Proceedings of the IASTED International Conference
Advances in Computer Science and Engineering (ACSE 2009)
March 16 - 18, 2009 Phuket, Thailand

IT CAN BE BENEFICIAL TO BE "LAZY" WHEN EXPLORING GRAPH-LIKE WORLDS WITH MULTIPLE ROBOTS

Hui Wang
Dept. of Computer Science and Engineering
York University
Toronto, ON, Canada
email: huiwang@cse.yorku.ca

Michael Jenkin and Patrick Dymond
Dept. of Computer Science and Engineering
York University
Toronto, ON, Canada
email: {jenkin,dymond}@cse.yorku.ca

ABSTRACT
This paper describes a technique that allows mobile robots to explore an unknown graph-like environment and construct a topological map of it. The robots explore in a "lazy" fashion in which identified "hard" tasks are put off to later steps taking advantage of the fact that certain tasks often become easier as more of the world is known. Experimental validation shows that multiple robots exploring in a lazy fashion can produce a reduction in exploration effort over multiple robots exploring without prioritizing tasks based on expected effort.

KEY WORDS
Multiple robots, Graph-like worlds, SLAM, Lazy exploration, partial map merging, intelligent computation.

1 Introduction

Consider the problem of having a team of robots explore an unknown environment. Solving such a task involves dealing with issues related to the coordination of the actions of elements of the team, dealing with uncertainty in terms of locations in the environment, and prioritizing those portions of the environment that should be explored next. Here we examine a particular issue related to multiple robot exploration – the problem of how to prioritize certain tasks within the exploration process: can the cost of environmental exploration be reduced by putting off certain "hard" tasks until later in the exploration process? That is, can it be good to be lazy?

Robotic mapping is commonly referred to as *SLAM*, *Simultaneous Localization and Mapping* [1, 2]. In the majority of *SLAM* approaches the environment is represented through a metric map that captures the geometric properties of the environment (e.g. [3]). An alternative to a metric-based representation is a representation based on a topological or graph-like formalism (e.g. [4, 5, 6]). A graph-like world represents the minimal information that a robot must be able to represent in order to distinguish one place from another, and provides a useful theoretical model within which to explore fundamental limits to exploration and mapping. In [6] Dudek et al. sketched how their single robot exploration algorithm of [4, 5] might be applied to the problem of multiple robot exploration. This sketch suggested how multiple mobile agents might exploit the abilities developed in [4, 5] in order to explore in a coordinated fashion. [7] formally develops the sketch provided in [6] to the problem of multiple robot graph exploration. This extension assumes the same formalism as described in [4, 5] and populates the world with two or more robots each of which is equipped with its own unique marker (pebble). Similar to [4, 5], the marker is used to disambiguate potentially confusing locations, solving the "have I been here before" problem, i.e., the loop-closing problem.

This paper investigates how the multiple robot exploration task can be conducted in a more intelligent way, taking advantage of the fact that exploration often becomes easier as more of the world is explored. We extend the multiple robot exploration strategy described in [6, 7] by allowing the robots to explore in a lazy fashion in which harder tasks are delayed until easier tasks are completed. Empirical validation shows that multiple robots exploring in a lazy fashion can provide a reduction in exploration effort over the original algorithm, in terms of the number of mechanical steps required for multiple robot teams.

The rest of this paper is organized as follows. Section 2 reviews the world model and exploration algorithms given in the above work. The model is adopted in the lazy exploration technique described in this paper. Related work is also reviewed. Section 3 presents the lazy exploration technique. Section 4 presents evaluations of incorporating lazy exploration in the original algorithm. Section 5 concludes the work and suggests directions for future research.

2 The world model and exploration algorithm

2.1 Basic model

The World Following [4] the world to be explored is modeled as an embedding of a finite 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. No spatial metric such as distance or ori-

entation is assumed. The algorithm therefore operates on a topological representation devoid of metric information, and can be viewed as assuming worst case performance bounds for environments with noisy metric observations.

Perception The robot can identify when it arrives at a vertex. The sensory information that the robot acquires at a vertex consists of the *edge-related* perception and *marker-related* perception. With *edge-related* perception, a robot can, by following the ordering convention, determine the relative positions of edges incident on the current vertex v_i in a consistent manner (e.g. clockwise enumeration). As a result, the robot can sense the number of incident edges (i.e. degree) of current vertex, identify the edge through which it entered the vertex and assign a label (index) to each edge in the vertex representing its current local ordering of the edges. This local edge ordering is not, in general, equal to the unknown ordering specified by the embedding, but is a rotation of it. *marker-related* perception enables a robot to sense whether its marker is present at the current vertex.

Movement and marker operation A robot can move from one vertex to another by traversing an edge (a *move*). At a vertex a robot can put down its marker and it can also pick up its marker if dropped (a *marker operation*).

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

Parallelism and synchronization All of the robots operate in parallel. The parallelism does not assume a global clock, or that distance or velocity information is available. The only "clock" robots have access to is the number of edges that they themselves have traversed. To simplify issues related to synchronization it is assumed that each robot's combination of sensing and motion is an atomic operation.

2.2 Exploring a graph with single robot

The goal is to build an augmented undirected graph that is isomorphic 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. The algorithm proceeds by incrementally building a map out of the known subgraph S. As new vertices are encountered they are added to the explored subgraph and their outgoing edges are added to U, which is the set of edges that lead to unknown places and therefore must be explored.

One step of the algorithm consists of selecting an unexplored edge $e = (v_1, v_2)$ from U, and disambiguating the unexplored end vertex v_2 against the known vertices – carried out by placing the marker at v_2 and visiting all potential confusing vertices of S, looking for the marker. A vertex in S is potentially confusing if it has unexplored edge(s) and has the same degree (number of incident edges) as v_2. (Vertices in the graph-like world model are featureless except degree.) If the marker is not found at one of the potential confusing vertices of S, then vertex v_2 is not in S and therefore v_2 is added to S as a new vertex, together with the previously unexplored edge e. Other (unexplored) edges incident on v_2 are added to the unexplored edge set U. If the marker is found at a potential confusing 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, (only) the edge is added to S and removed from U. The algorithm terminates when the set of unexplored edges U is empty.

2.3 Exploring a graph with multiple robots

[6] and [7] considered using two or more robots to solve the exploration problem in graph-like worlds. Each individual robot is equipped with its own marker and the robots can only communicate with each other when they are in the same node. 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 successive iterations of the independent exploration and merging phase 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 local edge ordering (the initial definition of S_m), and then partitioning the unknown edges leaving the known world S_m between the robots. With their assigned edges each robot explores independently using the exploration algorithm described in [4]. After exploring for a previously agreed-upon interval 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. The merged map (augmented S_m) is then shared between the robots becoming the new commonly known representation S_m and the remaining unknown edges of S_m are re-partitioned between the robots for the next phase of independent exploration. The entire algorithm repeats until the environment is fully explored, that is, when there are no unexplored edges in the (merged) known map.

The challenging task of multiple robot exploration is the task of merging the partially explored environments obtained by the robots. The merging process takes two partial maps and involves disambiguating possible confusions between the two partial maps. One of the partial maps is chosen as the *base map* which is augmented with information in the other map. Similar to the strategy for single robot exploration the disambiguation of possible locations involves choosing an unmerged location in the other partial map, dropping a marker at the location and searching the base map for the marker. If the marker is found in the base map, then this location corresponds to an known place in the base map. If the marker is not found, the location is new and should be added to the base map. In either case, additional information has been added and more of the base map has been augmented.

3 Lazy exploration

3.1 Disambiguation tasks in multi-robot exploration

In the original multiple robot exploration algorithm described above the core exploration task is the task of disambiguating possible locations, which involves choosing an ambiguous place and visiting all potentially confusing locations in the known subgraph to solve the "have I been here before" problem. As more and more of the environment becomes known to the robots, the robots can make much more informed decisions as to which parts of the world are ambiguous and they can also plan much more efficient motions within their environments. Given this, here we explore the potential advantages associated with putting off more complex tasks until more of the world is known and thus the tasks become easier. That is, having the robots explore in a lazy fashion.

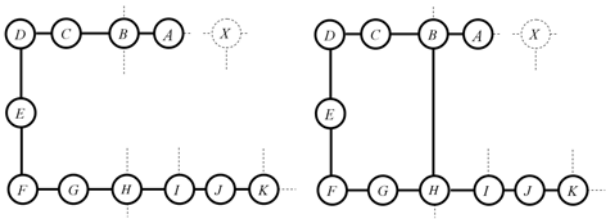
There are two stages of the original multiple robot exploration algorithm where disambiguation tasks are allocated and therefore the robots can be lazy: first, each of the robots can be lazy in terms of their efforts during the individual exploration phase, and second, the team of robots can be lazy during the merging phase.

In the independent exploration phase of the original multiple robot exploration algorithm, for each ambiguous place selected (the unknown end of an unexplored edge), the robot goes to the place, drops the marker and senses the degree of the place, identifies potential confusing vertices in the known map (based on the degree information), and then visits the potential confusing vertices for the marker. A known vertex is potentially confusing if it has unexplored edge(s) and has the same degree as the unknown place. This search is conducted in a closest-first order, and ends when either the marker is found or the search is exhausted.

In the merging phase of the original multiple robot algorithm, for each ambiguous place selected (the unmerged end vertex of an unmerged edge in the other partial map), when mechanical motion is needed, one of the robots drops the marker at the unmerged vertex and then searches potential confusing vertices in the base map for the marker. A vertex in the base map is potentially confusing if it has not been merged and has the same degree as the unmerged vertex. Note that unlike the situation in the independent exploration phase where the unexplored vertex is unknown, here the unmerged vertex is a known vertex in the other partial map and thus its degree information is known. As a result, the potential confusing vertices can be identified before traversing to the vertex. Both the disambiguation processes are summarized below:

while *there are unexplored / unmerged vertices*
- *choose one (closest) unexplored / unmerged vertex*
- *traverse to the vertex, drop marker (sense degree)*
- *identify potential confusing vertices based on degree*
- *search known map / base map for the marker*
- *augment known map / base map due to search result*

end while



(a) Disambiguating vertex X requires long path

(b) Disambiguating vertex X enjoys shorter path

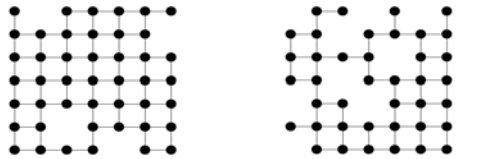
Figure 1. Growth of known graph produces different search tour length.

3.2 Delay opportunities

The main mechanical cost associated with the disambiguation task arises from the steps spent on visiting potential confusing vertices. In the original exploration algorithm each incoming disambiguation task (for an unexplored or unmerged vertex) is performed regardless of how "costly" the task is. It may be possible to reduce the overall disambiguation cost by delaying certain disambiguation task to later steps. A new vertex or edge can often be approached and eventually validated via different routes with different disambiguation costs, and as the map grows the disambiguation task may become easier. An illustrative example is shown in Figure 1, in which the solid graph represents the explored subgraph (during exploration) or the base map (during merging), and the dotted portions are unexplored vertices and edges (during exploration) or unmerged other partial map (during merging). As the known (solid) graph grows (Figure 1(b) versus Figure 1(a)), disambiguating the unknown (unmerged) vertex H requires a reduced disambiguation (search) cost, i.e., the search tour to potential confusing vertices (vertex I and K – same degree three as H) is reduced due to the newly validated "shortcut" edge (B,H). (Here as in the original work a constant mechanical cost is assumed for the traversal of each edge.)

3.3 Exploring in a lazy fashion

In the lazy approach the order of choosing ambiguous places is the same as that in the original exploration algorithm (e.g. closest-first) but the disambiguation tasks are prioritized, according to the expected difficulty of the task. Each chosen task is first evaluated. If it is judged easy it will be processed, otherwise it will be put off to later steps. Here the difficulty of the disambiguation task for an ambiguous place is evaluated based on the "length" (number of edge-traversals) of the search tour required to visit all potentially confusing vertices of the ambiguous place. The length l of a search tour is measured by the number of edges in the search tour. In Figure 1(a) the length of the search



(a) 7×7 lattice with 10% nodes removed (4 holes)

(b) 7×7 lattice with 20% nodes removed (9 holes)

Figure 2. Lattice graph with different fractions of holes.

tour for H is 10 (starting from A) whereas in Figure 1(b) the length is shortened to 5.

3.4 Deterministic and probabilistic lazy exploration

Here we present two lazy techniques. The first technique is a deterministic approach, in which the decision is made based on the direct comparison of the evaluated search tour length l of the task and a pre-defined threshold L_{max}, i.e., a disambiguation task that requires a search length that is below the threshold ($l < L_{max}$) is accepted and the task is rejected otherwise. In the deterministic approach once a task is delayed, the robot tries other tasks and the delayed task will not be considered until some other task is accepted and processed.

The second technique is a probabilistic approach in which the delay decision for a task is made probabilistically based on its search tour length l. In this approach each task is mapped into a value in the range of [0:1] according to the function

$$p = \epsilon + (1 - \epsilon)e^{-kl}$$

where $k > 0$ and $\epsilon > 0$ are tuning parameters. The robot chooses random numbers in the range of [0:1] and selects the first task for which a random number r has the property $r \leq p$. Assuming the fairness of the random number generator, the shorter the search tour length the larger value of p and the more likely that the task will be performed.

4 Lazy exploration and its effects in multiple robot graph exploration

Lattice hole graphs As in [7], the evaluation is conducted on two-dimensional square lattices with small numbers of holes – a type of random graph that represents the environment often encountered in the interior of modern buildings. Examples are shown in Figure 2. The graphs were generated by starting with a complete two-dimensional lattice (a grid) and deleting a specified fraction of randomly selected nodes such that the graph remains connected. Vertices in the graphs represent distinctive locations (e.g. rooms) in buildings and edges represent paths between the locations (e.g. corridors).

Evaluation Metric As in the original algorithm, here mechanical cost (amount of edge-traversals) by the robots is the main performance concern. As in the original algorithm, a unit cost is assigned for the traversal of each edge, and in the exploration phase in which robots explore in parallel, we take the larger mechanical cost required by the individual robots in the exploration phase (the mechanical steps by the robot that made more traversals) which represents the limiting cost of the exploration phase. For the merging phase in which 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 merging phase.

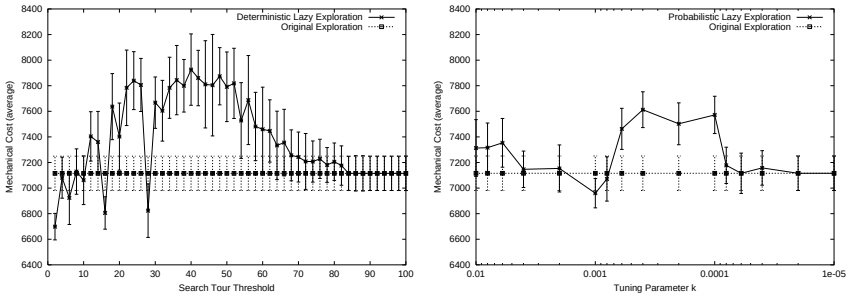
4.1 Lazy exploration in exploration phase

We first evaluate incorporating lazy exploration in the independent exploration phase of the algorithm. During independent exploration, the robot chooses a (closest) unexplored location, traverses to the location and senses the degree there, identify the potential confusing vertices based on the sensed degree, and then computes the search tour length l. Then it decides to accept or delay the chosen task based on the deterministic or probabilities criteria. If it decides to delay the task, it proceeds to choose another task (unknown location). This is outlined below:

while *there are unexplored vertices*
- *choose one (closest) undelayed such vertex*
- *traverse to the location and sense the degree there*
- *identify potential confusing vertices based on degree*
- *compute the search tour length l*
- *decide (deterministically or probabilistically)*
- ***if*** *decide to accept*
  . *drop the marker and search for the marker*
  . *augment the known map accordingly*
- ***else***
  . *delay the task until some other task is accepted*

end while

Experiments were conducted in which two robots explore a set of sample two-dimensional square lattices of size 30×30 with 20% holes. 50 input graphs were generated (each with randomly located holes) for each condition. Each input graph was explored using both the deterministic and the probabilistic lazy approaches, as well as the original exploration strategy described in [6, 7]. In all cases the robots start at a randomly chosen location in the graph and use a fixed rendezvous interval of 400 edge traversals (see [7] for discussion of the effects of different rendezvous intervals). For the deterministic approach, each input graph was explored with different sample length thresholds L_{max}. For the probabilistic approach, each input graph was explored with different values of the tuning parameter k ($\epsilon = 0$ is used for these tests). Performance



(a) Deterministic lazy exploration (b) Probabilistic lazy exploration

Figure 3. Lazy exploration in the independent exploration phase. Error bars show the standard errors.

of the deterministic lazy approach versus the original approach is shown in Figure 3(a), which shows the average cost of the lazy approach at each sample threshold L_{max}, together with standard errors. Average cost and standard errors of the original approach on the same set of input graph are also shown. Note that for the deterministic lazy approach when the threshold L_{max} is sufficiently large, all disambiguation tasks are accepted and the lazy exploration produces the same performance as the original strategy. Performance of the probabilistic lazy approach versus the original approach is shown in Figure 3(b). The average cost of the lazy approach at each sample k, together with standard errors are plotted. Performance of the original approach on the same set of input graph is also shown. Note that when k is sufficiently small, p approaches 1 and all tasks are accepted and therefore the lazy exploration produces the same performance as the original strategy.

Results show that in most cases the original algorithm outperforms the lazy exploration algorithm. This should not be of particular surprise as determining the value of expected search tour length can involve traveling to a chosen unexplored vertex (to sense the degree information of the vertex – before potential confusing vertices can be identified and search tour length computation and decision making can be done). In the lazy algorithms this cost is often wasted (i.e. when the task is delayed).

4.2 Lazy exploration in the merging phase

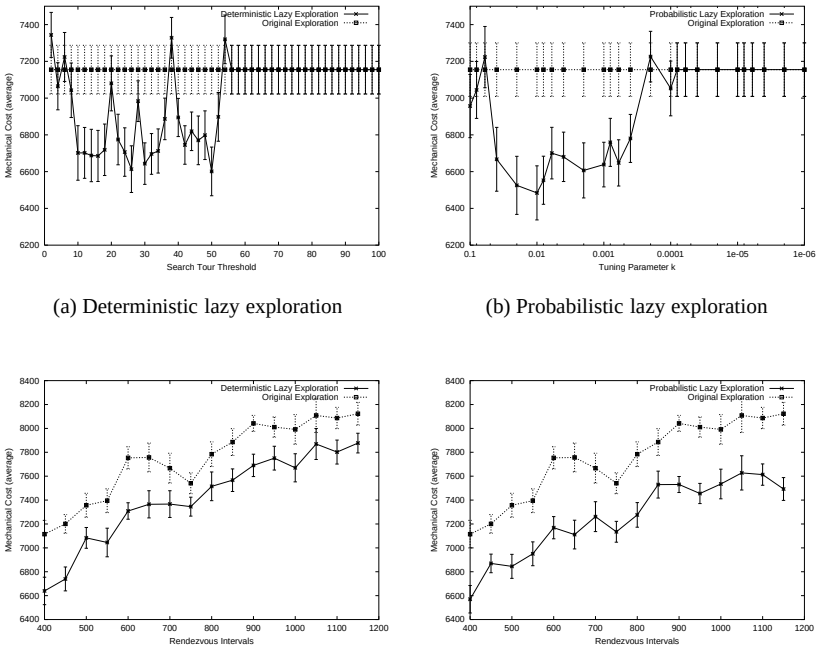
Unlike the situation in the independent exploration phase, in the merging phase for each ambiguous (unmerged) place – the known place in the other partial map – the robot retrieves degree information and hence an estimate of the task (search) cost by examining the other partial map. The robots keeps on selecting unmerged places without mechanical motion until some task is accepted.

while *there are unmerged vertices*
- *choose one (closest) undelayed such vertex*
- *identify potential confusing vertices based on degree*
- *compute the search tour length l*
- *decide (deterministically or probabilistically)*
- ***if*** *decide to accept*
  . *traverse to the location*
  . *drop the marker and search for the marker*
  . *augment the base map accordingly*
- ***else***
  delay the task until some other task is accepted

end while

A similar set of experiments to those described above were conducted, i.e., using both lazy approaches and original strategy, two robots explored a set of 50 input graphs each of which is a 30×30 two-dimensional lattice with 20% of the vertices randomly removed. Results are shown in Figure 4(a)–(b). Both the deterministic and the probabilistic approaches produce reduced average cost over the original algorithm for many values of the tuning parameters k and L_{max}. For the deterministic algorithm the majority of the threshold values give rise to better performance. For the probabilistic algorithm the majority of the tuning parameter values produce better performance. This improved performance is due to the fact that no mechanical cost is required before accepting a task.

In the above experiments a fixed rendezvous intervals was used for the robots. To further evaluate the effects of lazy exploration in the merging process, another set of experiments was conducted in which the 'best' threshold and tuning parameter from the above experiments (L_{max} = 50 and k = 0.01) were used and the same set of lattice hole graphs were explored using both algorithms with varying rendezvous intervals. Similar to the above experiments, each condition (rendezvous interval) was repeated on 50 input graphs. The results are shown in Figure 4(c)–(d). From the results we can see that for the particular threshold L_{max} and parameter k, there is an improvement in performance for the majority of the rendezvous intervals.



(a) Deterministic lazy exploration

(b) Probabilistic lazy exploration

(c) Deterministic lazy exploration

(d) Probabilistic lazy exploration

Figure 4. Lazy exploration in the merging phase. Error bars plot standard errors.

5 Conclusion and future work

This paper describes a technique whereby mobile robot(s) explore a graph-like world in a lazy fashion in which specific "hard" exploration tasks are put off to later steps, taking advantage of the fact that certain tasks can become easier as the known graph grows. Empirical results show that the mechanical cost required in estimating the difficulty of the disambiguation tasks has great impact on the performance of lazy exploration.When incorporated into the merging phase of the multiple robot algorithm in which the estimate of task cost and hence the decision to accept or delay the tasks can be made without mechanical motion, and both the deterministic and probabilistic approach present in this paper produce reduction in the cost over the original algorithm for most of the cases. The same is not true in the independent exploration phase where the mechanical cost associated with estimating the task cost outweighs any improvement associated with lazy exploration.

The lazy exploration technique in this paper represents our preliminary efforts towards intelligent computation in robotic topological exploration and mapping, and suggests a number of possible directions for future research. Current research includes further evaluation of lazy strategy in the merging phase, conducted on more environments (graphs), and with larger groups of robots. Plan for future work also includes developing heuristics to improve the unsatisfactory performance of lazy strategy in the independent exploration phase.

References

[1] S. Thrun, Robotic mapping: a survey, *Exploring Artificial Intelligence in the New Millennium*, (CA: Morgan Kaufmann Publishers Inc., 2003) 1-35.

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

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

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

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

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

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