Enhancing Exploration in Graph-like Worlds

Hui Wang Michael Jenkin Patrick Dymond
Department of Computer Science and Engineering, York University
4700 Keele Street, Toronto, Ontario, Canada
{huiwang,jenkin,dymond}@cse.yorku.ca

Abstract

This paper explores two enhancements that can be made to single and multiple robot exploration in graph-like worlds. One enhancement considers the order in which potential places are explored and another considers the exploitation of local neighbor information to help disambiguate possible locations. Empirical evaluations show that both enhancements can produce a significant reduction in exploration effort in terms of the number of mechanical steps required over the original exploration algorithms and that for some environments up to 60% reduction in mechanical steps can be achieved.

1. Introduction

The problem of exploring and mapping autonomously an unknown terrain is a core task in robotics. In the literature this is commonly referred to as *SLAM*, *Simultaneous Localization and Mapping* [12, 13]. In the majority of *SLAM* approaches the environment is represented through a metric map that captures the geometric properties of the environment (e.g. [6]). An alternative to a metric-based representation is a topological or graph-like representation [8, 9, 11, 1, 2, 3, 4]. Such a representation provides the minimal information that a robot must be able to represent in order to distinguish one place from another. [3, 4] developed a non-probabilistic *SLAM* algorithm for graph-like worlds in which no distance or orientation metric is defined. In this work the world is modeled as an embedded graph, i.e., a graph in which there exists an ordering of edges incident upon each vertex. The model supposes the existence of a unique *marker* that can be used to help disambiguate locations in the environment (vertices in the graph-like world). In [3, 4] it was shown that graph-like worlds can be fully explored and mapped by a single robot equipped with a unique marker. Later work [5] sketched how the single robot exploration algorithm of [3, 4] can be adapted to the problem of multiple robot exploration. This sketch suggested how multiple mobile agents might exploit the abilities developed in [3, 4] in order to explore in a coordinated fashion. The work in [14] formally develops the sketch provided in [5] and extends [3, 4] to the problem of multiple robot graph exploration. This extension assumes the same formalism as described in [3, 4] and populates the world with two or more robots each of which is equipped with its own unique marker. The exploration algorithm presented in [14] incorporates the core requirements for achieving multiple robot exploration: a technique to merge partial world representations obtained by the different robots, a technique to partition the merged portion of the world between the robots so that they continue to explore, and a rendezvous schedule for the robots to coordinate their exploration activities. Empirical evaluations show that multiple robots can provide a reduction in exploration effort in terms of the number of mechanical steps required over that of a single robot and that for some environments this improvement is super-linear.

This paper investigates how the exploration task in the above papers can be conducted in a more effective way. We explore two enhancements that can be made to both the single robot exploration algorithm [3, 4] as well as to portions of the multiple robot exploration algorithm [5, 14]. The first enhancement considers a breadth-first exploration in order to reduce repeated traversals during exploration, the second enhancement considers exploiting local neighbor structure information to help disambiguate possible confusions during exploration. Empirical evaluations show that both the enhancements can provide a reduction in exploration cost over that of the original algorithms and that for some environments up to a 60% reduction of the mechanical work required can be achieved.

The paper is organized as follows. Section 2 reviews the world model and both the single and multiple robot exploration algorithms given in [3, 4, 5, 14], which form the basis of the work in this paper. Section 3 presents a strategy in which a breadth-first exploration order is used. Section 4 presents a strategy in which the local neighbor information is exploited during exploration. Section 5 presents possible directions for future work.

2. The world model and exploration algorithms

2.1. Basic model

The world to be explored is modeled as a finite augmented undirected graph. The graph is augmented through an embedding for the edges at each vertex. The goal of the robot's exploration is to build an augmented undirected graph that is isomorphic [7] to the finite world it has been assigned to explore. The robot's inputs are its sensations and it can interact with the world only through its actions. These are described below.

The World The world is defined as an embedding of an undirected graph $G = (V, E)$ with set of vertices $V = \{v_1, ..., v_n\}$ and set of edges $E = \{(v_i, v_j)\}$. The definition of an edge is extended to allow for the explicit specification of the order of edges incident upon each vertex of the graph embedding. This ordering is obtained by enumerating the edges in a systematic (e.g., clockwise) manner from some standard starting direction.

Perception A robot's perception of its environment is limited to *edge-related*, *marker-related* and *robot-related* perception. With *edge-related* perception, a robot can determine the relative positions of edges incident on the current vertex v_i in a consistent manner, e.g., by a clockwise enumeration for a planar graph. As a result, the robot can identify the edge through which it entered the vertex and, because the graph is embedded, can assign a local label to each edge in the vertex. Note that this local edge ordering is not, in general, equal to the unknown ordering specified by the graph embedding, but will be a rotation of it. *marker-related* perception enables a robot to sense whether its unique marker is present at the current vertex. With *robot-related* perception, a robot is able to sense the presence of the other robot(s) at the current vertex (robots can meet in a vertex and sense each other when they meet).

Movement and marker operation A robot can move from one vertex to another by traversing an edge (a *move*). A sequence of moves along a path is reversible. A robot can put down the marker it holds at the current vertex and it can pick up its marker if it is located at the current vertex (a *marker operation*).

Inter-robot communication In multiple robot case the robots can communicate with each other only when they are at the same physical location (vertex of the graph).

2.2. Exploring a graph with single robot

It was shown in [3, 4] that as long as the explorer is equipped with a single unique marker that can be dropped and picked up at will it was possible for a robot to fully map its environment. The algorithm proceeds by incrementally building a known map out of the known subgraph. As new vertices are encountered, they are added to the explored subgraph, and their outgoing edges are added to the set of edges that lead to unknown places and therefore must be explored. The core technique is to "validate" (disambiguate) locations that could be confused, based on the fact that the path taken by the robot can be retraced, and that the marker is unique.

The algorithm maintains an explored subgraph S, and a set of unexplored edges U, which emanate from vertices of S. A step of the algorithm consists of selecting an unexplored edge $e = (v_1, v_2)$ from U, and validating the vertex v_2 at the unexplored end of the edge, i.e., to make sure that v_2 is not identical to any other vertex in the explored subgraph S. This process is carried out by placing the marker at v_2 and visiting all potential confusing vertices of S, looking for the marker. If the marker is found at vertex v_i of S, then vertex v_2 (where the marker was dropped) is identical to the already known v_i (where the marker was found). In this case, the edge is added to S and removed from U. If the marker is not found at one of the vertices of S, then vertex v_2 is not in S and is added to S. The previously unexplored edge e is also added to S, which has now been augmented by one edge and one vertex. Unexplored edges incident on v_2 are added to the set of unexplored edges U. The algorithm terminates when U is empty. The cost of exploring the graph in terms of edges traversed by the robot (its mechanical complexity) follows from the need to go back to the known subgraph and visit all potentially confusing locations there to solve the 'have I been here before' problem. In the probabilistic *SLAM* literature this problem is known as 'loop closing' [12, 13]. Note that the choice of which edge e to choose from the set of unexplored edges in U is arbitrary. In [3, 4] a 'closest-first' strategy was used to select the edge.

2.3. Exploring a graph with multiple robots

[5, 14] extended [3, 4] to use two or more robots to solve the exploration problem in graph-like world. Joint exploration is achieved through alternating phases of independent exploration by the individual robots and coordinated merging of the independently acquired partial world representations. At any time, the robots retain a common representation of some part of the world (the commonly known subgraph) S_m that evolves over time, as well as independent information regarding other parts of the world. As succes-

sive iterations of the exploration and merging process take place, S_m grows monotonically until it is isomorphic to the entire world map. The algorithm proceeds by having all of the robots start at a single location with a common reference direction (the initial definition of S_m), and partitioning the unknown edges leaving the known world (edges in U) so that each robot explores independently, using the exploration algorithm described in [3, 4]. After exploring for a previously agreed-upon interval, which is defined in terms of the number of edge-traversals, the robots return to a commonly known and agreed upon location to merge their individually acquired partial world representations. Each merge process takes two partial maps and involves disambiguating possible locations between the two maps. One of the partial maps is chosen as the *base map* which is augmented with information collected by the other robot. After merging, the merged map is then shared between the robots becoming the new commonly known representation S_m and the remaining unknown edges of the new S_m are re-partitioned between the robots for the next phase of independent exploration. The algorithm repeats until the environment is fully explored.

As in [3, 4] the cost of joint exploration is defined in terms of the amount of physical motion (edge traversals) that is required in order to perform the task. In the exploration phase, in which robots explore in parallel, the cost is the maximum mechanical cost required by the robots. For the merge phase, in which only one robot performs the task, the cost is the mechanical cost associated with the moving robot. The total task cost is the sum of the cost of each exploration and merge phase.

Figure 2 illustrates the multiple robot exploration algorithm operating as two robots explore the Toronto Subway System. A graph is constructed of the subway system with vertices in the graph representing stations and edges in the graph representing tunnels connecting the stations (Figure 1). In this example both of the two robots start at a common location ('*Union Station*', the highlighted vertex in the graph) which is the initial common map shared by the robots. The partition strategy is that unexplored edges of the common map are partitioned evenly between the robots. The rendezvous schedule is 370 mechanical steps and the starting vertex is the rendezvous location, i.e., each of the robots explores for 370 mechanical steps, and then returns to the starting place to merge their world representations. During the merge, robot$_1$'s map is designated as the base map which is augmented with information in robot$_2$'s map. After merging, the augmented base map is shared between the robots as the new common map and the unknown edges of the common map is partitioned. This example involves three phases of independent exploration and three phases of coordinated merging, with the last (third) merge phase generating the full map of the subway system. In the figure the

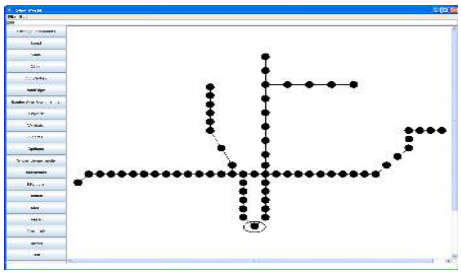
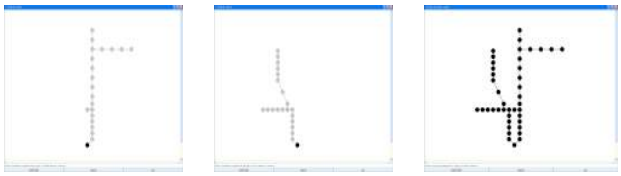


Figure 1. Graph simulating the Toronto Subway System.

Exploration-1 (370×1 exploration steps) and merge-1

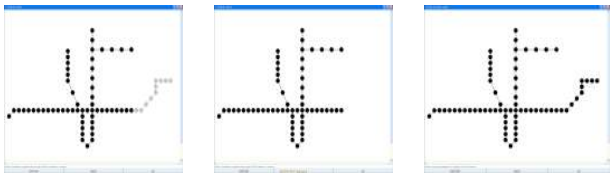


(a) r_1's map (b) r_2's map (c) Merged map

Exploration-2 (370×2 exploration steps) and merge-2

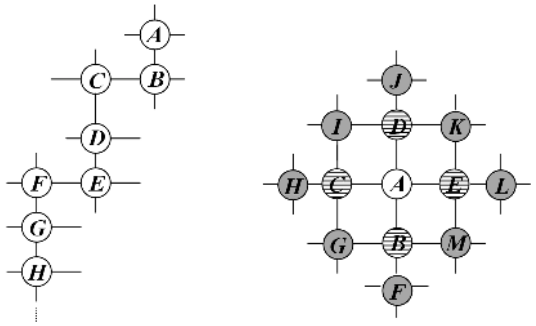
(d) r_1's map (e) r_2's map (f) Merged map

Exploration-3 (370×3 exploration steps) and merge-3
The full map is generated and the algorithm terminates



(g) r_1's map (h) r_2's map (i) Merged map

Figure 2. Multiple robot exploration on the Toronto Subway System.



(a) Closest-first exploration on lattice

(b) Breadth-first exploration on lattice

Figure 3. Exploration strategies on lattice.

darker portions represent the common map S_m and lighter portions represent the independently explored maps.

3. Breadth-first exploration

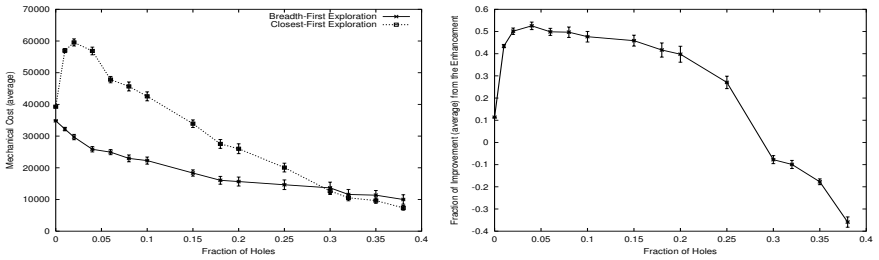
The next two sections consider enhancements that can be made to both the single robot exploration algorithm [3, 4] as well as to the multiple robot exploration algorithm [5, 14]. This section presents an enhancement that addresses the order in which new places are selected during the exploration process. In the original exploration algorithms, the robot always chooses the closest place to explore. Consider the example in Figure 3 where a lattice graph is explored. Suppose the robot starts at vertex A and chooses the unexplored edge leading to vertex B. According to the original algorithms, it will be at vertex B when it finishes validating vertex B. Suppose then it chooses an unexplored edge on vertex B leading to vertex C. When finished validating vertex C, the robot then chooses the unexplored edge at vertex C leading to vertex D, and so on. The labels in Figure 3(a) illustrate the order of exploring new vertices in the original algorithms. In disambiguating the unknown place later labelled vertex E, vertices C, B and A are potentially confusing vertices (they have unexplored edge(s) with the same degree as the unknown vertex). As a consequence, the robot has to go back to visit vertex C, vertex B, and all the way back to the furthest vertex A. In this example, as the exploration proceeds, the 'closest-first' selection of the new place to explore generates repeated traversals to the same vertices. To avoid the repeated traversals, a natural alternative to the closest-first exploration would be to use 'breadth-first' exploration, i.e., explore all unknown edges of vertex A first, then explore all unexplored edges of neighbors of vertex A, and so on, as shown in Figure 3(b) where the labels of the vertices illustrate the order and 'breadth' of the exploration. The goal is to try to maintain a compact, fully-explored region of the graph. Using this approach for this example the traversal cost during the search for vertices is reduced. Note that this is not universally true as there exist examples for which a closest-first exploration is more efficient than breadth-first exploration. Empirically, however, it appears to be more efficient as will be demonstrated for lattice graphs.

Correctness Incorporating this enhancement into the single robot exploration algorithm [3, 4] and the exploration process of the multiple robot exploration algorithm [5, 14] maintains the correctness of the original algorithms. The only difference between the enhanced algorithm and the original algorithms is the order in which the unknown places are selected and the details of exploring each unknown place are the same as the original algorithms and therefore the correctness of the original algorithms is not violated.

Empirical evaluation Experiments were conducted to examine the performance of the breadth-first exploration in the above example. In the experiments two robots explore a set of two-dimensional square lattice with holes. Two-dimensional square lattices with holes represent the type of environment that is often encountered in the interior of modern buildings. Experiments were conducted on both 20×20 and 28×28 fully connected lattices with varying number of holes (0%–40%). Holes were generated by randomly selecting vertices to remove while ensuring that the resulting graph remains connected. The breadth-first exploration is expected to work well in less heterogeneous (hole density) lattice graphs, where repeated traversals for disambiguation are more likely. Each condition was repeated 30 times, each with randomly generated holes in the lattice graph, and using both the original closest-first and the enhanced breadth-first exploration algorithm. The evaluation metric for the robot team is the mechanical cost defined in [14] and reviewed in Section 2.3. Results are shown in Figure 4, which illustrates the average mechanical cost of both the algorithms, as well as the corresponding fraction of improvement (reduction of the average mechanical cost) from the enhanced breadth-first algorithm over the original algorithm, given by

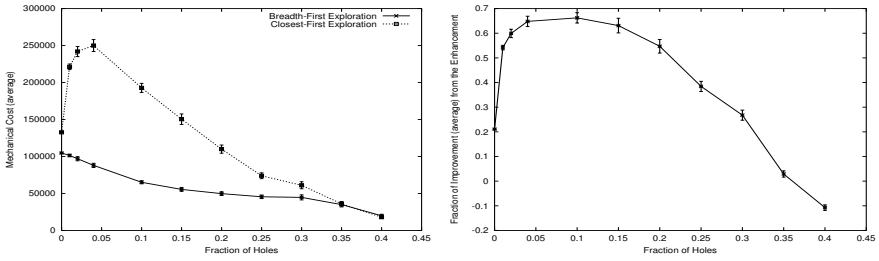
(original cost- enhanced cost) / original cost.

Corresponding standard errors are also shown. We can see that for both graph classes, the breadth-first exploration provides substantial improvement over the closest-first exploration approach for graphs with a low hole density. As the number of holes increases, the breadth-first exploration still provides an improvement but the improvement becomes



(a) Average mechanical cost - 20×20 lattice

(b) Average improvement - 20×20 lattice



(c) Average mechanical cost - 28×28 lattice

(d) Average improvement - 28×28 lattice

Figure 4. Breadth-first exploration on lattice hole graphs for two robots. Error bars show standard errors.

smaller, and eventually when the graph is sufficiently heterogeneous (with more than 30% holes), the improvement vanishes. A similar result was found with the single robot exploration algorithm.

4. Exploiting local neighbor information

This section considers the exploitation of the local neighbor structure information during the exploration. The algorithms in [3, 4, 5, 14] used the degree of a vertex (number of incident edges) to disambiguate vertices before doing mechanical search. For an unknown place, known vertices that have a different degree from the unknown place are identified as not being potentially confusing and are not visited. All vertices having the same degree as the marker-dropped place (and having unexplored edge(s)) are identified as being potentially confusing and are searched. Clearly the more information we have to describe a vertex, the more likely it is that we can disambiguate vertices without mechanical cost. In the previous work the degree of a vertex was referred to as the *signature* [10] of the vertex. Here we first extend the signature definition to include its immediate neighborhood. The neighborhood information includes the degree of each neighbor and how the neighbors are linked. Denote the extended signature of a vertex v as $sig(v)$. Then $sig(v)$ for a vertex v of degree k is an ordered set of integers

$$sig(v) = \{d_1, d_2, , ..., d_k\}$$

where d_i represents degree information of the (neighbor) vertex down edge i ($0 \leq i < k$) of vertex v. If the degree of edge i is unknown, then d_i is $null$. Note that $|sig(v)| = d(v)$. The order of the element in the signature set is based upon the normal enumeration of edges of v, but with an arbitrary initial orientation.

Retrieving the signature for new place Without extra mechanical effort, the existing neighbor information for an unknown vertex is not 'rich' enough for efficient disambiguation. For an unknown vertex where the marker is dropped, the only neighbor information available is the degree of the vertex where the robot (marker) came from. Consider the example in Figure 5, where solid portions represent the known subgraph S and dotted portions represent the unexplored edges and vertices in the real world. Assume the robot travels from vertex A to an unknown vertex X, drops its marker and senses degree 3 at X. Assuming the clockwise enumeration convention is used, then the available extended signature for X is $\{4, null, null\}$, where 4 is the degree of vertex A. Clearly the comparison of the

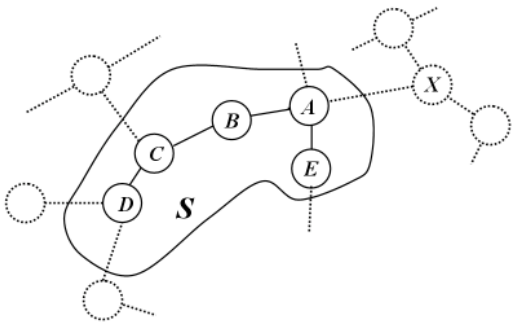


Figure 5. Retrieving neighbor information.

signatures with the known vertices in S would always result in many compatible matches. To obtain useful neighbor information, extra mechanical steps are required. For example, when a robot drops a marker at vertex X, it could also explore all of the incident edges in X (except the edge by which it entered) and sense the degree there and use this information to construct a more powerful local signature. With this extra effort, the signature of the unknown place X will be enriched to $\{4, 3, 2\}$. As new places are validated, the robot records extended signature information for all the known vertices in known graph S.

Aligning signatures In comparing the signatures of two vertices we need to align possible corresponding exits (edges) and examine if they lead to the same vertex. In comparing two arbitrary vertices where no alignment information is available we may need to take into consideration all permutations (e.g., all cyclic shifts), due to the fact that an arbitrary initial orientation is used in ordering elements (neighbors) in a vertex signature. Due to the availability of some alignment information in the exploration process, however, not all the cyclic permutations are 'valid' and need to be compared. This partial alignment information results from the fact that for a vertex v_1 in S to be the potentially confusing vertex of the unknown place, among v_1's incident edges, *it must be one of its unexplored edge(s) that corresponds to the edge incident on the unknown place which the robot used to enter this location*. For a vertex v_1 having n unexplored edges, there are at most n cyclic permutations that are 'valid' and need to be compared. Consider the known vertices C and D in Figure 5. In comparing the signatures of X with vertex D (having two unexplored edges) there are two valid cyclic permutations in their signatures, i.e., $\{4, 3, 2\}$ vs. $\{1, 3, 2\}$, and $\{4, 3, 2\}$ vs. $\{2, 1, 3\}$. On the other hand, comparing the signature of X with that of vertex C (having one unexplored edges) involves only one valid permutation, i.e., $\{4, 3, 2\}$ vs. $\{3, 2, 3\}$. Each aligned signature permutation establishes pairs of neighbors that are 'pointed to' by the aligned exits in the permutation. We call such aligned neighbors a *neighbor pair*.

Comparing signatures – radius *1* Two extended signatures $sig(v_1)$ and $sig(v_2)$ are considered compatible if the two vertices v_1 and v_2 have the same degree and, among the valid cyclic permutation(s), there exists at least one permutation such that each of the neighbor pairs in the permutation have the same degree. That is,

(i) $|sig(v_1)| = |sig(v_2)|$.

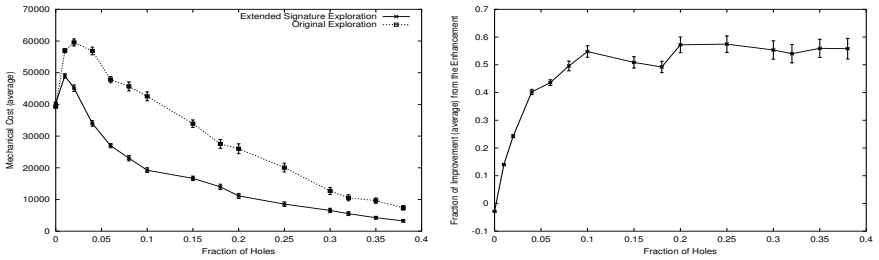
(ii) There exists a valid cyclic permutation in which for each neighbor pair $\langle d_i, d_j \rangle$ in the permutation, $d_i = d_j$.

According to the comparison rule, in the above example the signatures of X and D are not compatible and the signatures of X and C are not compatible either. Hence the unknown place X is successfully disambiguated against the potentially confusing vertices C and D.

Comparing signatures – radius *n* In the above approach the signature was extended by considering neighbors one edge traversal from the vertex being considered. The signatures were compatible if there was some permutation of the orderings of the edges such that the information known about the vertices were consistent. We can trivially extend the approach to any radius 'r'. In the radius 1 algorithm, step (ii) of the consistency check can be replaced with a recursive call to the consistency check. Note that this recursion must have some maximum limit in order to bound the radius of the search. Extra mechanical cost may be required to retrieve the topology information of the further neighbors.

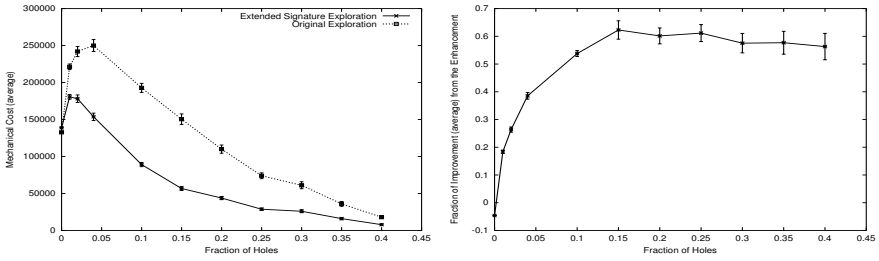
Correctness of the enhancement Incorporating this enhancement into the single robot exploration algorithm [3, 4] and the exploration process of the multiple robot exploration algorithm [5, 14] maintains the correctness of the original algorithms. The extended signature 'filters out' (disambiguates) some vertices that would have to be visited in the original algorithms. But any vertex that is filtered out cannot be a valid match due to the fact that the signature comparison process only disambiguates vertices for which there is no possible permutation of edges that allows a valid matching between the two subgraphs.

Empirical evaluation The enhancement does not change the order in which new places are explored, but it may require extra mechanical cost to retrieve the extended signatures. The same set of experiments in Section 3 were conducted to investigate the performance of the enhancement, i.e., two robots explore on 20×20 and 28×28 lattices with



(a) Average mechanical cost - 20×20 lattice

(b) Average improvement - 20×20 lattice



(c) Average mechanical cost - 28×28 lattice

(d) Average improvement - 28×28 lattice

Figure 6. Extended signature exploration on lattice hole graphs for two robots. Error bars show standard errors.

varying number of randomly generated holes. Each condition was repeated 30 times using both the original algorithm and the enhanced algorithm (using radius *2* comparison). The results for the team of two robots are shown in Figure 6, where both the average cost of the algorithms and the corresponding average fraction of improvement (cost reduction) from the enhanced algorithm over the original algorithm are reported, along with corresponding standard errors. We can see that in both graphs, when there are zero holes, i.e., when the graph is completely homogeneous, the performance of the enhanced algorithm is worse than the original algorithm. This is true because extra mechanical costs are spent at each new place for its neighbor information but the disambiguation tasks cannot benefit from the information. Even with 1% holes the enhanced algorithm shows improved performance. In this example for both graph classes, cost reductions of up to 60% are achieved. Similar results were obtained with the single robot exploration algorithm.

It is interesting to compare the breadth-first exploration, the extended signature exploration and the original (closest-first) exploration algorithm. When there are few holes, i.e., the graph is relatively homogeneous the breadth-first exploration outperforms the extended signature algorithm and the original algorithm. As the number of holes increases the improvement from the breadth-first exploration decreases whereas the improvement from the extended signature increases. Eventually when the graph is sufficiently heterogeneous, the extended signature exploration outperforms the breadth-first exploration and the original algorithm.

5. Summary and future work

This paper explores how the exploration task in single and multiple robot graph exploration can be conducted in a more effective way. We developed two enhancements that can be made to both the single robot exploration algorithm [3, 4] as well as to the exploration process in multiple robot exploration algorithm [5, 14]. One enhancement considers the order in which new places are explored and the other enhancement considers exploiting local neighbor information to help disambiguate possible confusions. Empirical evaluations for both enhancements were conducted on lattice hole graphs of varying homogeneity. Both the enhancements can produce a reduction in exploration effort and in some environments up to 60% reduction can be achieved. Graph homogeneity has a great impact on the performance of both the enhancements but in a different manner. When the graph is relatively homogeneous, adopting a breadth-first exploration outperforms the extended signature algorithm. As the number of holes increases the improvement

from the breadth-first exploration decreases whereas that from the extended signature increases. When the graph is sufficiently heterogeneous the extended signature exploration outperforms the breadth-first exploration.

The performance of the enhancements in more heterogeneous environment (e.g., lattice with more than 50% holes) remains to be further evaluated. Moreover it would be a challenging and interesting future task to integrate both the enhancements into the exploration algorithms in a strategic way. An algorithm may integrate both the closest-first exploration, the breadth-first exploration and the extend signature exploration strategies. Then during the exploration these strategies are selected in a case by case manner, e.g., based on the approximate homogeneity of the currently known graph.

Acknowledgments

The financial support of NSERC is gratefully acknowledged.

References

[1] E. Davis. *Representing and Acquiring Geographic Knowledge*. Morgan Kaufmann Publishers Inc., USA, 1986.

[2] G. Dudek, P. Freedman, and S. Hadjres. Mapping in unknown graph-like worlds. *Robotic Systems*, 13(8):539–559, 1998.

[3] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes. Robotic exploration as graph construction. Technical Report RBCV-TR-88-23, Research in Biological and Computational Vision, Department of Computer Science, University of Toronto, 1988.

[4] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes. Robotic exploration as graph construction. *IEEE Transactions on Robotics and Automation*, 6(7):859–865, 1991.

[5] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes. Topological exploration with multiple robots. In *7th International Symposium on Robotics with Application (ISORA)*, Anchorage, Alaska, USA, 1998.

[6] A. Elfes. *Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation*. PhD thesis, Carnegie Mellon University, 1989.

[7] L. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Transactions on Graphics*, 4(2):74–123, 1985.

[8] B. Kuipers. Modeling spatial knowledge. *Cognitive Science*, 2:129–153, 1978.

[9] B. Kuipers and Y. Byun. A qualitative approach to robot exploration and map-learning. In *Workshop on Spatial Reasoning and Multi-Sensor Fusion*, pages 390–404, St. Charles, IL, USA, 1987.

[10] B. Kuipers and Y. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and Autonomous Systems*, 8(1-2):47–63, 1991.

[11] B. Kuipers and T. Levitt. Navigation and mapping in large-scale space. *AI Magazine*, 9(2):25–43, 1988.

[12] S. Thrun. Robotic mapping: a survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millennium*, pages 1–35. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[13] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, USA, 2005.

[14] H. Wang. Multiple robot graph exploration. Technical Report CSE-2007-06, Department of Computer Science and Engineering, York University, 2007.