, Technical Report CMU-CS-92-102, School of Computer Science, Carnegie-Mellon University, Pittsburgh(Pennsylvania), 1992.

[8]. *M. Baglietto, M. Paolucci, L. Scardovi, R. Zoppoli,* Information-Based Multi-Agent Exploration.

[9]. *P. E. Hart, N. J. Nilsson, B. Raphael*, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", in IEEE Transactions on Systems Science and Cybernetics, **vol. SSC-4**, no. 2, 1968, pp. 100–107.

[10]. *J. Ota*, "Multi-agent robot systems as distributed autonomous systems", in Advanced Engineering Informatics, **vol. 20**, 2006, pp. 59–70.

[11]. *R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. Younes*, "Coordination for multi-robot exploration and mapping", in Proc. of the National Conference on Artificial Intelligence (AAAI), 2000, pp. 851–858.

[12]. *M. Mataric and G. Sukhatme*, "Task-allocation and coordination of multiple robots for planetary exploration", in Proc. of the Int. Conf. on Advanced Robotics, 2001, pp. 61–70.

[13]. *W. Burgard, D. Fox, M. Moors, R. Simmons, and S. Thrun*, Collaborative multi-robot exploration, 2000.

[14]. *I. A. Wagner, M. Lindenbaum, and A. M. Bruckstein*, "Distributed covering by ant-robots using evaporating traces", in IEEE Transactions on Robotics and Automation, **vol.15**, no. 5, 1999, pp. 918–933.