Competitive Robot Mapping with Homogeneous Markers

Xiaotie Deng and Andy Mirzaian
Department of Computer Science, York University
North York, Ontario, Canada, M3J 1P3
{deng, andy}@cs.yorku.ca.

Abstract: We consider the robot exploration problem of graph maps with homogeneous markers, following the graph world model introduced by Dudek et al. [DJMW]. The environment is a graph consisting of nodes and edges, where the robot can navigate from one node to another through an edge connecting these two nodes. However, the robot may not distinguish one node (or edge) from another in this unknown graph. All the nodes (edges) look the same. However, at each node, the robot can observe a consistent local relative orientation of its incident edges, that is, a cyclic order of edges incident to the node. To assist the robot's task of mapping the environment, it can put homogeneous (i.e., identical) marks on nodes or edges which can be recognized later.

The total number of edges traversed when constructing a map of the graph is often used as a performance measure for robot strategies. However, since the graph is unknown, a strategy may be efficient in one situation but not in others. Thus, there is a conceptual question about what is an optimal strategy. In this paper, we apply the competitive analysis method [ST,BRS,PY] for robot explorations. In particular, we compare the cost for constructing a map with the cost for verifying the same map and call their ratio by competitive ratio, an approach initially applied to a similar mapping problem by Deng and Papadimitriou [DP]. That is, we call a strategy optimal if it minimizes the worst-case ratio (over all allowable embedded graphs) of the total number of edges traversed when constructing a map of the graph to the optimum number of edges traversed in verifying the correctness of a given map of the same graph. If this competitive ratio is bounded above by a constant, we say the strategy is competitive. One of the main results of this paper is to show that a competitive strategy exists for mapping planar embedded graphs when the robot has n homogeneous markers at its disposal (throughout the paper n denotes the number of vertices and m denotes the number of edges of the unknown graph). We also show that in some alternative situations no competitive strategy exists.

1 Introduction

The study of robot navigation has several different facets which are important to robot performances in real world environments. One problem often discussed in the field of theoretical robotics is how fast a robot can move from the current position to the destination [Ca,CR,CDRX]. Usually, the robot is assumed to move perfectly along a planned route. In practice, however, it is often an important problem to deal with accumulative errors in robot navigation, which may cause the robot to lose track of its position in the real world. Several different approaches are suggested to relate the robot position with the external features of the environment using a map [GMR,BCR,BRS,KB,LL]. This leads to the task of mapping for robot navigation, i.e., learning the cognitive map from observations, as summarized by Kuipers and Levitt [KL]. Many researches have been done on map construction (or mapping) by robots for different external environments [DJMW,DP,DKP,RS].

Naturally, one major class of maps used in robot navigation is geometric. However, an autonomous robot still has to decide how to utilize the map and match it with the enormous amount of observed geometric information to make its own decisions. Alternatively, qualitative maps, such as topological graphs, are also used to model robot environments [DJMW,DP,RS]. This approach often focuses on a small set of landmarks in the environment, which requires much less information comparing with geometric models. Consequently, it simplifies the task of robot decision makings. Kuipers and Levitt proposed a spatial hierarchical presentation of environments consisting of four levels: *sensorimotor* level (a robot uses sensors to detect local features of the environment); *procedural* level where the robot applies its knowledge of the world to find its place in the world and to follow specified routes; *topological* level which describes places and their connecting paths, usually with the graph model; and *metric* level which includes necessary geometric information related to the topological representation [KL, also see KB,LL]. These approaches, neither purely metric nor purely qualitative, leaves certain features of the environment out but keeps necessary information helpful for robot motions.

The robot's perceptions of the world can also be different as a result of different built-in

sensors it carries. In many situations, it is assumed that nodes or edges traversed previously can all be distinguished. In contrast, it is assumed in [RS] that nodes are divided into a small number of classes, for example, white and black colors, and can only be recognized as such. They take the approach of Valiant learning model [Val] to obtain algorithms which correctly infer the environment with high probability. Dudek et al. [DJMW] apply the world model introduced by Kuipers and Levitt to a specific situation in which no global orientation information is possible. Globally, Dudek et al. divide the world into places represented by nodes in a topological (i.e., embedded) graph and use an edge between two nodes to represent a connecting path between the corresponding two places. They assume in this setting nodes may have ambiguous identifiers, and hence assume the robot may not be able to distinguish nodes from each other. Besides the graph representation for topological structures of the world model, a special local geometric feature is added: At each node, its incident edges are given a certain cyclic order. This emulates the fact that, at the crossroads, paths form a cyclic order because of the local planar geometric nature of the earth surface. Dudek et al. show that it is impossible to learn the graph in general if the robot uses only information collected under the above restriction. For instance, this happens when every node in the graph, representing the world, has the same number of incident edges. On the other hand, they show that in a total of $O(|V| \times |E|)$ traversals of edges, the map can be constructed with the help of a single marker which can be put down on nodes and picked up by the robot [DJMW].

In this paper, we follow the world model introduced by Dudek et al. [DJMW]. We will apply competitive analysis for the performance measurement of different strategies. The concept of *competitive analysis* was first introduced to deal with unknown future events of online problems [ST]. The main idea is to evaluate how good a strategy operating under incomplete information is by comparing it with the optimal solution with complete information. It has also been used as a measure of information-efficiency of algorithms for robot navigations and explorations [BCR, BBFY, BRS, DKP, DP, FFKRRV, Kl, KP, KRR, PY]. In this approach, exploration strategies S are evaluated by examining the (worst case) ratio of the cost of building the map, where we initially know nothing about the world, to the cost of verifying the

map, where we have a map of the world and the initial position-orientation of the robot in the map, but still want to verify the correctness of the given information [DP,DKP]. Over all the possible strategies, in principle, we want to obtain the one which has the minimum competitive ratio.

1. The competitive ratio of a strategy S is:

$$\max_{M \ a \ map} \frac{S(M)}{V(M)},$$

   where $S(M)$ stands for the total number of edge traversals by strategy S for the mapping problem and $V(M)$ stands for the minimum number of edges traversed in verifying map M.

2. Our optimization problem is to find a strategy which solves the following min-max problem.

$$\min_{S \ a \ strategy} \max_{M \ a \ map} \frac{S(M)}{V(M)}.$$

In Section 2, we formally introduce the world model and precisely define the competitive ratio as applied to the robot exploration problem. With the terminology of competitive analysis, the main result of Dudek et al. is a mapping strategy of competitive ratio $O(n)$. In Section 3, we show that if the robot has a single marker, then the result of Dudek et al. is asymptotically optimal within a class of fairly reasonably restricted strategies (namely, the class of Depth-One Search strategies, to be specified later). We show this by proving a lower bound of $\Omega(n)$ for the competitive ratio, and thus establish a tight bound of competitive ratio $\Theta(n)$. It is a nontrivial task to design a general optimal algorithm for verifying a map (a standard depth first search cannot accomplish this since the robot may not distinguish new and old nodes in the real environment). However, for the particular subclass of graphs used in the lower bound proof, we are able to evaluate the number of traversals needed for verifying a map. In this section we also show that no general strategy with a single marker for mapping general embedded graphs can achieve a competitive ratio better than $\Omega(\log n)$. We further discuss some results for mapping planar embedded graphs with a single marker. In Section 4, we discuss mapping algorithms with multiple markers. For embedded planar

graphs, we propose a competitive strategy (i.e., with constant competitive ratio) to map an unknown embedded planar graph using n identical markers. We conclude the paper in Section 5 with discussion and remarks.

2 The World Model and Competitive Analysis

Based on our discussion of the world model introduced by Dudek et al. [DJMW], the world is an undirected graph $G = (V, E)$ embedded on a (not necessarily planar) surface. Thus, at each node, edges incident to the node have a local planar embedding (Figure 1). Location orientations of edges at a node provides a natural cyclic order of the edges incident to a node. When we have specified an edge incident to the node as the reference edge, we can name other edges incident to this node with respect to the reference edge.

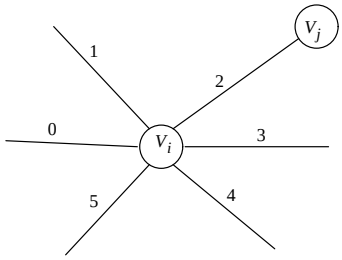
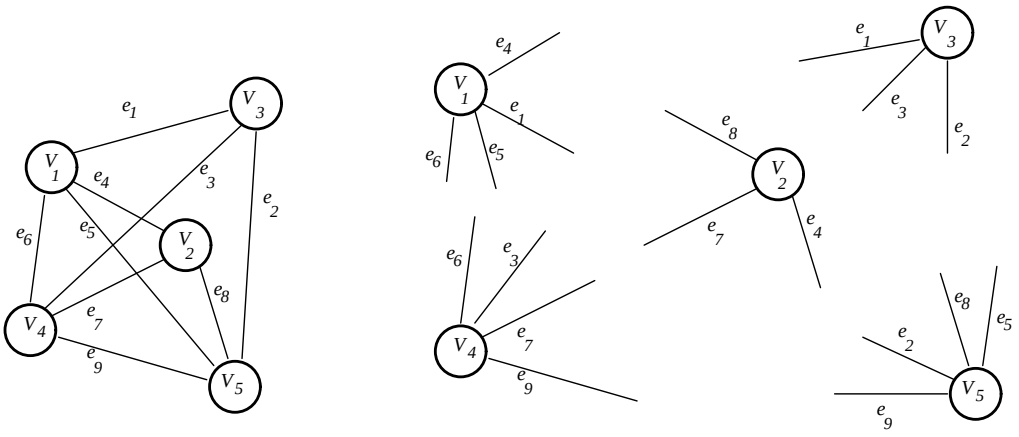


Figure 1: A cyclic order of edges incident to a node.

The Robot's Map of the Graph World. In the graph world model, nodes usually correspond to landmarks in the real world. To deal with general situations where landmarks may look similar by the robot's sensors, Dudek et al. assume the worst possible situation: nodes in the graph are indistinguishable to the robot. Therefore, the complete map of the graph is a triple $(V, E, \mathcal{S})$, where V and E are the node set and the edge set of the graph, and $\mathcal{S}$ is the collection of the local planar embeddings of edges incident to each node (Figure **??**).

Note that, we intentionally give an example, where, for several nodes, the cyclic orderings of incident edges are different from those in the topological graph to emphasize the general situation where local orientations of edges at each node can be arbitrary. In fact, given any graph $G = (V, E)$, given any set $\mathcal{S}$ of cyclic orders of edges incident to a node, there is a



(a) The graph

(b) Local embeddings of edges

Figure 2: A map.

surface on which the graph can be embedded such that the local planar embedding of edges incident to each node follows the cyclic order of $\mathcal{S}$ [HR]. Even in the real world, these may happen because tunnels and bridges can create generally embedded surfaces.

Robot Navigations with the Map. Once the map is given, the current location of the robot is matched to a node u_0 on the map, and a path from the robot location is matched to an edge e_0 incident to u_0 on the map, the robot can match all the paths at its current location to edges on the map incident to u_0. If the robot moves to another location through one of the paths, it knows which node matches the next location it reaches. The edge that leads the robot to the next location becomes its reference edge at the new location. From the local embedding of edges incident to the new node, with the help of the reference edge, it can again match edges incident to this new node with paths from its new location. This allows the robot to use the map navigating from one location to another location until its destination.

The mapping Problem. The mapping problem requires the robot to construct the world map through its observations while navigating in its environment. Dudek et al. show that, in general, it is not possible to obtain the map of the above graph world when no external help is given because of some symmetric situations, but is possible to construct the map with a single

marker which can be put down on nodes, and recognized and picked up by the robot. An algorithm using $O(|V||E|)$ traversals of edges for the mapping problem with a single marker is presented by Dudek et al.

We discuss several different situations in this paper:

1. Lower bounds for the mapping problem with a single marker.

2. A competitive algorithm for the mapping problem of planar graphs with $|V|$ homogeneous markers.

Competitive Analysis. We focus on the efficiency issue of strategies for the mapping problem. In particular, we (following that of Dudek et al. [DJMW]) define the (mechanical) cost of a strategy to be the total number of traversals on the edges of the graph and are interested in mapping strategies of small cost. However, since the graph is unknown and information is collected as the robot proceeds, the cost of mapping depends on both the strategy and the unknown graph, a typical problem in the area of *optimization with incomplete information*.

The *competitive analysis* method dealing with unknown future events of online problems [ST] evaluates strategies operating under incomplete information by comparing the solution of a strategy for an online problem with the optimal solution with complete information. It is also used as a measure of information-efficiency of algorithms for robot navigations and explorations [BCR, BBFY, BRS, DKP, DP, FFKRRV, Kl, KP, KRR, PY]). In this approach, exploration strategies are evaluated by examining the ratio of the cost of building the map (where we initially know nothing about the world) to the cost of verifying the map (where we have a map of the world and the initial position-orientation of the robot in the map, but still want to verify the correctness of the given information.) Hence, in our world model, the competitive ratio of a strategy is defined as the maximum ratio, over all allowable graphs, of the number of traversed edges for establishing the map to the minimum number of edges traversed for verifying a map of the same graph. A mapping strategy with the competitive ratio c will always traverse a total number of edges which is no more than c times the number of edges traversed in verifying the map. The smaller the competitive ratio, the smaller the extra factor for traversals in establishing a map.

Therefore, we are interested in finding a strategy that minimizes the competitive ratio. Note that, for robots with perfect sensors, mapping is the same as the graph traversal problem. However, without perfect sensing, mapping a graph (or even verifying a given map of the graph) becomes different from the traversal problem of the graph. We may not know the graph even after all the edges are traversed.

The Footprint Model. We introduce this as an auxiliary or intermediate model. The results established for this model will later be used to establish our results for the marker model (see Section 4). For the mapping problem in the footprint model the robot leaves an unremovable trace on nodes and edges it has passed. This is equivalent to having $|V| + |E|$ homogeneous nonerasable markers, one for each node and edge.

An Example. Below we propose a mapping strategy in the footprint model with $O(|V| \times |E|)$ edge traversals. This strategy is adapted from the solution of Dudek et al. for the mapping problem with a single marker.

Initially, the robot is placed at an arbitrary node u in the graph. It can see all the edges incident to u. The spatial orientation of incident edges gives their circular order around u. When the robot first reaches a node v adjacent to u, it can remember the edge $e = (u, v)$ from which it arrived at v. Therefore, edges incident to v can be circularly ordered with respect to edge (u, v). If the robot moves back along (u, v), it knows that it comes back to node u and its map shows the local orientation of edges it has made for node u before it went to node v. However, if the robot comes back to u from an unknown edge, it may not distinguish this node from other nodes with the same number of incident edges. Thus, the local maps of a node are different when the robot reaches that node from two different edges and the robot may not know how to match the two linear orders of incident edges from these two maps.

To deal with these difficulties, at each stage, the robot has explored a part of the graph. It draws a *partial map* $P = (V', s, E', E'')$, where $V' \subset V$, $E' \subset E \cap V'^2$, E'' are edges with just one endpoint known to belong to V'. Edges in E' are all the edges the robot has traversed and mapped up to this point. The robot is currently at node s, and knows how to match edges incident to s with edges in the partial map it has at hand. Thus, when it moves out of node s along an edge in E', it knows which node it will arrive by consulting the (partial)

map. The ability of remembering the edge, with the help of the map, enables the robot to match the edges it sees at the node with the edges in the map. It is easy to conclude that the robot can go to any edge in E' and any node in V' correctly by comparing with the map. However, the robot does not know where the edges in E'' will lead to. When E'' is empty, we know the whole graph.

Suppose now E'' is not empty. The robot will take a closest edge $e = (a, x) \in E''$, with $a \in V'$, and move to a and then move to the other endpoint along e, and continue until a node t already with footprint is reached. Denote this chain of edges we just visited by S. Denote by (s, t) the last edge along S. We want to find out which node of V' is t. The robot can come back along S to node a. Since a and t are the only two nodes in P incident to an increased number of foot-printed incident edges, the robot can come back to a along S and go through all the nodes in V' along edges in E' to find out which node in V' is t. Then the partial graph can be updated by $V' \leftarrow V' \cup \{V(S)\}$, $E' \leftarrow E' \cup \{E(S)\}$.

We notice that this algorithm may, in the worst case, exhaust all the nodes to validate a single edge. Thus the total number of traversals is $O(mn)$. Similar results using the same approach are obtained for the marker model in [DJMW], with a single marker. Both algorithms give a competitive ratio $O(n)$ since any map verification strategy (and hence the best once) obviously requires at least $\Omega(m)$ edge traversals. (Remark: whether there is any map verification strategy that makes at most $O(m)$ edge traversals is an entirely different issue.)

3 Mapping General Graphs with One Marker

In this section, we show an $\Omega(\log n)$ lower bound on the competitive ratio for the general mapping problem defined in the previous section, with a single marker. In addition, we prove that the upper bound $O(n)$ on the competitive ratio for the single marker model [DJMW] is tight under a restricted class of strategies. The lower bound is obtained by considering a special class of graphs with $O(n)$ edges, for which mapping takes $\Omega(n^2)$ traversals, and verification takes only $O(n)$ traversals.

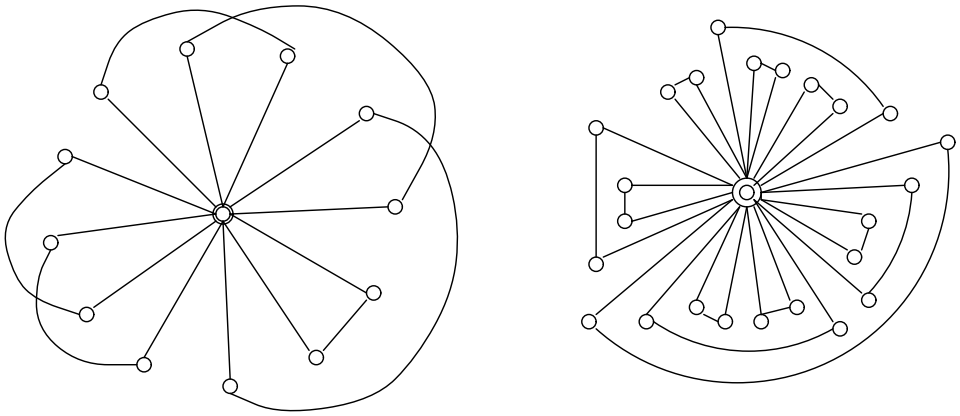
The restriction is that our strategy constructs a partial map as we explore. For a node u

already named in the map, let e_i be the i-th edge (in clockwise order) from a reference edge $ref(u)$, to which u is incident. We restrict our strategy to choose one such edge e_i, for which the other end node is not known on the partial map yet, (it is either not in the partial map or it is in the partial map but we don't know which one it is). We can put our marker at the unknown endnode v of e_i, then move back to the known endpoint u and traverse nodes in the partial map to find if v (the node with the marker) is already in the partial map. If it is, we add this edge to the partial map. Otherwise, we can give a new name to v and add both the edge (u, v) and node v to the partial map. We call such a strategy *depth-one search.* The algorithm of Dudek et al. is a depth-one search [DJMW].

We can define another class of more relaxed strategies similar to *depth-one search*: We allow the robot to put the marker on an unknown end of an edge already named (the same as depth-one search). The robot then can come back the the adjacent known node, it does not need to traverse all the named nodes in the partial map. Of course, if the robot finds the marker, it can establish the edge between u and v. But if it does not find the marker before exhausting all known nodes in the partial map, it can remember nodes in the partial map which cannot be the other endnode of the edge e_i. Then the robot can probably try it later after exploring some other edges. Obviously, all depth-one strategies fit into this class. We will call this class *relaxed depth-one strategy.* We will show later, this class is more powerful than *depth-one strategies* for planar graphs.

To establish the lower bounds we use a special subclass of embedded graphs we call star-shape graphs (see Figure 3).

We construct a star-shape graph of $2n + 1$ nodes, with one node of degree $2n$, called the *center*, and $2n$ nodes of degree two, each called a *branch node.* The edges whose both endpoints are branch nodes clearly form a perfect matching among the $2n$ branch nodes. (A perfect matching is a subset of the edges so that each vertex is incident to exactly one edge in the subset. The two end-points of each edge in the matching are called matched pairs or mates.) To construct the map of such a graph is equivalent to find this perfect matching. In our lower bound argument for the mapping problem, we let the adversary select this perfect matching in such a way to maximize the robot's mechanical cost.



(a) nonplanar **(b) planar**

Figure 3: A star-shape embedded graph.

We denote the center node by v_0. At the center the robot can obtain the cyclic order of incident edges according to the local surface at this node. An arbitrary edge can be used as the reference edge at node v_0. We name this edge e_1. Other edges are named as e_i, $2 \leq i \leq 2n$ in the clockwise order around v_0. Then, we name the branch nodes according to the circular order of edges incident to the center v_0, and denote them by v_i, $i = 1, 2, \cdots, 2n$, where $e_i = (v_i, v_0)$. Each branch node v_i, is matched to another branch node, which is called its mate and denoted by $M(i)$. To determine the perfect matching is equivalent to determine, for each node v_i, $i = 1, 2, \cdots, 2n$, its mate v_j. That is, to establish the correspondence $M(i) = v_j$. Obviously, the robot can tell the center v_0 from other nodes by its distinguished degree $(2n)$ while others have degree 2. Initially, with the above restriction on the unknown graph, the components of the partial graph $P = (V', s, E', E'')$ are $V' = V$, $s = v_0$, $E' = \{(v_0, v_i) : 1 \leq i \leq 2n\}$, and $E'' = \{(v_i, M(i)] : 1 \leq i \leq 2n\}$. What is still unknown is which node v_j, $1 \leq j \leq 2n$, does $M(i)$, $1 \leq i \leq 2n$, correspond to. For a general partial graph, the unknown edges are a subset of $\{(v_i, M(i)) : 1 \leq i \leq 2n\}$. When a *relaxed depth-one search* algorithm tries to determine an unknow endpoint of an edge, it must be one of $M(i)$, for some vertex v_i in the partial map. When traversing nodes in the partial map to find which node is $M(i)$, The branch nodes are indistinguishable from each other except for the one that contains the marker and the node the robot is currently located at. Therefore, each time the robot at a branch node

probes to see whether the current node contains the marker is equivalent to the test whether $M(i) = v_j$, for some values of i and j. The reason for the latter is that for a branch node there are essentially two ways of getting there: either through the center, in which case the current node is known as v_j, for some j, or through its known matched node v_i, in which case it is known to be $M(i)$. When we come to the node in one of these ways and drop the marker, and later on if we come to a node through the alternative way and probe to see whether the marker is there, that is like testing the equality of the form $M(i) = v_j$.

The algorithm of Dudek et al. [DJMW] does the following for this particular case. For each $i = 1, 2, \cdots, 2n$, we move from v_0 to v_i, then to $M(i)$. Put the marker at $M(i)$ and move back to v_0 via v_i (in the reverse order). This allows the robot to maintain the original view of the edges. Next, probe each of v_j, $j = 1, 2, \cdots, 2n$ until the marker is found. The adversary reveals the marker only when the node is the only remaining possible location of the marker. Therefore, to obtain $M(i)$, it takes $2n - 2i$ tests, $1 \leq i \leq n$, which totals $\Omega(n^2)$ traversals. To prove this holds for all possible *relaxed depth-one search* algorithms, we introduce a graph theoretical lemma.

Lemma 1 [Bol, Theorem 2.5, page 61] *If a graph on $2n$ nodes has a unique perfect matching, then it cannot have more than n^2 edges.*

Thus, for relaxed depth-one search algorithms, we start with a complete graph K_{2n}, representing all the possible connections between the branch nodes in the graph. Each test, whether $M(i)$ is the same as v_j for some i, j, removes one edge from the graph, except if the removal will eliminate all the perfect matchings in the graph. The above lemma states that we cannot determine a unique matching until there are only n^2 edges left. The total number of edges in K_{2n} is $n(2n - 1)$. Therefore, we need at least $n(n - 1)$ tests before we can decide that unique matching in the graph we try to map. □

Now suppose we are given a map of the star-graph and want to verify it. Suppose edge (v_i, v_j), between two branch nodes is in the map. To verify its existence in the graph we need to verify the matching $M(i) = v_j$ in the graph. To do this the robot moves from v_0 to v_i, then to $M(i)$ and puts the marker at $M(i)$. Moving back to v_i then to v_0 and then to v_j, the

robot checks if the marker is there, then moves back to v_0. Thus, we traverse six edges for verifying a matched pair. The total verification cost is thus $6n$. The ratio is $\Omega(n)$.

Theorem 1 *The competitive ratio of (relaxed) depth-one strategies for mapping general embedded graphs with a single marker is* $\Theta(n)$.

Proof: The above lower bound (i.e., the ratio $\Omega(n^2)$ for mapping over $O(n)$ for verification) for the subclass of star-shaped graphs is clearly a lower bound over all embedded graphs for relaxed depth-one strategies. On the other hand, Dudek et al. [DJMW] give a depth-one strategy with $O(|V| \times |E|)$ traversals for the mapping problem on any embedded graph. Since any strategy for verifying a map takes at least $\Omega(|E|)$ traversals. This gives an upper bound $O(|V|)$ for the competitive ratio. Therefore the matched lower bound and upper bound holds for both depth-one strategies and relaxed depth-one strategies.

We also have the following lower bound for mapping planar graphs using depth-one strategies.

Theorem 2 *The competitive ratio of depth-one strategies for mapping embedded planar graphs with a single marker is* $\Omega(\log n)$.

Proof: We consider a planar star-shape graph with $2n$ branch nodes (see Figure 3(b)). Note that, because of the planar embedding, the matching edges between branch nodes have the nested (or parentheses) property, i.e., any two such edges $[v_i, M(i)]$ and $[v_j, M(j)]$, viewed as cyclic intervals, are either disjoint or one contains the other. Consider an arbitrary depth-one strategy using a single marker to construct the map of such a graph. So, the task of the robot is to compute the matching function between branch nodes. We let the adversary determine the matching. Let $T(n)$ denote the cost of such a mapping. Suppose $M(i) = v_j$ is the first matching determined by the strategy. (We will let the adversary choose $M(i)$ as explained below.) Let the number of branch nodes nested with the interval $[v_i, M(i)]$ be $2k$ (it must be even), and hence, the number of other branch nodes (excluding v_i and v_j) is $2n - 2k - 2$.

In the entire sequence of edge traversals suppose $f(n)$ edge traversals were needed to infer this first matching edge. Remove this $f(n)$ edge traversals from the sequence. The

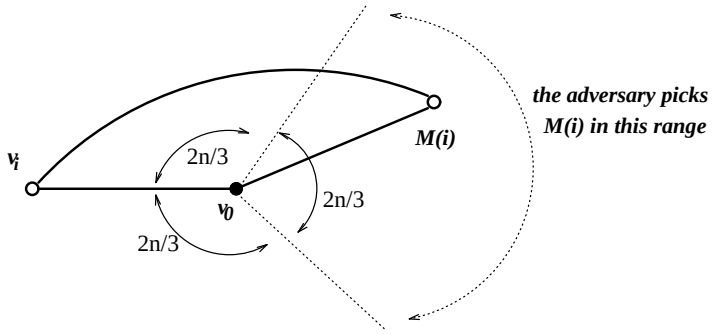


Figure 4: Proof of Theorem 2.

remaining sequence can be partitioned in two parts (possibly removing useless or redundant traversals), with one sequence corresponding to establishing matching edges nested within the interval $[v_i, M(i)]$, and the other sequence corresponding to establishing matching edges not nested in the latter interval. Because each of these two remaining subproblems can be chosen independently by the adversary, we observe the recurrence $T(n) \geq f(n)+T(k)+T(n-k-1)$.

Now let us explain how the adversary will choose $M(i)$ in the first place. The idea is suggested in Figure 4. The $2n$ branch nodes are partitioned into three (roughly) equal intervals cyclically, starting from v_i and going clockwise. Each of these disjoint intervals has approximately $2n/3$ branch nodes. The adversary will select $M(i)$ in the middle interval. Because of the parity forced by the nesting property, there are approximately $n/3$ candidate branch nodes in that interval that could possibly be $M(i)$. The adversary will let the robot's last probe in that interval be $M(i)$. Therefore, $f(n) = \Omega(n)$. Furthermore, each of the remaining two subproblems contain a range of approximately at least $2n/3$ branch nodes (i.e., respectively including the first and the last third of this triple partition). Thus, $k \geq n/3$ and $n - k - 1 \geq n/3$. Therefore, from the above recurrence we get $T(n) \geq \Omega(n) + \min_{n/3 \leq k < 2n/3} \{T(k) + T(n-k-1)\}$. The solution of the latter recurrence is $T(n) = \Omega(n \log n)$. Since a map of such a graph can be verified with $O(n)$ edge traversals, the theorem follows. □

We can, of course, design more complicated algorithms. However, no matter what the algorithm is, we have the following general lower bound:

Theorem 3 *The competitive ratio for any kind of strategy for mapping general embedded graphs with a single marker is* $\Omega(\log n)$.

Proof: We apply the adversary argument similar to that of lower bound for sorting. The number of perfect matchings in K_{2n} (the complete graph with $2n$ nodes) is $F_n = \frac{(2n)!}{n!2^n}$. Thus, for star-shape graph of $2n+1$ nodes, there are F_n possibilities. Initially, there are F_n graphs matching the observation of the robot. Denote this set of possible graphs by S. If the robot never drops the marker, it can gain no information about which one of the star-shape graphs it is exploring. So we ignore those moves when the robot carries the marker and focus on those moves when the marker is dropped somewhere in the graph. For each graph in S, the robot knows where the marker is. In this star-shape graph, it has to move at least one edge to reach a node different from its current position. After each move, depending on which graph in S the robot is on, it will reach a node with or without the marker. Let S_0 be the set of graphs in S where the robot does not see the marker and S_1 be the set of graphs in S on which the robot sees the marker. The adversary then will let the robot see the marker at the node if $|S_1| > |S_0|$, and let the robot see no marker otherwise. This will eliminate no more than half the possible graphs from S. Thus we have to move at least $\Omega(\log F_n) = \Omega(n \log n)$ steps to reduce F_n possibilities to one. □

However, we conjecture Theorem 1 holds for all possible strategies and with the above star-shape graph as the example for the lower bound.

Conjecture 1 *The competitive ratio for any kind of strategy for mapping general embedded graphs with a single marker is* $\Theta(n)$.

Conjecture 2 *The competitive ratio for any kind of strategy for mapping embedded planar graphs with a single marker is* $\Omega(\log n)$.

However, the star-shape planar graphs cannot be used to establish Conjecture 2. The reason is, the following algorithm maps the star-shape planar graphs within a linear number of traversals, using a single marker. Note, this algorithm does not work for arbitrary embedded planar graphs, it is designed only for planar star-shape graphs. And the strategy we use is

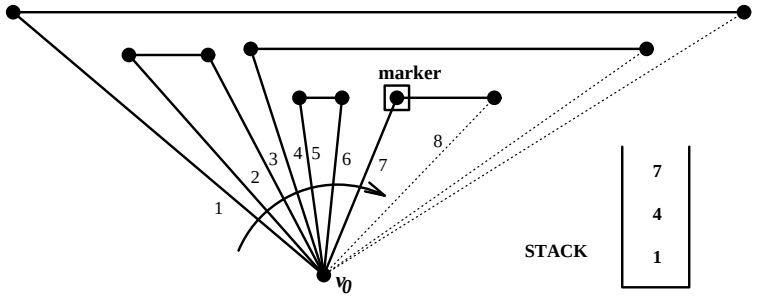


Figure 5: A snapshot of Algorithm 1.

a relaxed depth-one strategy. This gives an evidence that the relaxed depth-one strategy is more powerful than depth-one strategies.

Algorithm 1 *An algorithm with single marker for planar star-shape graphs.*

1. *Move to the center v_0 and consider incident edges in clockwise order: $1, 2, \cdots, 2n$.*
2. *Initialization: Make stack S empty, $PUSH(S, 1)$, move to v_1 and place the marker there and come back to v_0.*
3. **for** $i \leftarrow 2, \cdots, 2n$ **do**
   - *(a) $j \leftarrow TOP(S)$, move along the path $(v_0, v_i, M(i))$,*
   - *(b)* **if** *the marker is on $M(i)$*
   - *(c)* **then do** *record $M(i) \leftarrow v_j$, POP(S), $j \leftarrow TOP(S)$, pick up the marker, move along $(M(i), v_i, v_0, v_j)$ drop the marker at v_j and return to v_0.*
   - *(d)* **else do** *$PUSH(S, i)$, move along $(M(i), v_i, v_0, v_j)$, pick up the marker, move along (v_j, v_0, v_i), drop the marker at v_i and return to v_0.*

   end

Note: In the above algorithm assume $TOP(\text{empty}) = 0$.

A snapshot of this algorithm is shown in Figure 5 at the beginning of iteration $i = 8$. We clearly see that each iteration of the algorithm makes at most 8 edge traversals. Therefore, it makes a total of $O(n)$ edge traversals. Thus we have a constant competitive ratio for

mapping planar star-shape graphs with a single marker. For the correctness proof, it is essentially sufficient to notice the loop invariant of the algorithm. Suppose at the beginning of iteration i the contents of stack S from bottom to top are $j_1, j_2, \cdots, j_k$. Then (from the nesting property) the following hold:

1. $j_1 < j_2 < \cdots < j_k < i$,
2. $i \leq M(j_k) < M(j_{k-1}) < \cdots < M(j_2) < M(j_1)$,
3. the marker is at v_{j_k},
4. the robot is at v_0.

4 Exploration with Multiple Markers

As we have seen, a single marker does not lead to any competitive algorithm on general graphs. We now restrict our discussion on embedded planar graphs. In addition, we consider the use of multiple markers. We show that with $|V|$ *homogeneous* (or identical) markers, the map can be constructed in $O(|E|)$ traversals. In comparison, if we have $|V|$ *distinguished* markers, using one for each node, the general mapping problem can be solved in $2|E|$ edge traversals by a standard depth-first-search [DP]. The latter can be simulated using $\binom{|V|+1}{2}$ identical markers (where a bunch of i identical markers would simulate a distinguished marker labeled i).

As an intermediate step, we first propose a competitive algorithm in the footprint model. The footprints can of course be simulated using $|V| + |E|$ identical markers, one per node and edge. We then show how to maintain competitiveness while reducing the number of identical markers down to $|V|$.

4.1 A Mapping Algorithm for the Foot Print Model

Here, we introduce a competitive algorithm to explore unknown planar embedded graphs using foot-prints (no other markers). Next, we discuss how this algorithm can be modified to map unknown planar graphs of n nodes, using only n identical markers.

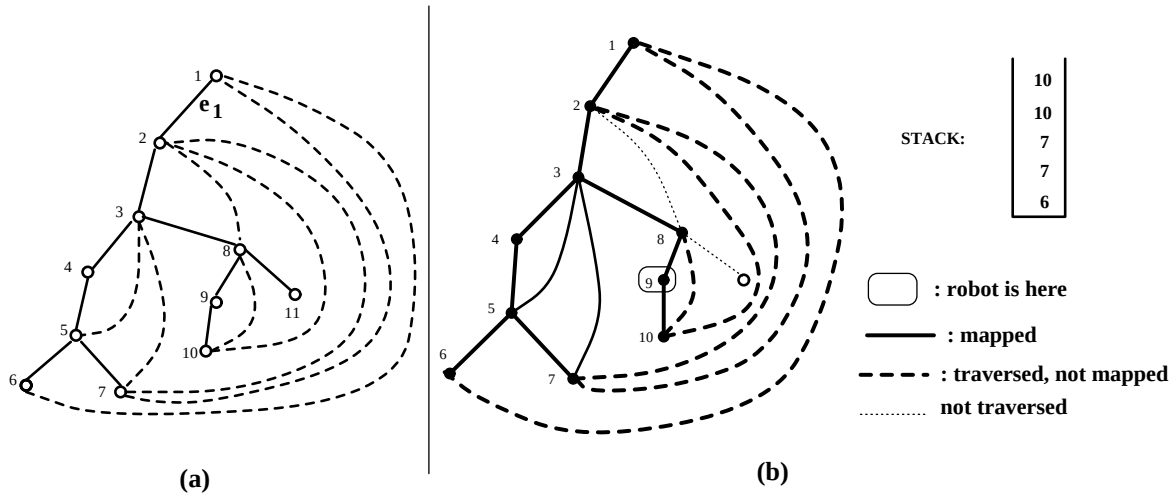


Figure 6: A rightmost-DFS traversal of a planar embedded graph.

The backbone of our algorithm for traversing the given embedded planar graph is a *rightmost depth-first-search* (or rightmost-DFS, or RDFS for short). Figure 6(a) shows an example. For the moment assume we know the given embedded planar graph. The first node we start the search becomes the root of the DFS-tree, and the first edge (e_1 in Figure 6(a)) leads to the leftmost child of the root. From then on, whenever we have to select the next edge out of the current node, in order to continue the DFS, we always select the rightmost one available, i.e., counter-clockwise first. Such a traversal of the graph gives us a DFS-tree which we call a *rightmost DFS-tree*. The non-tree edges are called back-edges. The crucial property of a rightmost DFS-tree is that all the back-edges appear on the right shoulder of the tree. This forces a generalized nesting (or parentheses) structure among the back-edges. We will exploit this property in our algorithm.

Now assume we are given an unknown planar embedded graph and the robot can use its footprints to traverse the graph. How can we use the footprints on nodes and edges to determine what kind of a node or edge we are traversing? If the robot moves from node v to node u and finds that there is no previous footprint on node u, then (v, u) becomes a tree-edge with $v = parent(u)$. In this case we can give a new name to u and add it to the partial map. However, if there is a previous footprint on u, then (v, u) is a back-edge. In this case u could

be either an ancestor or a descendent of v with respect to the rightmost DFS-tree. Node u is a descendent of v if there was a previous footprint on edge (v, u) (and is an ancestor otherwise). In case (v, u) is a back-edge, node u must be one of the nodes in the partial map, but which one? To determine that, we exploit the nesting structure of the back-edges by using a stack S. The first time a back-edge is traversed, its (known) starting end-point is pushed on the stack, and the second time a back-edge is traversed (when it already has a footprint), we pop from the stack to determine which node of the map matches with the other (known) end of the edge. For each edge traversed but not yet identified, one of its endpoints is put on $Stack$, we will determine the other endpoint later. Traversals of edges incident to a node i follow the counter-clockwise order (i.e., rightmost first), starting with the tree-edge connecting the node to its parent. Thus, our exploration algorithm will find the rightmost DFS tree.

Notation: The boolean function *footprint* indicates whether there is a previous footprint on the node/edge that appears as the function parameter. For a node v in the physical graph, $degree(v)$ denotes the number of edges incident to v. For a node v in the partial map, $RemDegree(v)$ is the number of edges incident to the physical node corresponding to v that are not yet traversed from v.

Algorithm 2 *Mapping a planar embedded graph using foot-prints.*
$v \leftarrow$ the starting node. $NodeCount \leftarrow 1$. In the map name v as $NodeCount$.
$RemDegree(NodeCount) \leftarrow degree(v)$. $Stack \leftarrow \emptyset$. Select an edge e out of node v.
repeat

1. **if** *not $footprint(e)$* **then do**

1.1. the robot moves from v along edge e to the other end, say node u.

1.2. **if** *$footprint(u)$*

1.2.1. **then** *backtrack from u to (its descendent) v along the back-edge e, and $PUSH(Stack, v)$, and decrement $RemDegree(v)$ by one.*

1.2.2. **else** *Increment $NodeCount$ by one. In the map name node u as $NodeCount$. Draw the tree-edge (v, u) in the map with u as the rightmost-child (so far) of v. Decrement $RemDegree(v)$ by one and set $RemDegree(u) \leftarrow degree(u) - 1$. Set $v \leftarrow u$.*

2. **else if** *$footprint(e)$ and $RemDegree(v) > 0$*

2.1. **then do** $u \leftarrow POP(Stack)$. *Add the back-edge* $e = (v, u)$ *from* v *to (its descendent)* u *to the map by drawing it to the relative right of everything drawn on the map so far. Decrement* $RemDegree(v)$ *by one.*

2.2. **else if** $RemDegree(v) = 0$ **then** *Backtrack from* v *to its parent along the tree-edge* e. *Set* v *to this parent node.*

3. Now the robot is at node v *and has incident edge* e *as reference. Let* $enext$ *be the counter-clockwise next edge incident to* v *relative to edge* e.
Set $e \leftarrow enext$.

until v *is DFS-root and* $RemDegree(v) = 0$
end

A snapshot of this algorithm is shown in Figure 6(b). Figure 8 gives a complete execution of the algorithm on a small example. (In the global and local views, solid edges have footprints, dashed edges do not. In the map, dangling edges incident to a node are those whose other end is not yet identified. Dashed circles represent robot positions.) The above algorithm traverses each edge no more than twice. We only need to prove that names of two nodes on a back-edge are correctly matched.

Theorem 4 *The above robot mapping algorithm maps an unknown embedded planar graph in the foot-print model by traversing each edge at most twice.*

Proof: We do an induction proof on the number of edges in the unknown graph. Let $G = (V, E)$ be the embedded planar graph to map. If $m = n - 1$, the algorithm correctly constructs the map of tree G without performing any push or pop operations on the Stack. Now assume $m > n - 1$. By the induction hypothesis assume that the claim is true for all graphs with less than m edges.

Now consider an unknown graph with m edges. We denote the depth first search tree by T and denote by $C_T(e)$ the only cycle in $T \cup \{e\}$ for each non-tree edge e. Consider the first time we do a $POP(Stack)$ operation. According to our algorithm, we do $u \leftarrow POP(Stack)$ and set $e = (v, u)$ when, at node v, we see a footprint on the non-tree edge e incident to v. Therefore, e is a back-edge from a node in the subtree rooted at v. We need to prove that $e = (v, u)$. Since this is the first time $POP(Stack)$ operation is done and each $POP(Stack)$

operation corresponds to a back edge, there are no other back-edges incident to the interior nodes from v to u along the rightmost depth-first search route $T(v,u)$ in the RDFS-tree T. Note that one tree edge may appear twice on $T(v,u)$. On the other hand, there is some back-edge f incident to u since u is on the Stack. Thus, the edge f and the RDFS-tree form a loop. Since we do the first $POP(Stack)$ operation at node v, the other end node of f cannot be on the interior of the route $T(v,u)$. If it is an ancestor of v on the tree, e should be incident to a node x on the route $T(v,u)$ and the cycle $C_T(e)$ would be incident in the cycle $C_T(f)$ by planarity. According to rightmost depth first search, the edge e will be searched after f is searched. Then x will be pushed into the stack after u. a contradiction. We must have $f = e = (v,u)$. Also it is not difficult to find out the position of f with respect to the DFS-tree edge from u to its parent (it is counter-clockwise the last traversed but unmapped edge incident to u). The similar property is true for the position of e with respect to node v.

To conclude the proof, consider the embedded planar graph $G - \{(v,u)\}$. Our algorithm will proceed in the same way as above except those involving the edge (v,u). By induction, our algorithm will correctly map $G - \{(v,u)\}$. Combined with the above discussion, our claim holds for the graph G. Therefore, the theorem follows by the induction hypothesis. □

4.2 Exploration with n Identical Markers

We extend the algorithm developed above to explore unknown planar embedded graphs of n nodes with n identical markers. However, the above algorithm does not trivially carry over: simply replacing foot-prints by markers would need $m + n$ markers. In fact, we achieve this with fewer markers by increasing the number of edge traversals.

Theorem 5 *There is an algorithm for a robot to map an unknown embedded planar graph of n nodes with n identical markers by traversing each edge at most four times.*

Proof: Whenever we reach a node, we put down one marker never to be removed. Thus, this is equivalent to have a foot-print at each node. However, we are not able to put markers on edges to simulate foot-prints on the edges. Instead, we use other operations to compensate functions of foot-prints on edges.

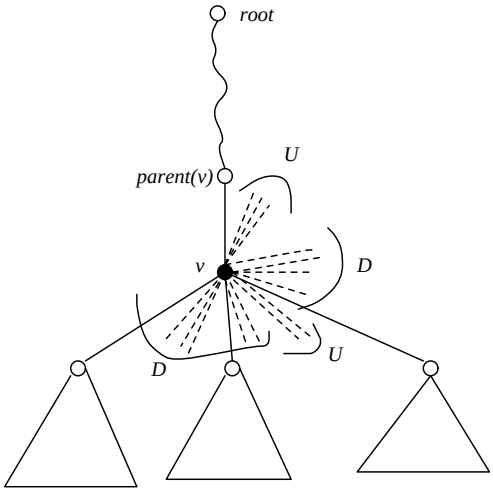


Figure 7: The scanning operation.

Footprints on edges are used in Algorithm 2 to distinguish whether a back-edge is being traversed for the first time (up the tree) and a Stack push operation must be done, or whether it is being traversed for the second time (down the tree) and a Stack pop operation must be done to match up the two ends of the edge. We use two new methods to achieve this without using foot-prints on edges. First, each time a node v is first reached (i.e., when we just put a marker on the node) we perform a scanning operation, denoted $scan(v)$. This procedure probes every edge incident to v to find out whether other end-nodes, at that instant, have markers or not. If there is a marker on the other end of an edge, we call it an *up-edge*. Otherwise, it is called a *down-edge*. We partially draw such (dangling) edges in the map and label U or D accordingly. (See Figure 7.) Procedure $scan(v)$ will cause the robot to traverse the edges incident to v twice (back and forth). The rest of the DFS traverses these edges twice more, for a total of four times. An up-edge is a back-edge leading to a node already named, with a name smaller than v. A down-edge is a back-edge or a tree-edge leading to a node to be named later, with a name bigger than v. The other modification needed in the algorithm is to replace the test for $footprint(e)$ in the algorithm by checking the label of the corresponding (dangling) edge in the map to determine whether it is an edge going up or down. □

5 Remarks and Discussion

The lower-bound result of this paper for the problem of mapping with a single marker show that the algorithm proposed by Dudek et al. is an asymptotically optimal solution in terms of competitive ratios. The constant competitive ratio mapping algorithm with n homogeneous markers for planar graphs is a significant improvement over the competitive ratio two results for mapping undirected graphs with distinguishable nodes (which demands $\binom{n+1}{2}$ markers. Several questions follow from this work on competitive mappings of unknown graph environments.

1. We have obtained a competitive algorithm for mapping embedded planar graphs with n identical markers. However, in our algorithm, the tokens are put down never to be picked up. How many tokens do we need to competitively map embedded planar graphs if we reuse the tokens? In particular, can we design a competitive algorithm with a sublinear number of markers?

2. What is the exact competitive ratio for mapping general or planar graphs with a single marker?

3. Can we identify other classes of practical graph maps for which competitive mapping is possible?

4. While the metric map takes too much detail into account, the graph map probably eliminates too much information. A more interesting/practical situation is when we allow the robot to have relatively accurate metric local views but relatively loose topological global views of the environment. What kind of mapping strategies can we construct in such situations?

5. Most of the work in competitive analysis takes distance as the performance measure. However, speeds may not be the same for two paths of the same lengths. The speed on a straight path would usually be faster than that of a zig-zagged path. How do we deal with such situations?

6 Acknowledgement

Authors' research was partly supported by NSERC grants.

References

[BBFY] E. Bar-Eli, P. Berman, A. Fiat, and P. Yan, "Online Navigation in a Room," SODA 1992, pp. 237-249.

[BCR] R.A. Baeza-Yates, J.C. Culberson, and G.J.E. Rawlins, "Searching in the Plane," To appear in *Information and Computation.*

[BL] K.S.Booth, G.S.Lueker [1976] "Testing the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms," JCSS 13 (1976), 335-379.

[Bol] B. Bollobas, "Extremal Graph Theory," Academic Press, San Francisco, 1978.

[BRS] A. Blum, P. Raghavan and B. Schieber, "Navigating in Unfamiliar Geometric Terrain," STOC 1991, pp.494-504.

[Ca] J. Canny, "A new algebraic method for robot motion planning and real geometry," FOCS 1987, pp. 39-48.

[CR] J. Canny, J.Reif, "New lower bound techniques for robot motion planning problems," FOCS 1987, pp. 49-60.

[CDRX] J. Canny, B.Donald, J.Reif, P.Xavier, "On complexity of kinedynamic planning," FOCS 1988, pp. 306-316.

[CNA] N.Chiba, T.Nishizeki, S.Abe, "A linear algorithm for embedding planar graphs using PQ-trees" JCSS 30 (1985), 54-76.

[DJMW] G.Dudek, M, Jenkin, E. Milios, D. Wilkes, "Using Multiple markers in graph exploration," *IEEE Trans. on Robotics and Automation*, Vol.7, 1991,pp.859-865.

[DKP] X. Deng, T. Kameda and C. H. Papadimitriou, "How to Learn an Unknown Environment," FOCS 1991, pp.298-303.

[DP] X. Deng, and C. H. Papadimitriou, "Exploring an Unknown Graph," FOCS 1990, pp.355-361.

[FFKRRV] A. Fiat, D.P. Foster, H. Karloff, Y. Rabani, Y. Ravid, and S. Vishwanathan, "Competitive Algorithms for Layered Graph Traversal," FOCS 1991, pp.288-297.

[GMR] L.J. Guibas and R. Motwani and P. Raghavan, " The Robot Localization Problem in Two Dimensions," SODA 1992, pp. 259-268.

[HR] N. Hartsfield, G. Ringel, "Pearls in Graph Theory", Academic Press, 1990.

[HT] J.E.Hopcroft, R.E.Tarjan [1973] "Efficient planarity testing", JACM 21(4), 1973, 549-568.

[KB] B. Kuipers, and Y. Byun, "A Robot Exploration and Mapping Strategy Based on a Semantic Hierarchy of Spatial Representatinos," *Robotics and Autonomous Systems* **8** (1991) 47-63.

[Kl] R. Klein, "Walking an Unknown Street with Bounded Detour," FOCS 1991, pp. 304-313.

[KL] B. Kuipers, and T. Levitt, "Navigation and mapping in large-scale space," *AI Mag.*, Summer 1988, pp.61-74.

[KP] B. Kalyanasundaram, and K. Pruhs, "A Competitive Analysis of Algorithms for Search Unknown Scenes," manuscripts, 1991.

[KRR] H. Karloff, Y. Rabani, and Y. Ravid, "Lower Bounds for Randomized k-Server and Motion-Planning Algorithms," *STOC 1991*, pp. 278-288.

[LL] T.S. Levitt, and D. T. Lawton, "Qualitative Navigation for Mobile Robots," *Artificial Intelligence* **44** (1990), 305-360.

[PY] C. Papadimitriou and M. Yannakakis, "Shortest paths without a map," *Theoretical Comp. Science 84*, 1991, pp.127-150.

[RS] R. L. Rivest and R. E. Schapire, "Inference of Finite Automata Using Homing Sequences," STOC 1989, pp.411-420.

[ST] Sleator, D.D. and Tarjan, R.E., Amortized efficiency of list update and paging rules," *Communications of the ACM 28(2)*, 1985, pp.202-208.

[Val] L. G. Valiant, "A theory of the learnable," STOC 1984, pp. 436-445. (Also CACM 1984, pp. 1134-1142.)

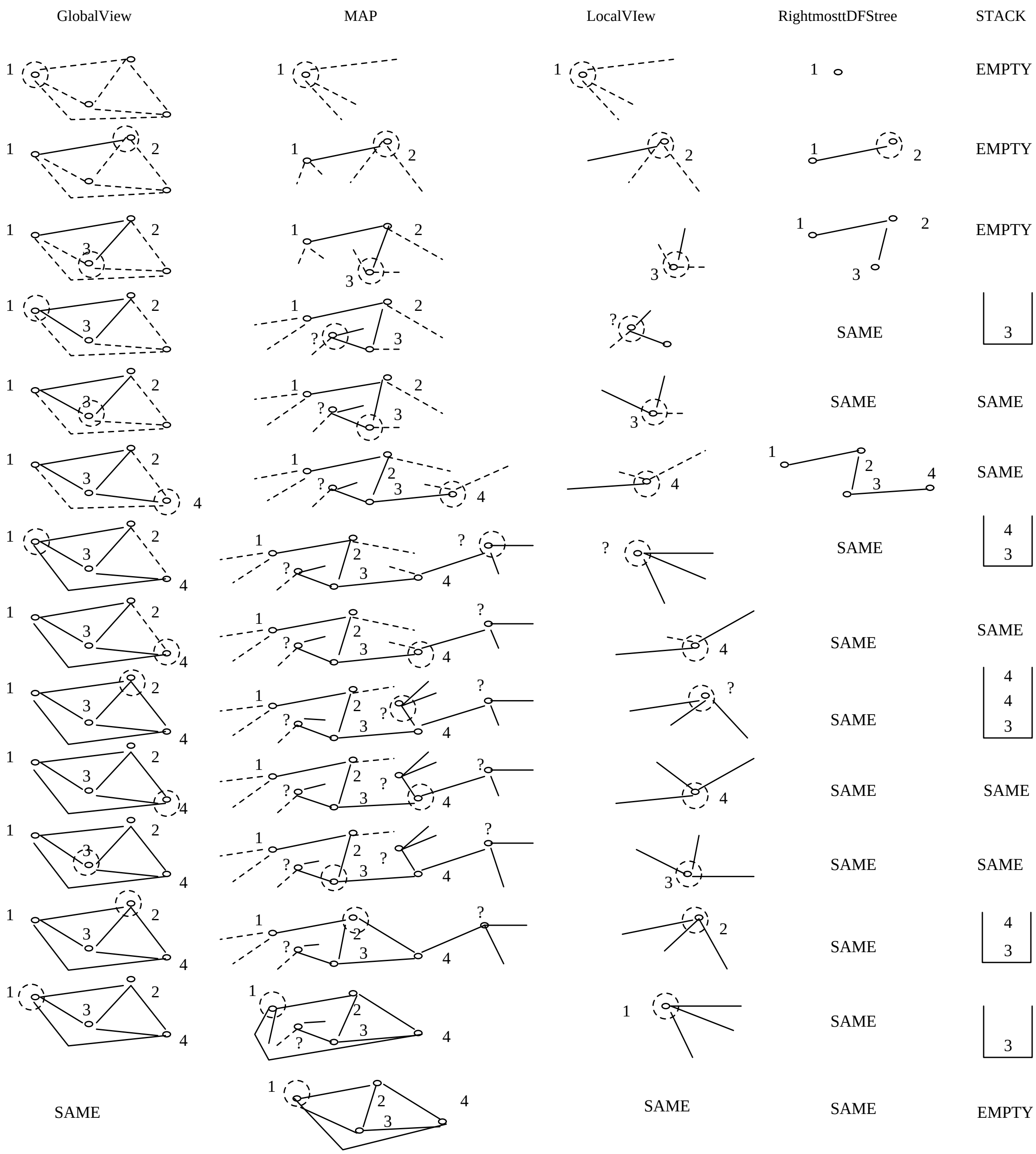


Figure 8: A complete example of mapping by Algorithm 2.