Multi-robot taboo-list exploration of unknown structured environments

Mihai Andries, François Charpillet

Multi-robot taboo-list exploration of unknown structured environments

Mihai Andries [1,2,3] and François Charpillet [1,2,3]

***Abstract*— This paper presents a new taboo-list approach for multi-robot exploration of unknown structured environments, in which agents are implicitly guided in their navigation on a globally shared map. Agents have a local view of their environment, inside which they navigate in a asynchronous manner. When the exploration is complete, agents gather at a rendezvous point. The novelty consists in using a distributed exploration algorithm which is not guided by frontiers to perform this task. Using the *Brick&Mortar Improved* ant-algorithm as a base, we add robot-perspective vision, variable vision range, and an optimization which prevents agents from going to the rendezvous point before exploration is complete. The algorithm was evaluated in simulation on a set of standard maps.**

I. Introduction

Exploration of unknown environments is an important field of research for autonomous robotics. Its applications include reconnaissance, search and rescue missions, and planetary exploration.

This paper presents a decentralized exploration algorithm, called BMILRV, able to identify the completion of exploration. Agents explore by leaving virtual traces in the environment, and by moving in a non-coordinated and asynchronous manner. In the end, the exploring agents gather at a pre-defined rendezvous point, if so required. Parameters of this algorithm include: the number of agents, and their viewing range.

BMILRV stands for Brick&Mortar Improved Long Range Vision. As its name suggests, the new algorithm is based on the Brick&Mortar Improved algorithm [1], to which it adds long range vision capabilities to enhance its exploration speed. This modification required the introduction of new marking rules, that respect the connectivity of the open subgraph even when an entire region of cells has to be marked. Theses rules define how to mark the frontiers of the visible area, the shadows, and the exit paths that robots leave for themselves. Another novelty is the addition of a trail to the rendez-vous point, which prevents agents from following it until the exploration is complete.

The remainder of this paper is organized as follows: Section II presents the related work in exploration of unknown environments. Section III presents a description of the exploration and rendezvous problem. A detailed description of the proposed BMILRV algorithm is given in section IV. Experimental simulation results are presented and analyzed in section V. We discuss the issues of implementing the proposed algorithm in robots in section VI. We conclude this paper with perspectives for future work.

[1]Inria, Villers-lès-Nancy, 54600, France
[2]CNRS, LORIA, UMR 7503, Vandoeuvre-lès-Nancy, 54506, France
[3]Université de Lorraine, LORIA, UMR 7503, Vandoeuvre-lès-Nancy, 54506, France
firstname.lastname@inria.fr

II. Related work

Two main families of algorithms exist today for exploration of unknown environments: (1) frontier-based algorithms, and (2) ant-algorithms.

Frontier-based algorithms guide the exploration by pushing the agents towards the boundaries between the open explored space and the space yet unexplored [2], [3], [4], [5], [6]. The exploration ends when all the space was discovered, implying that there are no more frontiers to move to (see Fig. 1). The navigation to these frontiers implies planning on the whole map of the enviornment. In case of large maps, this planning may be computationally expensive.

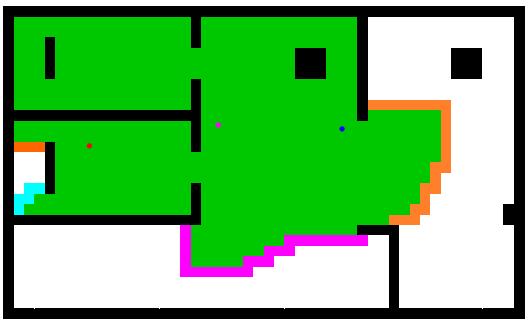
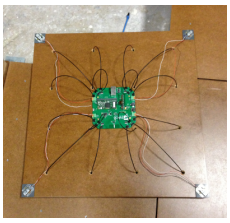


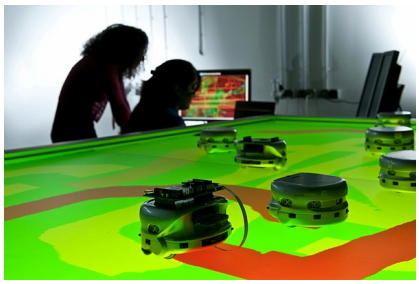
Fig. 1: Frontier exploration example. Frontiers between unexplored and explored regions are highlighted in color. Agents, shown here as dots, explore by navigating towards these frontiers. Exploration ends when there are no more frontiers and all the environment is explored.

Ant-algorithms also use a shared map, on which agents lay traces that guide their exploration. For instance, these traces may allow agents to do a gradient-descent exploration of the environment [7], [8]. Compared to frontier-based algorithms, where agents plan their navigation using their entire knowledge of the map, ant agents reason only on the locally visible fragment of the map. This reduces the complexity of navigation planning.

Although ant-algorithms were longtime considered non-realistic for robotic implementations, the emergence of environments with embedded sensor networks shifted this point of view (see Fig. 2). Load-sensing floors [9] with embedded processing units (see Fig. 2a) can store and dissipate artificial pheromones, and could one day guide cleaning robots through the home. Interactive screen surfaces can be used in a similar manner (see Fig. 2b).

(a) A PekeeII robot on a tiled load-sensing floor with embedded processing units [9]



(b) Khepera III robots performing a foraging tasks on an interactive table [10]. The intensities of pheromone traces are color coded using a red gradient.

Fig. 2: Platforms capable of supporting pheromone traces.

III. Exploration and rendezvous: problem description

In practice, mapping scenarios usually involve three tasks:

1) Explore and perceive all the space.
2) Identify the end of exploration.
3) Gather explorers at a rendezvous point.

This multi-agent exploration of an environment can be formally described as follows: a set of agents explore a graph $Graph$, starting from node $Node_{start}$. The graph known to agents, called $Graph_{agents}$, is expanded through actions of type $navigateTo(node)$. Every time an agent sees a new portion of the graph, this portion is added to the known graph $Graph_{agents}$.

$$
\begin{aligned}
Graph &= \{Nodes, Links\} \\
Graph_{agents} &= \{Node_{start}, Links_{start}\} \\
Agents &= \{a_1, \ldots, a_n\} \\
Actions &= \begin{cases} navigateTo(node) \\ mark(node, markType) \end{cases}
\end{aligned}
$$

Exploration ends when there are no more nodes to expand. When a global map of the environment is available, this end condition can be checked by looking if all the nodes have been expanded. However, another solution is required when agents have only a partial knowledge of the map.

Ant agents possess a local vision of the map, seeing only the surroundings in their field of view. However, agents lay their pheromones on the same environment, meaning that access to the map is shared. Shared map access can be easily implemented in environments able to support virtual traces (pheromones), like, for example, the floor presented in Fig. 2a. In cases when environment-embedded memory is not available, map sharing can be implemented through communication with a central node: agents share only their views and markings of the environment, without sharing their navigation plans or anything else.

Situations where agents explore a shared environment while having only local views of it can be solved using taboo-list approaches. A taboo-list approach prevents agents from going to regions of the graph that were already explored, and concentrates them in the remaining non-explored parts of the graph. By continually reducing the region of the graph inside which they can navigate, agents get grouped together in the remaining part of the graph. Ultimately, they meet in the last available node, which is the rendezvous point. When this last node is closed, exploration is considered complete. Having gathered all the agents in this node ensures that they are all informed about the end of exploration by looking at the state of the open graph.

$Graph_{agents}$ = $\text{Taboo}_t \bigcup \overline{\text{Taboo}}_t \bigcup \text{Unexplored}_t$ nodes

Taboo = explored and closed nodes

$\overline{\text{Taboo}}$ = explored, not yet closed nodes

Unexplored = unexplored nodes

IV. Algorithm description

The algorithm described in this paper is a continuation of our previous work [1] on the original Brick&Mortar algorithm developed by Ferranti et al. [11].

Our current contribution consists in developing new marking rules for the case of agents with an extended viewing range. This allows agents to mark multiple cells of the environment in a single step. This transforms the ant-like algorithm BMI into one which uses a realistic vision representation. The visible portion of the map is calculated from each agent's perspective, as compared to a top-down view of the map in previous versions of the Brick&Mortar algorithm (see figures 3 and 5).

Agents share a common map, of which they can access only the portion surrounding them. A node can be occupied by multiple agents at the same time. Agents do not coordinate their decisions and act asynchronously. However, the marking and navigation decisions they make are considered atomic. Cell-marking cost is zero, while navigating from one cell to another neighboring cell takes 1 timestep.

In comparison to frontier-closing exploration algorithms, where agents plan their movements toward the frontiers of already explored regions, the proposed algorithm requires only minimal navigation planning. After marking the nodes in its field of view, agents move to one of the cells surrounding their current location.

The exploration algorithm discretizes the surrounding world, treating it as a graph (nodes correspond to cells in a 2D grid world). This is convenient, as both 2D and 3D maps can be represented as graphs. Agents walk through this graph, trying to reduce it in size by limiting access to explored nodes. By continually decreasing the number of nodes in the open graph, while keeping it connected at the same time, agents end up gathered in the last remaining node, which is the rendezvous point (see Fig. 3 for an example

of the algorithm execution). For clarity reasons, we shall employ 2D grids instead of abstract graphs in the rest of this paper, and use the word *cell* instead of *node*.

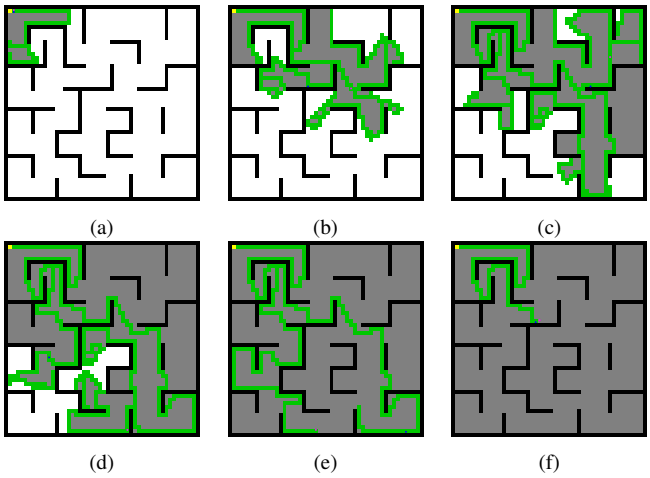


Fig. 3: Exploration performed by two BMILRV agents. The rendezvous point is in the top-left corner of the map.

The algorithm uses 5 types of cell markings : walls, unexplored regions, explored regions, closed regions, and the rendezvous point (see Fig. 4). Unexplored and explored cells form the open parth of the map, in which agents are authorized to navigate. The closed cells are the ones added to the taboo-list, and to which access is forbidden.

Fig. 4: Cell types used by BMILRV

At each time step, each agent performs 2 activities: marking and navigation. During marking, the agent considers all the cells inside its field of view (see Fig. 5). If the removal of a cell doesn't break the connectivity of the remaining open portion of the map, then this cell is declared closed for further access. If a cell is necessary for maintaning the connectivity of unexplored regions, this cell is declared as *explored* and left open for further access. Thus, a cell is left open if:

- it is at the edge of the field of view, potentially linking the environment outside the field of view to the one inside it;
- it is adjacent to any non-visible cells inside its field of view (i.e. shadows) which potentially hide unexplored cells;
- it disconnects the map open for access inside the agent's field of view.

Considering the multi-agent nature of the algorithm, a 4th condition is required: a cell cannot be closed if that will block other agents. This includes checking for the presence of robots or their mutex traces in the cells surrounding the cell being analyzed.

The order in which the cells inside an agent's field of view are closed influences the shape of paths that remain open for continuing the exploration of the graph. In Fig. 5, for instance, the cells were analyzed in clockwise direction, from the ones closest to the agent to the farthest ones, starting from the 12 o'clock position.

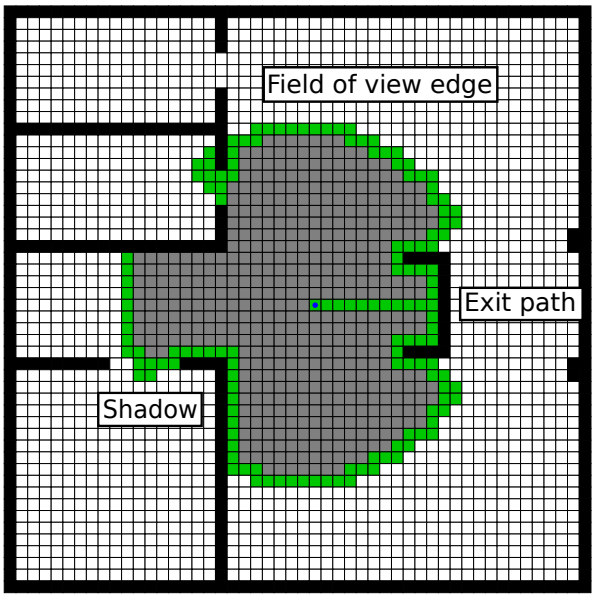
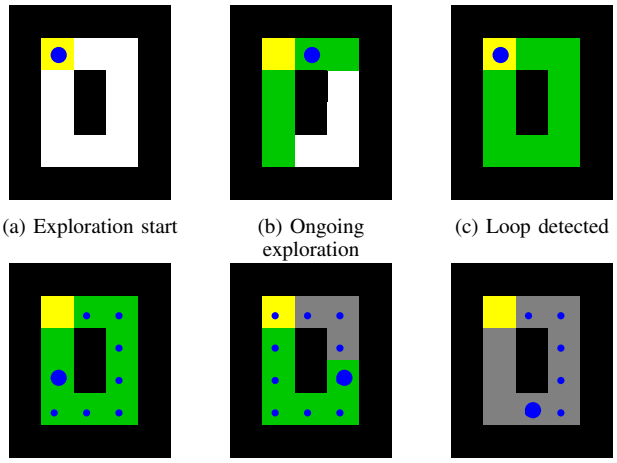


Fig. 5: BMILRV marking illustration. All the cells in the viewing range are closed, except those on its edge, and those near the shadows of objects. An exit trail is implicitely left by the algorithm, as its absence would have cut the agent from the rest of the environment. Its shape depends on the order in which the cells in the viewing range are closed. All the unexplored regions remain accessible.

After the agent has marked the cells inside its viewing range, it will mark the cell underneath itself. If this cell is left open, the marking will also contain the direction followed to exit this cell. The agent will also set the value of the dispersion gradient inside this cell, which is used to direct agents between explored cells left open for navigation. Initially, the dispersion gradient value for all the cells is set to 0. Updating the dispersion gradient value of a cell assigns it the minimal surrounding gradient value plus one, a policy known as LRTA* in the literature [8]. This pushes agents down the dispersion gradient to the surrounding cells.

It may happen that several agents share the same cell (particularly in the beginning of exploration), when only the first agent that marks the surrounding environment gets to influence it. If a single exit path is left open by the first agent, the remaining agents will have to follow it, preventing their dispersion. This behavior can be avoided by forcing agents to limit their marking range to the distance between them and their closest neighbor inside their field of view. If no other agent falls inside their field of view, then the normal marking range is used. Experimental results have shown that this decreases the time till full exploration of the environment, as determined by an external observer. However, this does not reduce the time till mission completion and rendez-vous.

In the navigation step, the agent moves to one of the open neighboring cells, giving priority to unexplored cells over the explored ones. If it travels from an explored cell to another explored cell, it will follow the dispersion gradient, which



(a) Exploration start (b) Ongoing exploration (c) Loop detected

(d) Loop control (e) Loop closing (f) Loop cleaning

Fig. 6: The phases of the loop closing algorithm: loop detection (6c), loop control (6d), loop closing (6e), and loop cleaning (6f).

will dictate its direction of movement.

A problem which arises out of this algorithm is that an agent cannot close a environment of an annular (ring or loop) topology, of which it only sees a fragment at any given time. Closing any cell inside its field of view would violate the connectivity condition for the remaining visible environment (see Fig. 6a).

Solving this problem requires an algorithm for detecting and closing such loops. Considering that several agents can simultaneously attempt to close the same loop, the algorithm should include a distributed mechanism for priority resolution among agents. In Brick&Mortar, this translates into a 4-step algorithm for loop closing (see Fig. 6). Agents leave traces in the environment that enable them to detect whenever they enter a cell for a second time, which implies that they went through a loop (except for cases when they re-enter the cell from opposite direction). First, on detection of a loop, the control over it is gained by a single agent in a distributed mutual exclusion manner. Then, the agent breaks the loop by closing a part of it, while still preserving the connectivity of the environment. Finally, the agent relinquishes the loop by cleaning the marks it used for exclusive control.

We also improved the exploration by preventing the agents from going to the rendezvous point if the exploration hasn't yet completed. This is done when there is a unique way that leads from the rendezvous point to the rest of the graph. It was implemented using the dispersion gradient that agents employ to disperse themselves in the environment. Maximum gradient values are set to the cells composing this unique path leading to the rendezvous point, which prevents access to them for the agents performing gradient descent, unless they have no other choice.

The pseudocode of the BMILRV multi-agent exploration algorithm is given below, in Algorithm 1. The state-machine describing this algorithm is presented in Fig. 7.

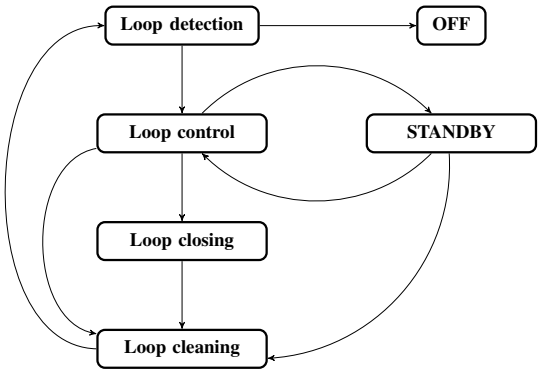
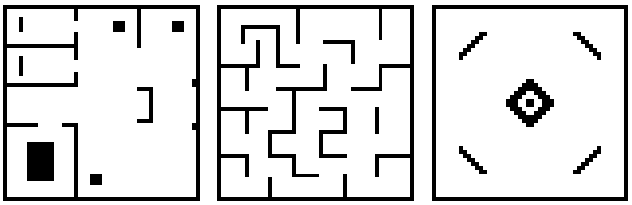


Fig. 7: The state machine of the BMILRV algorithm.



(a) Office (b) Maze (c) Garden

Fig. 8: Maps explored during the experiments.

V. Experiments

We evaluated the performance of BMILRV in a series of simulations on maps mimicking human environments, such as offices cluttered with obstacles, mazes and gardens (see Fig. 8). BMILRV is currently the only known multi-agent algorithm where agents use a shared map, have a local vision of this map, don't communicate their exploration targets, and, most importantly, identify the end of exploration. No comparable algorithm of the same class exists, to our knowledge.

However, in the absence of comparable state-of-the-art algorithms, we can still evaluate BMILRV relative to some baseline. In this sense, we used a group greedy frontier explorationg algorithm as baseline. This is a sub-optimal heuristic, which avoids the complexity of solving a Hungarian algorithm to calculate an optimal assignment of agents to frontiers. It assigns agents to frontiers in a greedy manner, incrementally selecting for assignment pairs of agents and frontiers with minimal distance between them.

In both cases agents communicate to share a common map (in BMILRV, this is done via virtual pheromone traces). However, in comparison to BMILRV, frontier exploration algorithms make 2 additional strong assumptions. First, agents communicate between themselves to distribute the exploration effort by selecting different frontiers to explore, and thus avoid following each other. Second, agents use their knowledge of the entire map (as opposed to knowledge of only the visible surroundings in BMILRV) to plan optimal navigation routes.

The parameters that varied in our experiments were: the number of agents performing the exploration, and the size of the map. The viewing range of agents in all our experiments was fixed at 10 grid cells. The results are shown in Fig. 9.

Algorithm 1. BMILRV

```
BMILRV_step()
{ // The agent first checks its state
  // and then performs the corresponding
      action
  if (agentState == LOOP_DETECTION)
  {
    // Marking
    Repeat
    {
      Mark all open cells inside your field
          of view (except the one you are
          standing on, to avoid being
          blocked).
    } while at least one cell was closed
        inside field of view

    If you have already been here (not
        coming from the opposite direction),
        and if the cell is not controlled by
        another agent, and if you have not
        closed cells since your last visit
        to this cell, then switch directly
        to LOOP CONTROL.
    Else
      - mark the cell you are standing on.
      - update the gradient for dispersing
          the agents on this cell.

    // Navigation
    Move to open neighboring cell. Prefer
        unexplored cells over explored
        cells. Prefer those with more walls
        and closed cells around. If nowhere
        to go, close the cell you are on and
        turn OFF.
  }

  else if (agentState == LOOP_CONTROL)
  {
    Mark the cell as controlled by you,
      continuing the same path as the one
          which led you into this loop.
    When the entire loop is under your
        control, switch to LOOP CLOSING.

    If you could not control the entire
        loop, because:
      - the cell was closed by someone else,
      - you did not find your trace,
      - the loop is controled by someone
          with higher priority,
    then switch to LOOP CLEANING.

    If someone with lower priority controls
        this cell, then switch to STANDBY
        until this cell gets cleaned.
  }

  else if (agentState == LOOP_CLOSING)
  {
    Close cells of the marked loop until the
        first bifurcation after the place
        where you started the closing phase.
    Then switch into LOOP CLEANING state.
  }

  else if (agentState == LOOP_CLEANING)
  {
    Clean your loop control traces, by
        moving backwards through the loop,
        while these traces exist.
    When cleaning is over, switch to LOOP
        DETECTION.
  }

  else if (agentState == STANDBY)
  {
    If the cell on which the agent stands:
      - becomes closed,
      - or is taken over by an agent with
          higher priority
    then switch to LOOP CLEANING.

    If the cell gets cleaned of other agents
        traces, then continue in LOOP
        CONTROL.

    Else remain in STANDBY.
  }

  else if (agentState == OFF)
    Agent is turned off.
} // End of "BMILRV_step" function


/* Marks a cell while in LOOP DETECTION */
markCell(cell)
{
  If the cell is the rendezvous point, then
      leave no mark on it.
  Else if the cell is unexplored:
    - if it is blocking, then mark is as
        explored.
    - else if it is not blocking,
      then close it, and update the time of
          the last cell closing.
  Else if the cell was explored:
    - if it is not blocking, then close it,
        and update the time of the last cell
        closing.
}


/* Identifies if a cell is blocking inside
    the field of view of an agent*/
isBlocking(cell, field of view)
{
  A cell is blocking if:
    - it is at the edge of the field of view,
    - or it is adjacent to a cell in shadow
        (non-visible cell behind an
        obstacle),
    - or if closing it would disconnect the
        open environment inside the field of
        view,
    - or if closing it would block other
        agents behind.
}
```

Increasing the number of exploring agents leads to shorter exploration times, as seen in Fig. 9a. This means that the algorithm is able to distribute the exploration effort among its agents. However, exploration efficiency is also dependent on the size and topology of the environment, which can get saturated with agents. In Fig. 9a, the environment gets saturated at 5 agents, after which adding new agents doesn't decrease the exploration time. As compared to the greedy Frontier Exploration algorithm, BMILRV is slowed down by obstacles present in the environment, which force agents to go through the costly loop closing algorithm, in order to close the loops that form around such obstacles. Navigation in the environment is also suboptimal for BMILRV, as the paths left open by agents are not of shortest possible length.

Fig. 9b shows the number of steps required for exploration completion and rendezvous in maze-type environments. Again, BMILRV is slowed down by the number and size of isolated obstacles present in the environment. Better comparative results are obtained for environments of type *Garden*, where the number and size of obstacles is reduced, as compared to *Maze* maps (see Fig. 9c).

VI. Discussion

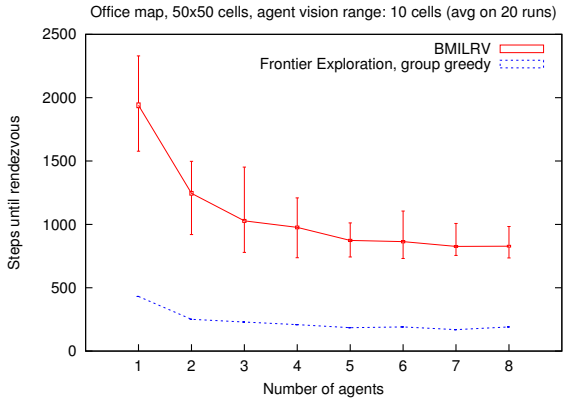
Considering that all the presented experiments were performed in simulation, it would be interesting to implement the BMILRV algorithm in real robots. This raises the issue of uncertainty in sensing, in localization, and in the movements of robots.

Regarding the uncertainty in localization, the environment itself could help agents localize themselves, as the agents do not maintain the global map in their memory. For instance, a sensing floor as the one presented in Fig. 2a can identify which tiles are occupied. By tracking the robots during their exploration, the floor can also uniquely identify them, and communicate them their exact location.

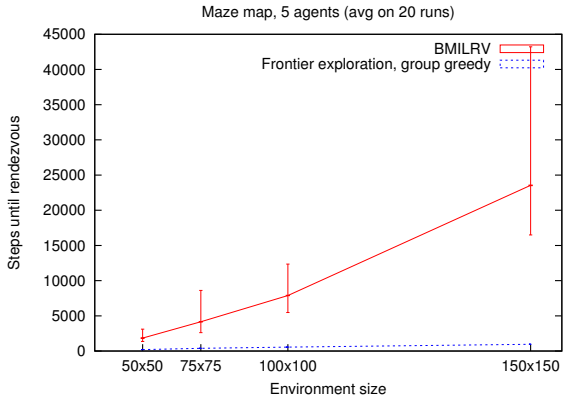
The uncertainty in robot movements will only impact their ultimate location. Errors in localization generated by erroneous movements can be identified by the mismatch between the expected location of the robot and the one communicated by the environment. Robots could then attempt to return to their intented location. Another solution would be to allow the robot to reset the markings in the visible portion of the environment surrounding the robot, allowing it to move out of any region it might have stepped into, including previously closed regions.

The sensing uncertainty would influence the way the cells of the environment are perceived and marked. To prevent errors, the graph connectivity conditions could be checked by the intelligent environment itself, which is supposed to have a perfect knowledge of its topology and of all cell markings.

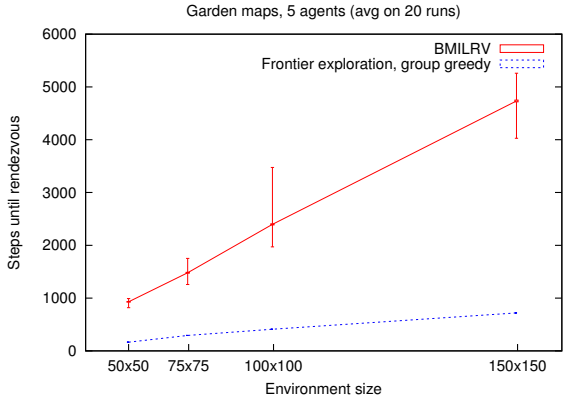
The long range vision capability of BMILRV brings with it an acceleration in marking speed. Compared to Brick&Mortar Improved, where agents only mark the cell underneath themselves, in BMILRV the marking process is accelerated by a factor proportional to (half) the perimeter of the field of view.



(a) Time-steps till exploration completion and rendezvous on the office map (Fig. 8a). Size of the map: 50x50 pixels. Agent viewing range: 10 cells. BMILRV is capable to distribute the exploration effort.



(b) Time-steps till exploration completion and rendezvous on Maze maps (Fig. 8b). Exploring agents: 5. Agent viewing range: 10 cells. The size and number of isolated obstacles present in the environment heavily impact the efficiency of BMILRV.



(c) Time-steps till exploration completion and rendezvous on Garden maps (Fig. 8c). Exploring agents: 5. Agent viewing range: 10 cells.

Fig. 9: Experimental results obtained using simulated map explorations. The performances of BMILRV and greedy Frontier exploration are shown together for illustration purposes only. These algorithms are of different categories and are not directly comparable.

However, the long range vision doesn't influence the way agents close loops around obstacles, since obstacles will always occlude a portion of the environment behind them, if seen from agents' perspective. Nevertheless, the situation is

different for agents with a top-down viewing perspective (e.g. aerial vehicles), from which obstacles generate no occlusions or shadows. In this case, obstacles of size inferior to the field of view will not generate loops at all.

VII. Conclusions and future work

We have presented a multi-agent algorithm for exploration of unknown environments, based on a taboo-list approach. Agents mark explored parts of the environment as closed for access, while keeping the unexplored regions interconnected until the end of exploration. As agents can only travel through open paths in the environment, they decide on the direction of the next movement without using high-level planning. Our contribution is in generalizing this algorithm to agents with variable viewing ranges, allowing them to mark multiple cells at a time. We also optimized it by preventing agents from going to the rendezvous point before exploration is complete.

Future work will focus on a better dispersion of exploring agents. Agents could also reshape the corridors left open to keep the non-explored regions connected, and which are often sub-optimal in length. The impossibility of deadlock among agents is also a property we would like to prove.

References

[1] M. Andries and F. Charpillet, "Multi-robot exploration of unknown environments with identification of exploration completion and post-exploration rendez-vous using ant algorithms," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Tokyo, Japan, Nov. 2013.

[2] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*, Jul 1997, pp. 146–151.

[3] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. Younes, "Coordination for multi-robot exploration and mapping," in *AAAI/IAAI*, 2000, pp. 852–858.

[4] W. Burgard, M. Moors, and F. Schneider, "Collaborative exploration of unknown environments with teams of mobile robots," in *Advances in plan-based control of robotic agents*. Springer, 2002, pp. 52–70.

[5] W. Burgard, M. Moors, C. Stachniss, and F. Schneider, "Coordinated multi-robot exploration," *Robotics, IEEE Transactions on*, vol. 21, no. 3, pp. 376–386, June 2005.

[6] A. Bautin, O. Simonin, and F. Charpillet, "SyWaP: Synchronized Wavefront Propagation for multi-robot assignment of spatially-situated tasks," in *ICAR 2013 : International Conference on Advanced Robotics*, Uruguay, Nov. 2013, p. 7.

[7] I. A. Wagner, M. Lindenbaum, and A. M. Bruckstein, "Distributed covering by ant-robots using evaporating traces," *IEEE Transactions on Robotics and Automation*, vol. 15, pp. 918–933, 1999.

[8] S. Koenig, B. Szymanski, and Y. Liu, "Efficient and inefficient ant coverage methods," *Annals of Mathematics and Artificial Intelligence*, vol. 31, pp. 41–76, May 2001.

[9] N. Pepin, O. Simonin, and F. Charpillet, "Intelligent Tiles: Putting Situated Multi-Agents Models in Real World," in *International Conference on Agents and Artificial Intelligence - ICAART'09*, Porto, Portugal, 2009. [Online]. Available: http://hal.inria.fr/inria-00339231

[10] O. Simonin, T. Huraux, and F. Charpillet, "Interactive Surface for Bio-inspired Robotics, Re-examining Foraging Models," in *23rd IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*. Boca Raton, USA: IEEE, Nov 2011.

[11] E. Ferranti, N. Trigoni, and M. Levene, "Brick&Mortar: An online multiagent exploration algorithm," in *IEEE Int. Conf. Robotics and Automation (ICRA)*, 2007.