The Online Graph Exploration Problem on Restricted Graphs

Shuichi MIYAZAKI\(^{(a)}\), Member, Naoyuki MORIMOTO\(^{(b)}\), Nonmember, 
and Yasuo OKABE\(^{(c)}\), Senior Member

SUMMARY The purpose of the online graph exploration problem is to visit all the nodes of a given graph and come back to the starting node with the minimum total traverse cost. However, unlike the classical Traveling Salesperson Problem, information of the graph is given online. When an online algorithm (called a searcher) visits a node \(v\), then it learns information on nodes and edges adjacent to \(v\). The searcher must decide which node to visit next depending on partial and incomplete information of the graph that it has gained in its searching process. The goodness of the algorithm is evaluated by the competitive analysis. If input graphs to be explored are restricted to trees, the depth-first search always returns an optimal tour. However, if graphs have cycles, the problem is non-trivial. In this paper we consider two simple cases. First, we treat the problem on simple cycles. Recently, Asahiro et al. proved that there is a 1.5-competitive online algorithm, while no online algorithm can be \((2-\epsilon)\)-competitive for any positive constant \(\epsilon\). In this paper, we give an optimal online algorithm for this problem; namely, we give a \(1+\sqrt{3}\) (= 1.366)-competitive algorithm, and prove that there is no \((1+\sqrt{3})-\epsilon\)-competitive algorithm for any positive constant \(\epsilon\). Furthermore, we consider the problem on unweighted graphs. We also give an optimal result; namely we give a 2-competitive algorithm and prove that there is no \((2-\epsilon)\)-competitive online algorithm for any positive constant \(\epsilon\).

key words: the graph exploration problem, online algorithm, competitive analysis

1. Introduction

In the Traveling Salesperson Problem (TSP)\([14]\), we are given a graph and non-negative weights (lengths) on edges. Our task is to find a tour visiting all the nodes and coming back to the starting node with minimum cost. The cost of a tour is the total length of the tour. This problem is a well-known NP-hard problem, and there have been intensive studies such as heuristics and approximation algorithms. Apparently, TSP has plenty of practical applications, which includes determining a pickup or delivery tour for delivery companies or minimizing the total movement cost of robot arms in LSI wiring.

In TSP, all information on the graph is given to the algorithm in advance. However, in some cases of real applications, the terrain may be unknown until the algorithm visits the place, and the algorithm learns the local environment when it actually visits there. For example, suppose that we wish to gather complete information of an unknown environment using a robot searcher. At the beginning, the robot has no knowledge of the environment. It should decide where to visit next depending on the partial information of the environment that it has gained through the exploration so far. This kind of problem is known as the exploration or the map construction problem and there are several models. In \([13]\), the problem is formulated in an online problem on undirected edge-weighted graphs as follows: At the beginning, the searcher is at the starting node \(o\), called the origin, and it knows the local information, namely, the labels of the nodes adjacent to \(o\), and the weights of edges incident to \(o\). When the searcher visits a node \(v\), then it learns the labels of nodes adjacent to \(v\) and the weights of edges incident to \(v\). When the searcher moves from \(u\) to \(v\) along the edge \((u, v)\), it costs the weight of \((u, v)\). The task of the searcher is to visit all the nodes and return to the origin with as small cost as possible. The goodness of the algorithm is evaluated by the competitive analysis\([5], [10]\).

The most natural algorithm one may consider is the greedy type Nearest Neighbor algorithm (NN), which always visits a node nearest to the current node, among those that have not yet been visited. However, it has been shown that NN is not competitive even for planar graphs; there exists a planar graph \(G\) with \(n\) nodes such that the competitive ratio of NN is \(\Omega(\log n)\)\([15]\). Kalyanasundaram and Pruhs\([13]\) proposed a modified version of NN, called Short-Cut, and proved that it is 16-competitive for planar graphs.

Note that if input graphs to be explored are restricted to trees, the depth-first search always returns an optimal tour because to visit all nodes and come back to the origin, each edge must be traversed at least once, and the depth-first search traverses each edge exactly twice. Hence, the simplest non-trivial case is probably cycles. Recently, Asahiro et al.\([2]\) considered the graph exploration on cycles. They proved that NN achieves the competitive ratio of 1.5 and showed that no online algorithm can have the competitive ratio better than 1.25.

Our Results.

In this paper we consider the problems on two classes of graphs and give tight bounds on the competitive ratio for both cases. First we improve both upper and lower bounds of the problem on cycles, and give a tight bound \(1+\sqrt{3}/2\) (= 1.366). For improving the upper bound, we propose a new
algorithm called DIST, which decides the next node to visit depending on the (weighted) distances between the current node and each of the unvisited two nodes, the total length of the exploration so far, and the distance from the origin to the current node. We also consider the problem on unweighted graphs, and give a tight bound of 2; namely, we prove that algorithm DFS is 2-competitive and that no online algorithm can have the competitive ratio better than 2 (this lower bound holds even when graphs have planarity).

Related Results.

There are several variants of the problem of exploring unknown environment online. Deng and Papadimitriou [6] considered the problem of exploring a directed unweighted graph. This problem requires us to explore not only all nodes but also all edges, and the cost of the searcher is measured by the total number of edges traversed. They gave an online algorithm with \( d^{\text{Out}} m \) edge traversals, where \( m \) is the number of edges in the graph and \( d \) is the minimum number of edges that have to be added to make the graph Eulerian. Albers and Henzinger [1] presented an algorithm that achieves an upper bound of \( d^{\text{Out}}(\log d)m \), and Fleischer and Trippen [9] gave an algorithm with an upper bound of \( O(d^8 m) \). Fleischer and Trippen [8] also made an experimental study of major online graph traversal algorithms and evaluated their practical performance on various graph families. In the polygon exploration problem (e.g. [7], [12]), an unknown environment is modeled by a polygon. The task of a searcher is to see all the boundaries of the polygon and come back to the starting point. Ausiello et al. [3] and Ausiello et al. [4] have studied the online traveling salesperson problem in which requests are presented online, and the aim of the searcher is to visit each requested point (not necessarily in the order of requests, unlike the \( k \)-server problem).

2. Preliminaries

The purpose of the Online Graph Exploration problem is to visit all the nodes of a given graph \( G = (V, E) \), where \( V \) and \( E \) denote the sets of nodes and edges, respectively. For each edge \((u, v) \in E\), a non-negative weight \( \ell(u, v) \), sometimes called the length, is associated. Initially, the searcher is at the specified node \( o \in V \), called the origin. It knows only the labels of the nodes adjacent to \( o \), and the length of edges connecting \( o \) with those neighborhood nodes. Once the searcher visits a node \( v \), it learns the labels of nodes adjacent to \( v \) and the length of edges incident to \( v \). The searcher has a sufficiently large memory so that it can store all information obtained so far, namely, the labels of nodes, the weights of edges, and the topology of the subgraph consisting of nodes and edges it has already learned. The task of the searcher is to determine the next node to visit, using only the current knowledge. The goal of the searcher is to visit all the nodes and return to the origin. The cost of the searcher for the graph \( G \) is the total length of the tour made by the searcher on \( G \).

The performance of an online algorithms is evaluated by the competitive analysis: Let \( \text{ALG}(G) \) denote the cost of an algorithm \( \text{ALG} \) on \( G \), and let \( \text{OPT}(G) \) denote the cost of an optimal offline algorithm \( \text{OPT} \) for \( G \). We say that \( \text{ALG} \) is \( c \)-competitive for a class of graphs \( G \) if \( \text{ALG}(G)/\text{OPT}(G) \leq c \) for any graph \( G \in G \). We may write \( \text{ALG} \) and \( \text{OPT} \) instead of \( \text{ALG}(G) \) and \( \text{OPT}(G) \), respectively, when \( G \) is clear.

3. A Tight Bound on Cycles

In this section we consider the problem on cycles and give a tight bound for the competitive ratio. Here is one simple but important fact [2]. Let \( \ell_{\max} \) be the maximum length of edges and \( L = \Sigma_{(u,v)\in E} \ell(u,v) \) be the sum of the length of all edges.

Fact 1: For any cycle \( C \), \( \text{OPT}(C) = L \) if \( \ell_{\max} \leq \frac{L}{2} \), and \( \text{OPT}(C) = 2(L - \ell_{\max}) \) if \( \ell_{\max} > \frac{L}{2} \).

3.1 A Lower Bound

In this section, we give a lower bound on the competitive ratio for any online algorithm.

Theorem 1: For any positive constant \( \epsilon \), there is no \((\frac{1+\sqrt{3}}{2} - \epsilon)\)-competitive online algorithm for cycles.

Proof. We will introduce an adversary giving the above mentioned lower bound. Fix an integer \( n \) and a constant \( \mu \) such that \( n > \frac{2\mu}{\epsilon^2} \) and \( \mu < 1 \). First, the adversary reveals two edges \((o,u_1)\) and \((o,v)\) incident to the origin with the equal length one. Without loss of generality, assume that the searcher moves to \( u_1 \). Then, the adversary reveals an edge \((u_1,u_2)\) such that \( \ell(u_1,u_2) = 1 \). If the searcher visits \( u_2 \), then a new edge \((u_2,u_3)\) with \( \ell(u_2,u_3) = 1 \) is revealed. Similarly, as long as the searcher visits a new node \( u_i (i \leq n - 1) \), the adversary gives an edge \((u_i, u_{i+1})\) with \( \ell(u_i,u_{i+1}) = 1 \).

Suppose that the searcher visits the node \( v \) before visiting \( u_{n-1} \), and suppose that this happens just after it visited \( u_i \) where \( t < n - 1 \) (i.e. it went back to \( v \) when it saw the edge \((u_i,u_{i+1})\) (Fig. 1 (a)). Then the edge \((v,u_{n+1})\) with weight \( t \) is revealed (Fig. 1 (b)). The only unvisited node is \( u_{n+1} \), and the best way for the searcher is to go to \( u_{n+1} \) directly from
\[ v, \text{ and go back to the origin by either clockwise or counter-clockwise direction. The cost of the searcher is then } 4t + 2. \]

The optimal tour is to visit all nodes along the cycle, whose cost is \( 2t + 2 \). The competitive ratio in this case is then \( (4t + 2)/(2t + 2) \geq 1.5 \) since \( t \geq 1 \).

Next, suppose that the searcher visits \( u_n \) before visiting \( v \). Then the adversary gives an edge \((u_n, w)\) with length \( \sqrt{3}n \) (Fig. 2 (a)). We have two cases. First, suppose that the searcher visits \( w \). Then the adversary reveals the edge \((w, v)\) such that \( \ell(w, v) = \mu \) (Fig. 2 (b)). The best way for the searcher is to visit \( v \) and \( o \) in this order (note that \( \mu < 1 \)). The cost of the searcher is then \( n + \sqrt{3}n + \mu + 1 = (1 + \sqrt{3})n + \mu + 1 \).

Note that the edge \((u_n, w)\) has the length more than half the total length of the whole cycle. So, by Fact 1, the optimal cost is \( 2(n + \mu + 1) \). The competitive ratio is \( (1 + \sqrt{3})/(2(n + \mu + 1)) > \frac{\sqrt{3} - \sqrt{3}(\mu + 1)}{2} \approx \frac{1 + \sqrt{3}}{2^{2/5}} - \epsilon \).

Finally, suppose that after the edge \((u_n, w)\) is revealed, the searcher goes back to the node \( v \). In this case, the adversary reveals the edge \((v, w)\) with \( \ell(v, w) = (\sqrt{3} + 1)n - 1 \) (Fig. 2 (c)). Then the only unvisited node is \( w \), and the best way for the searcher is now to visit \( w \) directly from \( v \), and then go back to the origin in either clockwise or counter-clockwise direction. The total cost of the tour is \( n + n + 1 + (\sqrt{3} + 1)n - 1 + (\sqrt{3} + 1)n = (2 + \sqrt{3} + 4)n \). The optimal tour is a one along the cycle, whose cost is \( (2 + \sqrt{3})n \) along the cycle, whose cost is \( (2 + \sqrt{3})n \). The competitive ratio in this case is \( 1 + \frac{\sqrt{3}}{2} \).

3.2 An Upper Bound

In this section, we give an online algorithm \( \text{DIST} \) and analyze its competitive ratio.

3.2.1 Algorithm \( \text{DIST} \)

Since a given graph is a cycle, there are always two choices for the searcher: (except for the 1st step), either to go forward or to go back. (See Fig. 3 (a). The visited nodes are surrounded by a dotted curve, and the current position of the searcher is indicated by the black node.) Before presenting the algorithm, we give a few notations. Suppose that as shown in Fig. 3 (a), the searcher is currently at the node \( u \), and is to determine which of \( x \) and \( y \) to visit. For any two nodes \( v_1 \) and \( v_2 \), let \( d(v_1, v_2) \) denote the distance between \( v_1 \) and \( v_2 \) along the edges already known. Let \( X \) be the total length the searcher has traversed so far, and define \( W = X - d(o, u) \). The value of \( W \) may change as time goes, so it might be appropriate to express it as e.g. \( W_i \) for Step \( i \). However, for conciseness, we use \( W \) when there is no fear of confusion, or we sometimes say as “\( W\)-value at this moment”. Now, we are ready to give our algorithm \( \text{DIST} \):

**Step 1:** The searcher is at the origin \( o \), and there are two nodes adjacent to \( o \). It moves to the node closer to \( o \). If both are in the same distance, it chooses arbitrary one.

**Step \( i \) (\( i \geq 2 \)):** Suppose that the searcher is at a node \( u \) as shown in Fig. 3 (a). If \( \ell(u, x) \leq \sqrt{3}d(u, y) - W \), then the searcher moves to \( x \). Otherwise, i.e., if \( \ell(u, x) > \sqrt{3}d(u, y) - W \), then the searcher moves to \( y \).

**Final step:** The current situation is like Fig. 3 (b). When the searcher visits a node \( u \), it learns that \( u \) is connected to the unvisited but known node \( y \) (since it has seen \( y \) when it was on the node of the other side). Now, it
knows the entire graph, and there is only one unvisited
time before the final step,
the current
has traversed so far minus

3.2.2 Competitive Analysis

In this subsection, we prove the following theorem:

Theorem 2: $\text{DIST}$ is $1 + \sqrt{3}$-competitive for cycles.

Proof. Consider any time step of the online game, and suppose
that the situation is like Fig. 3 (a). (In the case just before
the final step, $x$ is equal to $y$ as Fig. 3 (b).) Let $W$ be
the current $W$-value, namely, the total distance the searcher
has traversed so far minus $d(o,u)$. The following lemma is
crucial in our analysis:

Lemma 3: For anytime before the final step, $W \leq (\sqrt{3} - 1)d(o,y)$.

Note: In the case of Fig. 3 (b), $d(o,u)$ is the distance from $o$
to $u$ in a clockwise direction, and $d(o,y)$ is the distance from
$o$ to $y$ in a counter-clockwise direction.

Proof. The proof is by induction. After Step 1 is
performed, the situation is like Fig. 4 (a). Since $W = \ell(o,u) - \ell(u,o) = 0$, the inequality holds clearly.

Next, we assume that the inequality holds after Step $i$, and show that it holds after Step $i + 1$. Suppose that after the execution of Step $i$, the situation is like Fig. 3 (a). We denote
the $W$-value at this moment by $W_i$. By the induction hypothesis, the following inequality holds: $W_i \leq (\sqrt{3} - 1)d(o,y)$.

There are two cases to consider depending on whether the
searcher moves to $x$ or $y$ in the next step.

Case 1. The searcher moves to $x$ at Step $i + 1$. Then
the situation is like Fig. 4 (b). Note that by definition, the
current $W$-value, denoted by $W_{i+1}$, is $W_{i+1} = W_i + \ell(u,x) - \ell(u,y) = W_i$. (This means that if the searcher goes forward,
then the $W$-value remains unchanged. This property will be
sometimes used hereafter.) So $W_{i+1} \leq (\sqrt{3} - 1)d(o,y)$
by the induction hypothesis, which implies that the inequality
holds after Step $i + 1$.

Case 2. The searcher moves to $y$ at Step $i + 1$. Then
the situation is like Fig. 4 (c). The total length of the tour
by the searcher increases by $d(y,o) + d(o,u)$ at this step.
The distance from the origin was $d(o,u)$ but is now $d(y,o)$.
Hence, $W_{i+1} = W_i + d(y,o) + d(o,u) - d(y,o) + d(o,u) = W_i + 2d(o,u)$. Since the searcher selected $y$ rather than $x$, $\ell(u,x) > \sqrt{3}d(u,y) - W_i$ holds. Also, by the induction hypothesis,
$W_i \leq (\sqrt{3} - 1)d(o,y)$. From these two inequalities and
the equality $d(u,y) = d(u,o) + d(o,y)$, we have $d(o,y) < (\ell(u,x) - \sqrt{3}d(u,o))$. Now using the hypothesis again, we have

$$W_{i+1} = W_i + 2d(o,u)$$
$$\leq (\sqrt{3} - 1)d(o,y) + 2d(o,u)$$
$$< (\sqrt{3} - 1)\ell(u,x) - \sqrt{3}d(u,o) + 2d(o,u)$$
$$= (\sqrt{3} - 1)\ell(u,x) + d(o,u))$$
$$= (\sqrt{3} - 1)d(o,x)$$

as required.

Now, suppose that we are at the moment just before the
final step. Then, the current situation looks like Fig. 5. The
searcher is at the node $u$, and it just learned that the node
adjacent to $u$ is $y$ is the same as the one it saw from $v$ before.
Since the searcher learned $\ell(u,y)$, it came to know the whole
information on the cycle.

For simplicity, let $a$, $b$, $c$, and $d$ denote the lengths
of paths (edges) $d(o,u)$, $\ell(u,y)$, $d(o,v)$, and $\ell(v,y)$, respectively,
as depicted in Fig. 5. Let $W^*$ be the $W$-value at this
moment. Then, by Lemma 3, $W^* \leq (\sqrt{3} - 1)d(o,y) = (\sqrt{3} - 1)(c + d)$ holds. The only unvisited node is $y$, and the
searcher will visit $y$ in either clockwise or counter-clockwise
direction depending on which route is shorter, and then go
back to $o$ from $y$ in either clockwise or counter-clockwise
direction, again depending on which route is shorter. We
will do a case analysis.

Case 1. $b > a + c + d$. In this case, $a + b > c + d$ holds.
So, the searcher visits $y$ by way of $o$, in a counter-clockwise
direction, and goes back to $o$ by way of $v$, in a clockwise
direction. Since the cost of the searcher before the final step
is $W^* + d(o,u) = W^* + a$ by the definition of the $W$-value,
the final cost is $\text{DIST} = W^* + a + (a + c + d) + (c + d) = W^* + 2(a + c + d)$. Because $b > a + c + d$, $\ell_{\text{max}} = b > L/2$.
So, the optimal cost is $\text{OPT} = 2(a + c + d)$ by Fact 1. The
competitive ratio is
\[
\frac{\text{DIST}}{\text{OPT}} = \frac{W^* + 2(a + c + d)}{2(a + c + d)}
\]
\[
\leq 1 + \frac{(\sqrt{3} - 1)(c + d)}{2(a + c + d)}
\]
\[
\leq 1 + \frac{\sqrt{3} - 1}{2}
\]
\[
= 1 + \frac{\sqrt{3}}{2}.
\]

**Case 2.** \(b \leq a + c + d\) and \(a + b \leq c + d\). The searcher visits \(y\) using the edge \((u, y)\), and goes back to \(o\) in a counter-clockwise direction, i.e., by way of \(u\). So, DIST = \(W^* + a + b + (b + a) = W^* + 2(a + b)\). Let \(e_{\text{max}}\) be an edge with maximum length, namely, \(\ell(e_{\text{max}}) = \ell_{\text{max}}\). We consider subcases according to the length and position of \(e_{\text{max}}\).

**Case 2-(i).** \(\ell_{\text{max}} \leq L/2\). By Fact 1, OPT = \(a + b + c + d\).

Thus,
\[
\frac{\text{DIST}}{\text{OPT}} = \frac{W^* + 2(a + b)}{a + b + c + d}
\]
\[
\leq \frac{(\sqrt{3} - 1)(a + b) + 2(a + b)}{a + b + c + d}
\]
\[
= \sqrt{3} - 1 + \frac{(3 - \sqrt{3})(a + b)}{a + b + c + d}
\]
\[
\leq \sqrt{3} - 1 + \frac{(3 - \sqrt{3})(a + b)}{2(a + b)}
\]
\[
= 1 + \frac{\sqrt{3}}{2}.
\]

**Case 2-(ii).** \(\ell_{\text{max}} > L/2\) and \(e_{\text{max}} = (u, y)\). This does not happen because \(b \leq a + c + d\).

**Case 2-(iii).** \(\ell_{\text{max}} > L/2\) and \(e_{\text{max}} = (v, y)\). By Fact 1, OPT = \(2(a + b + c)\). Consider the time when the searcher was at \(v\) (Fig. 6(a)), and let \(W^*\) be the \(W\)-value at this moment. Let \(w\) be an unvisited node other than \(y\) (this notation is used sometimes hereafter). Then, by Lemma 3, \(W^* \leq (\sqrt{3} - 1)d(a, w) \leq (\sqrt{3} - 1)a\). Note that at the next step, the searcher moved to \(w\) because \(y\) is the last node visited by the searcher. Let \(W''\) be the \(W\)-value just after the searcher moved to \(w\). Then, the total length of the tour increased by \(c + d(o, w)\), and the distance between the origin and the searcher changed from \(c\) to \(d(o, w)\). Hence, by a simple calculation, \(W'' = W^* + c + d(o, w) + c - d(o, w) = W^* + 2c\). Note that the searcher does not change the direction hereafter until it reaches \(u\). So, the \(W\)-value remains unchanged until the searcher reaches \(u\), namely, \(W^* = W'' = W^* + 2c \leq (\sqrt{3} - 1)a + 2c\) (recall that \(W^*\) is the \(W\)-value when the searcher is at \(u\)). Now,
\[
\frac{\text{DIST}}{\text{OPT}} = \frac{W^* + 2(a + b)}{2(a + b + c)}
\]
\[
\leq \frac{(\sqrt{3} - 1)a + 2c + 2(a + b)}{2(a + b + c)}
\]
\[
= 1 + \frac{(\sqrt{3} - 1)a}{2(a + b + c)}
\]

\[
\leq 1 + \frac{\sqrt{3} - 1}{2}
\]
\[
= 1 + \frac{\sqrt{3}}{2}.
\]

**Case 2-(iv).** \(\ell_{\text{max}} > L/2\) and \(e_{\text{max}}\) is in the path from \(o\) to \(v\) (in a clockwise-direction). We can show that this case does not happen in the following way: Suppose, on the contrary, that this happens. Consider the time when the searcher was at \(v\) (Fig. 6(b)). Since the searcher decided to move to \(w\) rather than \(y\), it must be the case that \(\ell(v, y) > \sqrt{3}d(v, w) - W^*\) where \(W^*\) is the \(W\)-value at this time. Also, by Lemma 3, \(W^* \leq (\sqrt{3} - 1)d(a, w)\). So, \(\ell(v, y) > \sqrt{3}d(v, w) - (\sqrt{3} - 1)d(a, w) = \sqrt{3}d(v, o) + d(o, w) \geq \ell_{\text{max}}\) because \(d(v, o) \geq \ell_{\text{max}}\) by assumption. But this is a contradiction. So, we can conclude that this case does not happen.

**Case 2-(v).** \(\ell_{\text{max}} > L/2\) and \(e_{\text{max}}\) is in the path from \(o\) to \(u\) (in a clockwise direction). This does not happen because \(a + b < c + d\).

**Case 3.** \(b \leq a + c + d\) and \(a + b > c + d\). The searcher visits \(y\) using the edge \((u, y)\), and goes back to \(o\) in a clockwise direction, i.e., by way of \(v\). So, DIST = \(W^* + a + b + (b + c)\). Similarly, we consider the following subcases: **Case 3-(i).** \(\ell_{\text{max}} \leq L/2\). By Fact 1, OPT = \(a + b + c + d\).

Thus,
\[
\frac{\text{DIST}}{\text{OPT}} = \frac{W^* + a + b + c + d}{a + b + c + d}
\]
\[
\leq 1 + \frac{(\sqrt{3} - 1)(c + d)}{a + b + c + d}
\]
\[
< 1 + \frac{(\sqrt{3} - 1)(c + d)}{2(c + d)}.
\]
Since the searcher visited \( u \), the searcher traversed \( e_{\max} \). Searcher was at \( u \) pose that it had already traversed \( w \) searcher visits \( w \) happen because \( a \). Not happen because \( a \). Case 3-(v). \( \ell \) \( \max \) \( L/2 \) and \( e_{\max} \) is in the path from \( o \) to \( v \) (in a counter-clockwise direction). This does not happen because \( a + b > c + d \). Case 3-(v). \( \ell \) \( \max \) \( L/2 \) and \( e_{\max} \) is in the path from \( o \) to \( u \) (in a clockwise direction). Consider the time when the searcher was at \( v \). We first show that the searcher had not yet traversed \( e_{\max} \) at this time. On the contrary, suppose that it had already traversed \( e_{\max} \) (Fig. 7(a)). Since the searcher visits \( w \) at the next step, \( \ell(v, y) > \sqrt{3}d(v, w) - W' \), where \( W' \) is the \( W \)-value at this moment. Also, by Lemma 3, \( W' \leq (1 - \sqrt{3})d(o, w). \) Hence, \( \ell(v, y) > \sqrt{3}d(v, w) - (1 - \sqrt{3})d(o, w) = \sqrt{3}d(o, v) + d(o, w) \geq \ell_{\max} \), a contradiction. So, the searcher traversed \( e_{\max} \) for the first time after it left \( v \).

Now, let \( e_{\max} = (u', u'') \) and consider the time when the searcher was at \( u' \) (Fig. 7(b)). Let \( W'' \) be the \( W \)-value at this moment. Since the searcher visited \( u'' \) next,

\[
\ell_{\max} \leq \sqrt{3}d(u', y) - W''
\]

\[
= \sqrt{3}(d(o, u') + d(o, y)) - W''
\]

\[
\leq \sqrt{3}(d(o, u) - \ell_{\max} + d(o, y)) - W''.
\]

The last inequality follows from the fact that \( d(o, u') + \ell_{\max} \leq d(o, u) \). From this inequality,

\[
\ell_{\max} \leq \frac{\sqrt{3}(d(o, u') + d(o, y)) - W''}{1 + \sqrt{3}}
\]

\[
= \frac{\sqrt{3}(a + c + d) - W''}{1 + \sqrt{3}}.
\]

Here, recall that \( L \) is the total length of the cycle. Because \( y \) is the last node visited by the searcher, the searcher does not change the direction hereafter, until it gets \( u \). Hence \( W' = W'' \) (recall that \( W' \) is the \( W \)-value when the searcher is at \( u \)). By Fact 1, \( \text{OPT} = 2(L - \ell_{\max}) \). Hence,

\[
\frac{\text{DIST}}{\text{OPT}} = \frac{W' + a + b + c + d}{2(L - \ell_{\max})} \leq \frac{2(L - \sqrt{3}(L - b) - W'')}{1 + \sqrt{3}}
\]

\[
= \frac{(1 + \sqrt{3})(W' + L)}{2(L + W' + \sqrt{3}b)} \leq 1 + \frac{\sqrt{3}}{2}.
\]

\[\square\]

4. A Tight Bound on Unweighted Graphs

In this section we consider the problem on graphs in which all edges have the same cost 1. Note that we do not restrict the topology of graphs.

4.1 An Upper Bound

The Depth-First Search (DFS) gives a good upper bound. When new edges and nodes are revealed, DFS chooses one of the unvisited nodes adjacent to the current node arbitrarily and visits it. If there is no such node, DFS backtracks, i.e., it goes back to the previous node through the edge used to come to the current node for the first time, and does the same procedure there.

To describe the behavior of DFS precisely, we give a recursive procedure Search. Inputs of Search are a node \( x \) and a sequence of nodes \( p \) (\( p \) could be empty). Intuitively, \( x \) is the searcher’s current position and \( p \) is the record of the exploration by the searcher so far.

Procedure Search\((x; \text{vertex}, p; \text{a sequence of nodes})\)

The searcher is now at \( x \). If there is an unvisited node \( z \) adjacent to \( x \), go to \( z \) and Search\((z, px)\). Otherwise,

If \( p \neq \phi \), let \( p = p'y \) where \( y \) is the last node of \( p \), and \( p' \) is a sequence of nodes obtained by eliminating \( y \) from \( p \). Go back to \( y \), and execute Search\((y, p')\).

If \( p = \phi \), halt.

Algorithm DFS

Search\((o, \phi)\)

Theorem 4: DFS is 2-competitive for unweighted graphs.
Proof. For any given graph $G$, the set of the edges that algorithm DFS traverses is a spanning tree of $G$. Let $n$ denote the number of nodes of $G$. Since DFS traverses each edge exactly twice, DFS is done in $2(n-1)$. On the other hand, OPT $\geq n$ holds because any algorithm should traverse at least $n$ edges in order to visit all the nodes and return to the origin. So, $\frac{OPT}{n} \leq \frac{2n-1}{n} < 2$. □

4.2 A Lower Bound

In this section we prove the following theorem. Note that this theorem holds even when graphs have planarity.

Theorem 5: For any positive constant $\epsilon$, there is no $(2-\epsilon)$-competitive online algorithm for unweighted graphs.

Proof. We will introduce an adversary giving the above mentioned lower bound. Fix an integer $n$ such that $n > \frac{3}{\epsilon}$. For a path $v_1, v_2, \ldots, v_k$, let $\langle\langle v_1, v_2, \ldots, v_k \rangle\rangle$ denote its total length.

First, the adversary reveals two edges $(o, u_1)$ and $(o, v_1)$ incident to the origin. If the searcher moves to $u_1$, a new edge $(u_1, u_2)$ is revealed. As long as the searcher visits a new node $u_i (i \leq n - 1)$, the adversary gives an edge $(u_i, u_{i+1})$. Similarly, if the searcher visits $v_i (i \leq n - 1)$, a new edge $(v_i, v_{i+1})$ is revealed. This procedure continues until the searcher reaches $u_n$ or $v_n$. Without loss of generality we can assume he reaches $u_n$ before $v_n$.

Now we assume that $v_1, v_2, \ldots, v_t$ have been visited, and $v_1$ has not been visited. Let $D_d$ denote the total length of the exploration so far. Because the searcher visited $v_t$, before reaching $u_n$,

$$D_d \geq 2\langle\langle o, v_1, \ldots, v_t \rangle\rangle + \langle\langle o, u_1, \ldots, u_n \rangle\rangle = n + 2t_1.$$

Then, new edges $(u_n, p_1)$ and $(u_n, q_1)$ are revealed (Fig. 8 (a)). When the searcher visits the new node $p_1 (i \leq n + t_1 - 1)$, $(p_1, p_{i+1})$ will be revealed, and when the searcher visits $q_i (i \leq n + t_1 - 1)$, $(q_i, q_{i+1})$ will be revealed (Fig. 8 (b)). Hereafter, we will do a case analysis depending on the searcher’s behavior.

First we consider the case that the searcher reaches $v_{t+1}$ before visiting $p_{n+1}$ or $q_{n+1}$. Let $t_2 (\leq n + t_1 - 1)$ and $t_3 (\leq n + t_1 - 1)$ be integers such that $p_{t_2}$ and $q_{t_3}$ have been visited, and $p_{n+1}$ and $q_{n+1}$ are unvisited (Fig. 8 (c)). The adversary does not reveal new edges anymore. Let $D_e$ denote the total length of the exploration so far. The searcher moved from $u_n$ to $v_{t+1}$ after visiting $p_1$ and $q_1$, so

$$D_e \geq D_d + 2\langle\langle u_n, p_1, p_2, \ldots, p_{n} \rangle\rangle + 2\langle\langle u_n, q_1, q_2, \ldots, q_{n} \rangle\rangle + \langle\langle u_n, u_{n+1}, \ldots, u_1, o, v_{t+1} \rangle\rangle = D_d + 2t_2 + 2t_3 + n + t_1 + 1 \geq 2n + 3t_1 + 2t_2 + 2t_3 + 1.$$

Hereafter, the searcher visits $q_{t_3+1}$ and $p_{n+1}$, and returns to $o$ finishing exploration. The total distance is

$$\text{ALG} \geq D_e + 2\langle\langle v_{t+1}, v_{t+2}, \ldots, o, u_1, \ldots, u_n \rangle\rangle + 2\langle\langle u_n, q_1, q_2, \ldots, q_{n+1} \rangle\rangle + \langle\langle u_n, u_{n+1}, \ldots, o \rangle\rangle = D_e + 2(n + t_2 + t_3 + 3) + 2t_3 + 1$$

while OPT $= 2(n + t_1 + t_2 + t_3 + 3)$. So, $\text{ALG} > 2 - \epsilon$.

Secondly, we consider the case that the searcher reaches $p_{n+1}$ or $q_{n+1}$ before visiting $v_{t+1}$. Without loss of generality we can assume that the searcher reaches $q_{n+1}$. Now suppose that $p_{t_2}$ has been visited and $p_{n+1}$ is unvisited. Let $D_d$ denote the total length of the exploration so far. By a similar observation as before,

$$D_d \geq D_e + 2\langle\langle u_n, p_1, p_2, \ldots, p_{n+1} \rangle\rangle + \langle\langle u_n, q_1, q_2, \ldots, q_{n+1} \rangle\rangle = D_e + 2t_4 + n + t_1 \geq 2n + 3t_1 + 2t_4.$$
the tour is
\[
\text{ALG} \geq D_d + \ell(q_{n+1}, v_{n+1}) + \\
\langle v_{1}, q_{n+1}, q_{n+1}, \ldots, u_n, p_1, \ldots, p_{t+1} \rangle + \\
\langle p_{t+1}, p_{t+1}, \ldots, u_n, u_{n-1}, \ldots, o \rangle \\
= D_d + 1 + 1 + n + t_1 + t_4 + 1 + \\
1 + t_4 + n \\
\geq 4(n + t_1 + t_4 + 1).
\]
The total length of an optimal offline tour is
\[
\text{OPT} = \langle o, v_1, \ldots, v_{t+1} \rangle + \ell(v_{t+1}, q_{t+1}) + \\
\langle q_{t+1}, q_{t+1}, \ldots, q_1, u_n, p_1, \ldots, p_{t+1} \rangle + \\
\langle p_{t+1}, p_{t+1}, \ldots, p_1, u_n, u_{n-1}, \ldots, u_1, o \rangle \\
= (t_1 + 1) + 1 + (n + t_1 + t_4 + 1) + \\
(t_4 + 1 + n) \\
= 2(n + t_1 + t_4 + 2).
\]
So, \( \frac{\text{ALG}}{\text{OPT}} \geq 2 - \epsilon. \)

5. Concluding Remarks

In this paper, we have studied the online graph exploration problem on two graph classes. First, we have given a tight competitive ratio of \( \frac{1+\sqrt{5}}{2} \) for the problem on cycles. We have also studied the problem on unweighted graphs and have given a tight bound of 2.

For planar graphs, the best known upper bound is 16, as mentioned in Sec. 1. Since Theorem 5 holds for planar graphs, 2 is the current best lower bound for planar graphs. There still remains a large gap between these upper and lower bounds. Narrowing the gap is a challenging problem. Another future work is to consider randomized algorithms to break deterministic lower bound.

Acknowledgements

The authors would like to thank anonymous reviewers for their helpful comments. This work was supported by KAKENHI (20700009).

References


Shuichi Miyazaki is an associate professor of Academic Center for Computing and Media Studies, Kyoto University, Kyoto, Japan. He received BE, ME, and Ph.D. degrees from Kyoto University in 1993, 1995 and 1998, respectively. His research interests include algorithms and complexity theory.

Naoyuki Morimoto is a student of Graduate School of Informatics, Kyoto University, Kyoto, Japan. He received BE and ME degrees from Kyoto University in 2006 and 2008, respectively. His research interests include online algorithms.

Yasuo Okabe is a professor of Academic Center for Computing and Media Studies, Kyoto University, Kyoto, Japan. He received BE and ME degrees from Kyoto University in 1993, 1995 and 1998, respectively. His research interests include Internet architecture, ubiquitous networking and distributed algorithms. He is a member of IPSJ, JSSST, IEEE, ACM and EATCS.