Revue Roumaine des Sciences Techniques – Électrotechnique et Énergetique
Vol. **62**, *1*, pp. 87–92, Bucarest, 2017

MULTI-AGENT EXPLORATION BASED ON CONSTRAINTS IMPOSED WITH GRAPH SEARCH ALGORITHMS

BOGDAN-FLORIN FLOREA[1], OVIDIU GRIGORE[1], MIHAI DATCU[1,2]

Key words: Autonomous agents, Cooperative systems, Intelligent robots, Mobile agents.

In this paper we present a constraint based strategy for collaborative spatial exploration. Using multiple agents and a set of soft constraints modeled as costs we have developed a collaborative system where each agent chooses its next state optimally with respect to compactness and distance based rules, which can model real-world soft constraints. Using random successor state reordering, this exploration system is capable to obtain a stochastic behavior, yet keeping the path planning optimality at each time step. The constraints are imposed at each time step by using graph search algorithms for planning the route to determine the next location that should be visited by the intelligent agent.

1. INTRODUCTION

There has been a lot of interest in the scientific community for self-healing and self-configuring multi-agent systems for a multitude of tasks which require redundancy and fault tolerance.

The collaborative exploration system proposed in this paper is based on multiple agents which plan their next move optimally with respect to some costs using graph search algorithms.

The goal has been to design a multi-agent exploration algorithm suitable for the exploration of unknown terrains of very large size.

Our approach has been to design agents with individual intelligence, so that each agent plans its next move by itself using local information provided by its sensors and a global map of the already explored area, which is shared with the other agents. Since each agent is subject to some soft constraints with the aim of obtaining some desirable properties for the exploration strategy, by choosing the path costs wisely and using multiple agents, a collaborative behavior can emerge.

In this paper we show that this technique based on soft constraints and path planning algorithms is suitable for spatial exploration and mapping missions.

2. EXPLORATION PROBLEM

The problem of finding minimum length exploration tours for offline scenarios (map known in advance) with polygonal maps that contain obstacles is known to be NP hard [1], but some approximations exist. An approximate algorithm for lawn mowing and milling is proposed in [2] which is based on constructing an approximate TSP (traveling salesman problem) tour. In [3], it has been shown by Arora that the Euclidian TSP (traveling salesman problem), which is important for the offline exploration problems has a polynomial time approximation scheme, proposing an algorithm that computes an $(1+1/c)$ approximation to the optimal tour in $O\left(n\cdot(\log n)^{O(c)}\right)$ where n is the number of vertices, c is a positive constant and O is the Landau notation for the algorithmic complexity. Also, O (a) is the algorithmic complexity, the run time grows proportionally to a.

Even without obstacles, finding the optimal solution for the single agent exploration problem in the offline case is a difficult problem. The first polynomial time algorithm for computing the optimum watchman tour (a route so that each point from the polygon and the boundary is visible from at least one point along the route) was proposed by Chin and Ntafos [4] and it has a time complexity of $O\left(n^4\right)$ where n is the number of vertices. This has latter been improved by Tan and Hirata to a time complexity of $O\left(n^4\right)$ [5].

An online exploration algorithm for generating in any polygon a tour that is shorter than 133 times the optimal watchman tour length has been proposed in [6] and it has been further improved to a competitive factor of 26.5 [7]. This approach is based on a new geometric structure called the *angle hull* which is concerned of the visibility of the points of an inner polygon from the outer polygon.

It has also been proven that the competitive complexity of online exploration algorithms for unknown environments is 2 [8]. The problem of creating exploration tours for unknown 4-connected cellular environments with obstacles by using a single agent has been analyzed in [8], where a solution that focuses on short exploration tours is given by the CellExplore algorithm. CellExplore focuses on an exploration strategy that attempts at each step to reserve the right cell for the return path, preferring a left turn over a straight step, over a right turn and walking back along the reserved cells when no forward step is possible. It has been shown that this algorithm explores a polygonal map with $|V|$ cells, $|E|$ edges and H obstacles in $S \le |V| + |E|/2 + H - 3$ steps [8].

Albers, Kursawe and Schuierer have shown that no deterministic or randomized online algorithm can be better than $\Omega\left(\sqrt{n}\right)$ competitive for the exploration of a map with n rectangle obstacles [9]. They have also presented an algorithm for piecemeal exploration (the piecemeal constraint has been defined in [10] and implies that the agent must return to the start vertex from time to time) that explores a grid with an arbitrary number of obstacles using $O\left(|E|\right)$ edge traversals which is optimal [9]. A sub-exponential graph exploration algorithm that achieves an upper bound

[1] "Politehnica" University of Bucharest, Department of Applied Electronics and Information Engineering, 030018, Romania, E-mail: bogdan.florea@ai.pub.ro, ovidiu.grigore@ai.pub.ro, mihai.datcu@dlr.de
[2] Remote Sensing Technology Institute. German Aerospace Center (DLR), Oberpfaffen, Weßling, 82234, Germany

of $d^{O(\log d)}|E|$ where d is the deficiency of the graph has been proposed by Albers and Henzinger in [11].

The collaborative mapping and exploration problem has been studied by Cohen who has proposed a method for using heterogeneous robot teams based on a navigator and several cartographer robots for collective navigation and mapping tasks [12].

Bender and Slonim show that a team of two cooperating robots can learn any strongly connected graph with n indistinguishable nodes in time polynomial in n by using a cooperative algorithm based on random walks. Even more notable is their result showing that a team of two cooperating robots can be exponentially more efficient than a single robot with a constant number of pebbles.

Autonomous robots have been used successfully for geological exploration scenarios, one of the notable examples being the Antarctic meteorite search carried out by the Nomad robot [13].

3. THE COLLABORATIVE EXPLORATION PROBLEM

We have taken into consideration the exploration problem for very large areas, commonly encountered in extraterrestrial exploration. The exploration is usually carried out on vast areas and it is usually preferred that the explored area be compact in order to use it for mapping.

The goal is to explore the area as efficiently as possible given a set of exploration rules or desirable properties which arise from the exploration task (e.g. area compactness as explained above, keeping agents close enough to each other to facilitate communication, etc.).

Without reducing too much the generality of the technique that we propose in this paper, we have considered the area to be represented by a discrete 8-connected grid (V,E) and a visibility horizon of one cell for each agent.

Since we modeled the area as a discrete grid, it is suitable to work with discrete time steps.

We have considered that an idealized localization system and a communication system are already available for the agents, but we have taken into account the range limitations of the real world communication systems and we have modeled them as a soft constraint. During the exploration process, the agents had to create a map of explored area. We considered the mapping in terms of obstacle identification, but the technique can easily be adapted to perform different kinds of mapping, like elevation mapping, terrain density mapping or almost any kind of mapping which is of scientific interest.

For convenient reasoning, we have used a state based approach for modeling the environment.

Formally, the terrain is modeled by the graph (V,E) where V is the set of vertices and E is the set of edges.

On this graph we define several functions.

a) The function which models the obstacles:

$$f_O : V \to \{0,1\}, \tag{1}$$

$$f_O(v) = \begin{cases} 1, & \text{if } v \text{ is an obstacle} \\ 0, & \text{otherwise} \end{cases}. \tag{2}$$

b) The accessibility function, which tells whether a vertex v is accessible from the start vertex v_0 or not:

$$f_a : V \to \{0,1\}, \tag{3}$$

$$f_a(v) = \begin{cases} 1, & \neg f_o(v) \wedge \exists u \in V : \big((u \downarrow v) \wedge f_a(u)\big) \vee (u = v_o) \\ 0, & \text{otherwise} \end{cases}, \tag{4}$$

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

This can also be expressed in terms of connectivity as follows:

$$f_a(v) = \begin{cases} 1, & \neg f_o(v) \wedge \big(v_0 \text{ connected to } v\big) \\ 0, & \text{otherwise} \end{cases}. \tag{5}$$

c) The cost function, which models the cost for moving along an edge from one vertex to another:

$$f_c : E \to \mathbb{R}. \tag{6}$$

The exploration problem consists in finding a path which passes through all accessible nodes while minimizing the cost:

$$P = \underset{\substack{(v_0, v_1, v_2 \ldots v_{n-1}) \\ v_i \downarrow v_{i-1}, i=\overline{1,n-1} \\ f_a(v_i)=1, i=\overline{1,n-1}}}{\operatorname{argmin}} \sum_{i=1}^{n-1} f_c\left(e_{i-1,i}\right), \tag{7}$$

where $P = (v_0, v_1, v_2 \cdots v_{n-1}) \in V^n$ is the path (vertices composing the path are not necessarily consecutive) $e_{i-1,i} = \{v_{i-1}, v_i\}$ is the edge connecting v_{i-1} to v_i.

The collaborative exploration problem consists into finding a path for each agent of the team, so that all the accessible nodes are visited at least once by one of the paths, while minimizing the cumulative cost:

$$\left(P_1, P_2 \ldots P_N\right) = \underset{\substack{(P_1, P_2 \cdots P_N) \\ \{v \in V \mid f_a(v)=1\} \subseteq Vertices(P_1, P_2 \cdots P_N)}}{\operatorname{argmin}} \sum_{i=1}^{N} f_{pc}\left(P_i\right), \tag{8}$$

where:

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

The exploration of all the accessible vertices can be used for building a map of the environment.

For the purpose of our research, each agent has been modeled as an intelligent agent with its own computational capabilities, obstacle detection (within a limited horizon), memory and a communication system for sharing information with the other agents.

The agents feature individual intelligence, each agent being able to compute his next move by himself, by taking into account the information received from its sensors and the information from its memory, part of which is shared with the other agents.

Each agent has obstacle sensors with binary outputs, each of them signaling if there is an obstacle in the adjacent cell from its direction. We performed the simulations using an 8-connected grid, so the simulated agents had 8 obstacle sensors.

Each agent also keeps in its memory a map which represents the already explored cells (explored by itself or by other agents from the team). This map of explored cells is shared between all agents. Similarly, each agent keeps a map with the cells representing obstacles. This obstacle map is built from the sensor inputs and it is also shared between all agents.

We have considered that each agent has a localization and communication system already in place, and being therefore able to compute the distance from itself to each of the other agents.

4. THE PROPOSED ALGORITHM

4.1. MODELING THE SOFT CONSTRAINTS AS COSTS

We have considered a cost composed from the following components:

1. A cost which penalizes the re-exploration of the already visited locations

$$c_1(v) = f_e(v), \tag{9}$$

where:

$$f_e(v) = \begin{cases} 1, & \text{if vertex } v \text{ is already explored} \\ 0, & \text{otherwise} \end{cases}. \tag{10}$$

2. A locally computed cost which penalizes for the non-compactness of the explored area:

$$c_2(v) = \sum_{\substack{u \in V \\ v \downarrow u}} (1 - f_e(u)). \tag{11}$$

The non-compactness cost is higher if a cell has fewer adjacent cells already explored.

3. A cost which penalizes the agent if it gets too close or too far from the other agent

$$c_3(v) = \sum_i f\left(\left(d_p - \|r_v - r_i\|_2\right)^2\right), \tag{12}$$

where:

r_s is the position corresponding to the vertex v;

r_i is the position of the i^{th} agent from the team;

d_p is the preferred distance between agents;

$f(x) = g(x) + g(-x)$;

$g(x) = x^Q \cdot \sigma(x - R)$;

$\sigma(x)$ is the Heaviside step function;

R is the radius inside which this cost is zero.

Q controls how steep this cost rises with the deviation from the preferred distance

The total cost for exploring a state can be expressed as follows:

$$c(v) = \sum_i w_i \cdot c_{i(v)}, \tag{13}$$

where $w_i \geq 0$ are constant weights for each term of the cost function.

The higher the value of the parameter w_1, the higher is the weight of the cost for exploring again the same locations from the grid. Similarly, increasing the parameter w_2 increases the cost for non-compactness of the explored area. The parameter w_3 is the weight allocated to the 3rd cost term, the cost for not staying within the preferred distance from the other agents.

Using the cost function, the cost for moving from the vertex v_i to the vertex v_j (along the edge $e_{i,j}$) can be expressed as follows:

$$f_c(e_{i,j}) = c(v_j), \tag{14}$$

where $e_{i,j} = \{v_i, v_j\}$ is the edge which connects v_i to v_j.

As it can be observed from the definition of the costs, the cost for moving from the vertex v_i to the vertex v_j (along the edge $e_{i,j}$) depends only on the destination vertex v_j. Of course, this is an idealized scenario for demonstrating the algorithm and in some real-world scenarios one might want to also model particular terrain features (slopes, rocks, etc) which can lead to a slightly different cost structure, but for the exploration algorithm choice of the destination vertex is one of the most important aspects.

4.2. OPTIMAL PLANNING USING GRAPH SEARCH ALGORITHMS

In order to use graph search algorithms like uniform cost search or informed search algorithms like A* [14], IDA* (Iterative deepening A*) [15] or SMA* (Simplified Memory Bounded A*) [16], we have transformed the exploration problem into multiple path planning problems. We consider that a goal state has been reached during the path planning if that state has not been previously explored. Therefore, using an optimal graph search algorithm each agent finds the solution with the lowest cost at each time step, and it moves one step towards that solution.

Since the agents work in a dynamic environment and the other agents also move and explore on their own, in order to preserve the optimality at each time step, each agent searches again for the solution with the lowest cost, even if it hasn't moved completely along the previously planned path.

In our study, we have used the A* algorithm with a consistent heuristic as a particular implementation of the optimal graph search algorithm. The A* graph search algorithm is optimal when the heuristic is consistent [14].

In order to guide the search towards the frontier, we have used the following heuristic:

$$h(s) = \max\left(\frac{1}{\sqrt{2}} \min_{\substack{u \in V \\ f_e(u)=0}} \|r_s - r_u\|_2 - 1{,}0\right), \tag{15}$$

where

u is a vertex which has not been previously explored;

r_u is the position corresponding to the vertex u;

r_s is the position corresponding to the state s.

Latter in this section, we provide detailed explanation on the introduction of the $1/\sqrt{2}$ constant.

In order to prove that $h(s)$ is a consistent heuristic for $w_i \geq 1$ and $w_i \geq 0$ for $i \neq 1$ we have proved that it never overestimates the real cost of the least cost path to an unexplored vertex.

Under these assumptions, for any explored vertex, the

following inequality holds true:

$$1 = c_1(s) \le w_1 \cdot c_1(s) \le c(s), \forall s \in V_e, \quad (16)$$

where $V_e = \{v \in V \mid f_e(v) = 1\}$ is the set of the explored vertices.

Under the same assumptions, for any unexplored vertex, the following inequality holds true:

$$0 = c_1(s) \le w_1 \cdot c_1(s) \le c(s), \forall s \in V \setminus V_e. \quad (17)$$

The optimal path for reaching an unexplored vertex from the current vertex is:

$$P^* = \underset{\substack{(v_0, v_1, v_2 \ldots v_{k-1}) \\ v_i \downarrow v_{i-1}, i=1,k-1 \\ f_a(v_i)=1, i=1,k-1}}{\operatorname{argmin}} \sum_{i=1}^{k-1} c(v_i). \quad (18)$$

The real cost of the optimal path *P** for reaching an unexplored vertex from the vertex *s* is:

$$g(s) = \sum_{\substack{v \in P^* \\ v \ne vo}} c(v). \quad (19)$$

Considering the equations 16 and 17 and the fact the only the last vertex from the optimal path *P** is an unexplored vertex, the following inequality holds true:

$$\left|P^*\right| - 2 \le g(s), \quad (20)$$

where $\left|P^*\right|$ is the number of vertices from the optimal path.

Considering the grid that we used to model the graph that describes the spatial exploration problem, the Euclidian distance between two adjacent nodes is at most $\sqrt{2}$ (for the diagonal neighbors), therefore the Euclidian distance from the current node to the closest unexplored vertex by following the optimal path is at most $\sqrt{2} \cdot \left(\left|P^*\right| - 1\right)$ and the following inequality holds true:

$$\min_{\substack{u \in V \\ f_e(u)=0}} \|r_s - r_u\|_2 \le \sqrt{2} \cdot \left(\left|P^*\right| - 1\right). \quad (21)$$

Based on this result, we can conclude that the following inequality is true:

$$\frac{1}{\sqrt{2}} \min_{\substack{u \in V \\ f_e(u)=0}} \|r_s - r_u\|_2 - 1 \le \left|P^*\right| - 2. \quad (22)$$

Since the path always contains at least two vertices (at least the start vertex and the end vertex), the following inequality holds true:

$$|P^*| \ge 2. \quad (23)$$

Considering the equations (20), (22) and (23), we have proved that *h(s)* is consistent:

$$h(s) \le |P^*| - 2 \le g(s). \quad (24)$$

We have shown that the heuristic *h(s)* that we employed is a consistent heuristic and therefore the A* algorithm is optimal when used with this heuristic.

Considering that the search algorithm employed is optimal, we can conclude that by following the first step from the optimal plan, the agent chooses the optimal action available from the current state considering his knowledge of the environment (the obstacles that have been already discovered, the pheromone map and the sensory input). Although not optimal from global perspective, this approach is optimal from the perspective of each agent and allows each agent to plan its next move independently, making the algorithm useful for fault tolerant and self-healing parallel exploration systems.

Each agent chooses his next action according to the following algorithm, which is based on A* with a consistent heuristic:

Initialize the priority queue *q* with the current state

While *q* is not empty **do**:
  $(state, plan, cost) := q \cdot pop()$

  If *state* was already expanded **then continue**

  If *state* is unexplored:
    $nextAction := plan[0]$
    Break
  End

  Mark *state* as expanded

  For each $(succ, action, actionCost)$ **in** $successors(state)$ **do**:
    If *succ* was already expanded **then continue**

    $candidate\ Plan := (plan, action)$
    $candidate\ Cost := cost + actionCost$
    $tuple := (succ, candidate\ Plan, candidate\ Cost)$
    $priority := candidate\ Cost + heuristic(succ)$
    $q \cdot push(tuple, priority)$
  End
End

This approach is optimal at each step from the perspective of each agent given his knowledge of the environment, but in the worst case, the algorithm needs to explore many states for finding the solution. The worst case time complexity of the algorithm is $O(|E|) = O(b^d)$ and its worst case space complexity is $O(|V|) = O(b^d)$ where *b* is the branching factor and *d* is the search depth.

If the space complexity is of primary concern, then the algorithm could be easily adapted to use the IDA* (iterative deepening A*) algorithm which has a polynomial space complexity $O(b \cdot d)$.

Although the worst case time complexity for A* is exponential, when the agent is at the exploration frontier, the solution states are very close to the current state.

4.3. NEAR-OPTIMAL PLANNING FOR LOW COMPUTATION REQUIREMENTS

Since the optimal planning at each time step is computationally intensive, with a worst case time complexity

$O(b^d)$, we have also analyzed a more computationally efficient approach which still yields near-optimal planning at each time step. This method is based on computing the path to a solution state and then following this path at every time step until the agent reaches that solution state. Afterwards, the agent plans a new path to the next solution, and so on.

We have compared the exploration strategy proposed in this paper with other approaches found in the literature. In order to perform an accurate comparison of the performance, we have implemented the other algorithms and we have run the benchmark on the same set of maps.

For the benchmark, we used two maps: one of 30×30 cells ("Tiny World" from Fig. 1) and another one of 50×50 cells ("Small World" from Fig. 2).

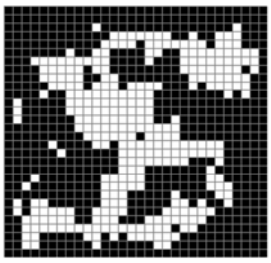

Fig. 1 – The "Tiny World" map. Fig. 2 – The "Small World" map.

In the Fig. 3, we have shown a typical run of the algorithm on a big map, the "World" map. The algorithm has been run with 6 agents using the cost function introduced in equation 14 and the following parameters: $w_1 = 8, w_2 = 1, w_3 = 0.25, R = 6, Q = 2, d_p = 10$. The values for the parameters w_1, w_2, w_3 have been chosen so that the three costs have comparable values on average. It can be observed that the explored area is compact and that the agents stay within the preferred distance from the rest of the exploration team.

In Fig. 4 we present the performance comparison graph from the results obtained by running the algorithms on the "Tiny World" map. The exploration algorithms have been ran 100 times, with 1, 3 and respectively 6 agents. The results are compared in terms of the total path length for all agents required to complete the exploration.

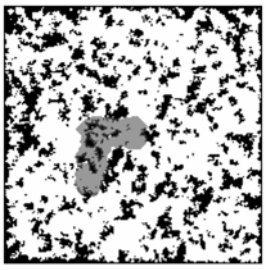

Fig. 3 – A typical run of the exploration algorithm on the "World" map (the explored area is marked with light gray).

We have compared the exploration strategy proposed by us with an information driven algorithm based on entropy minimization [17], with the node counting algorithm [18] which is based on avoiding the states visited in the past, with the algorithm based on reflex agents driven by Thrun's rule [19], with vertex ant walk [20] which is a smell oriented exploration algorithm and with learning real-time A* with a look-ahead of one cell [21, 22].

From Fig. 4 "Tiny World" (1 agent), we can see that the average step count obtained by our algorithm is slightly higher than the one obtained by the entropy based algorithm, but significantly lower than the step count obtained by the other algorithms.

In Fig. 4 "Tiny World" (3 agents and 6 agents), it can be observed that when using multiple agents, the algorithm proposed in this paper outperforms both the entropy based algorithm and the other ant based algorithms.

We have also performed the comparisons on a slightly larger map, the "Small World" map. It can be observed that on the "Small World" map, the performance of the exploration strategy proposed in this paper is close to the performance of the entropy based algorithm and it outperforms the ant based algorithms. A graphical comparison of the performance of the algorithms can be seen in Fig. 5.

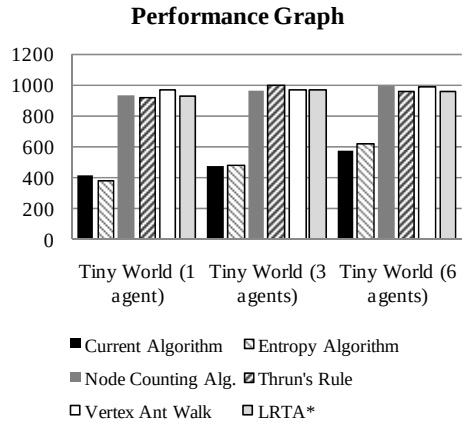

Fig. 4 – Performance comparison graph for "Tiny World" map (lower is better).

In order to show that our results are consistent for a wide range of situations, we have also performed an analysis on a large number of maps. The maps were randomly generated maps with the same size as "Tiny World".

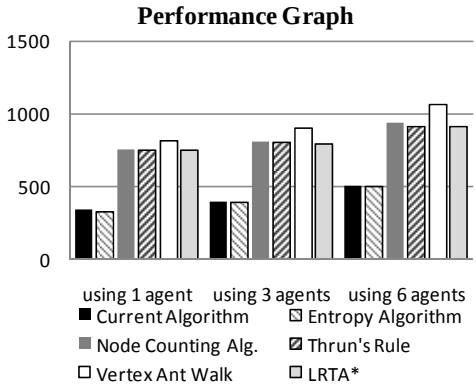

Fig. 5 – Performance comparison graph for the 30×30 sized map (lower is better).

As it can be seen from Fig. 5, the performance of the exploration algorithm proposed in this paper is close to the performance of the entropy based exploration algorithm, but with the added benefit of compactness. In Fig. 5, we have represented graphically the number of steps required

for completely exploring the map, using each of the compared algorithms. It can be seen that the results are consistent with the findings from the detailed analysis on particular maps, presented earlier in this section.

We have also analyzed the performance of the algorithm on larger maps, of 100×100 size, using the same value of the parameters, and the results were consistent with the ones from smaller maps, as it can be observed from Fig. 6.

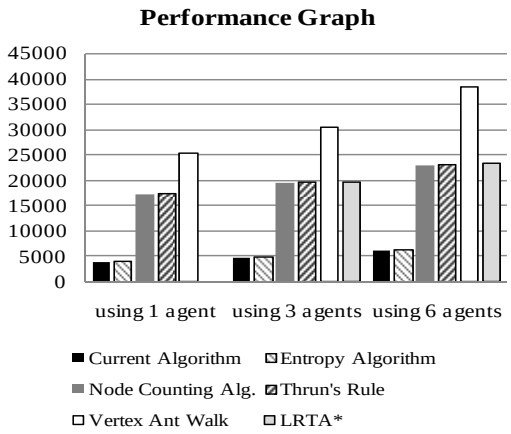

Fig. 6 – Performance comparison graph for 1 000 runs on different maps of size 100×100.

5. DISCUSSIONS AND CONCLUSIONS

Although the graph search algorithms are usually very computationally intensive, by keeping the agents close to the frontier, it is possible to find the solution for the next step by expanding only a small number of nodes. Only in special cases (when a solution does not exist close to the current position of the agent) the full power of the graph search algorithm needs to be used in order to find the path and it can become computationally intensive.

In this paper we have shown that it is possible to improve the shape of the explored area by imposing soft constraints and finding the optimal solution from the perspective of each agent (at each time step).

We have shown that imposing local constraints it is possible to obtain a compact exploration area. This behavior has been obtained by using only local costs, without the need to compute shape factor over the entire explored area.

The exploration strategy proposed in this paper has been studied through simulation and we have shown that in certain scenarios it performs better that the entropy based exploration algorithm and better than the ant-based algorithms, although the advantage of the planning algorithms over the ant based algorithms shrinks as the number of agents increases.

Received on March 19, 2016

REFERENCES

1. E.M. Arkin, S.P. Fekete, J.S.B. Mitchell, *Approximation algorithms for lawn mowing and milling*, Technical report, Mathematisches Institut, Universität zu Köln, 1997.
2. W.-P. Chin, S. Ntafos, *Shortest watchman routes in simple polygons*, Discrete Comput. Geom., **6**, *1*, pp. 9–31(1991).
3. S. Arora, *Polynomial time approximation schemes for Euclidean TSP and other geometric problems*, Proc. 37th Annu. IEEE Sympos. Found. Comput. Sci., 1996, pp. 2–11.
4. X. Tan, T. Hirata, *Constructing shortest watchman routes by divide and conquer*, Proc. 4th Annu. Internat. Sympos. Algorithms Comput., **762**, Lecture Notes Comput. Sci., Springer-Verlag, 1993, pp. 68–77.
5. F. Hoffmann, C. Icking, R. Klein, K. Kriegel, *A competitive strategy for learning a polygon*, Proc. 8th ACM-SIAM Sympos. Discrete Algorithms, 1997, pp. 166–174.
6. F. Hoffmann, C. Icking, R. Klein, K. Kriegel, *The polygon exploration problem: A new strategy and a new analysis technique, Robotics: The Algorithmic Perspective*, Proc. 3rd Workshop Algorithmic Found. Robot, A.K. Peters, 1998, pp. 211–222.
7. S. Albers, K. Kursawe, S. Schuierer, *Exloring unknown environments with obstacles*, Algotithmica, **32**, pp. 123–143 (2002).
8. A. Itai, C.H. Papadimitriou, J.L. Szwarc- fiter, *Hamilton paths in grid graphs*, SIAM Journal of Computing, **11**, pp. 676–686 (1982).
9. M. Betke, R. Rivest, M. Singh, *Piecemeal learning of an unknown environment*, Proc. 5th Conference on Computational Learning Theory, 1993, pp. 277–286.
10. S. Albers, M. Henzinger, *Exploring unknown environments*, SIAM Journal on Computing, **29**, pp. 1164–1188 (2000).
11. W. Cohen, *Adaptive mapping and navigation by teams of simple robots*, Journal of Robotics & Autonomous Systems, **18**, pp. 411– 434 (1996).
12. D. Apostolopoulos, L. Pedersen, B. Shamah, K. Shillcutt, M. Wagner, W. Whittaker, *Robotic antarctic meteorite search: Outcomes*, Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), 2001, pp. 4174–4179.
13. M. Bender, D. Slonim, *The power of team exploration: two robots can learn unlabeled directed graphs*, Proc. of the 35rd Annual Symposium on Foundations of Computer Science, 1994, pp. 75–85.
14. C. Icking, T. Kamphans, R. Klein, E. Langetepe, *Exploring an Unknown Cellular Environment*.
15. I. Wagner, M. Lindenbaum, A. Bruckstein, *Distributed covering by ant-robots using evaporating traces*, IEEE Transactions on Robotics and Automation, **15**, *5*, pp. 918–933 (1999).
16. S. Russell, *Efficient memory-bounded search methods*, Proceedings of the 10th European Conference on Artificial intelligence, Vienna, Austria, 1992.
17. M. Baglietto, M. Paolucci, L. Scardovi, R. Zoppoli, *Information-Based Multi-Agent Exploration*.
18. A. Pirzadeh, W. Snyder, *A unified solution to coverage and search in explored and unexplored terrains using indirect control*, Proceedings of the International Conference on Robotics and Automation, 1990, pp. 2113–2119.
19. S. Thrun, *Efficient exploration in reinforcement learning*, Technical Report CMU-CS-92-102, School of ComputerScience, Carnegie Mellon University, Pittsburgh (Pennsylvania), 1992.
20. I. Wagner, M. Lindenbaum, A. Bruckstein, *On-Line Graph Searching by a Smell-Oriented Vertex Process*, AAAI Technical Report WS-97–10, 1997.
21. S. Koenig, Y. Liu, *Terrain Coverage with Ant Robots: A Simulation Study*, AGENTS'01, Montreal, 2011.
22. R. Korf, *Real-Time Heuristic Search*, Artificial Intelligence, **42**, pp. 189–211 (1990).