Directed Graph Exploration

Klaus-Tycho Foerster and Roger Wattenhofer

Computer Engineering and Networks Laboratory,
ETH Zurich, 8092 Zurich, Switzerland
{k-t.foerster,wattenhofer}@tik.ee.ethz.ch

Abstract. We study the problem of exploring all nodes of an unknown directed graph. A searcher has to construct a tour that visits all nodes, but only has information about the parts of the graph it already visited. The goal is to minimize the cost of such a tour. In this paper, we present upper and lower bounds for both the deterministic and the randomized online version of exploring all nodes of directed graphs. Our bounds are sharp or sharp up to a small constant, depending on the specific model. Essentially, exploring a directed graph has a multiplicative overhead linear in the number of nodes. If one wants to search for just a node in unweighted directed graphs, a greedy algorithm with quadratic multiplicative overhead can only be improved by a factor of at most two. We were also able to show that randomly choosing a starting point does not improve lower bounds beyond a small constant factor.

Keywords: : *online algorithms, graph exploration, mobile agents and autonomous robots*

1 Introduction

The hotel concierge promised that this tourist attraction is easy to find, just a short drive in your car, and she was right. However, how do you now get back to your hotel, in this cursed city full of one-way streets? After finally being back at your hotel, totally exhausted, you have a hunch that one-way streets render navigation more difficult, but is it true?!

In this paper we quantitatively analyze navigation problems in unknown directed graphs from a worst-case perspective. We present a whole flurry of tight upper and lower bounds, showing that directed graphs exhibit a penalty in the order of the number of nodes of the graph.

Navigation problems in directed graphs are not restricted to the playful introductory example of one-way streets. Staying in the car context, if we are for instance interested in minimizing gasoline cost, any hill-side city becomes directed, as driving downhill is virtually free, whereas driving uphill may incur a high cost. As such, when applying a cost measure, edges of a graph must often be represented by two directed edges with an appropriate cost.

The most important applications for investigating navigation in directed graphs are however beyond street networks. In computer networks, for instance,

directed graphs have for instance been studied in the context data aggregation [29], routing [33], or traversing social networks [34]. Brass et. al. [7] compared the exploration of directed graphs to exploring the state space of a finite automaton, where the states are nodes and the transitions are edges. Deng and Papadimitriou [15] proposed the exploration of directed graphs as a model for learning, for example for a newborn: current states can be detected by sensor information (like eyes or ears) and possible actions leading to other states are known, but it is not known what the situation will be at a not yet explored state. And last not least, exploring an unknown graph is considered one of the fundamental problems in robotics [11, 25]. Because of all these applications, directed graph exploration will be the main focus in this paper. In addition, we look at other navigation problems, such as searching for a node, which turn out to be related to exploration.

1.1 Model

We only consider the common model of strongly connected directed graphs [1, 14, 15, 25, 30], since a searcher else might get stuck right away (Section 8). We call a graph explored, if a searcher starting from some node s has visited all nodes and returned to s. The cost of such an online exploration tour is measured by the total sum of the weight of the traversed edges. It is allowed (and might be necessary) to visit nodes multiple times, but if we traverse an edge again it costs the same as for the first time. The competitive ratio of a tour is measured by the ratio of the cost of the tour divided by the cost of a tour of minimum cost. The competitive ratio of an algorithm is measured by the largest competitive ratio of all tours generated over all input graphs. For randomized algorithms, it is the largest expected competitive ratio.

For ease of notation, in the remainder of our paper a graph $G = (V, E)$ has $|V| = n \geq 6$ nodes and $|E| = m$ edges. All nodes have unique IDs, and all edges have non-negative weights. A searcher has unlimited computational power and memory and may only traverse edges from tail to head. Upon arriving at a node v, the following information is made available: all outgoing incident edges including their weight, plus the IDs (cf. [27, 31]) of the corresponding nodes at the head of these edges. Graph exploration is an online problem since only partial information about the graph is available [9, 10]. For other exploration models, e.g. unique edge names or information about incoming edges, we refer to Section 8.

1.2 Results

In our paper we give the first matching lower and upper bounds for the competitive exploration of an unknown directed graph. Our results are sharp for both the weighted and the unweighted case. For randomized exploration, our results only have a gap of less than four. We prove similar results for various commonly used graph classes, like planar or complete graphs or bounding different parameters like degree or diameter. We also discuss changes in the model,

like randomly choosing a starting position or more powerful searchers. We are able to show that in all these cases, the exploration of unknown directed graphs has a multiplicative overhead of $\Theta(n)$.

In a similar fashion, searching for a single node has $\Theta(n^2)$ overhead if all edges have unit weight (see Section 6). Furthermore, we look at the impact of randomly choosing a starting point. It turns out that even the best possible starting node can decrease any lower bound only by a factor of at most four.

To the best of our knowledge, sharp results regarding deterministic and randomized exploration of directed graphs have not yet been published. We summarize our main results in Table 1.

Table 1. Short overview of our main results: In the weighted general case we only need to use two different edge weights to achieve the bounds. A randomized starting node can only decrease our lower bounds by a factor of four.

type of graph \ competitivity	lower bound	upper bound	multiplicative gap
(deterministic) general*^{c}	$n-1$	$n-1$	sharp
(randomized) general*$^{+c}$	$\frac{n}{4}$	$n-1$	≤ 4
(determ.) unweighted general*	$\frac{n}{2}+\frac{1}{2}-\frac{1}{n}$	$\frac{n}{2}+\frac{1}{2}-\frac{1}{n}$	sharp
(random.) unweighted general*	$\frac{n}{8}+\frac{3}{4}-\frac{1}{n}$	$\frac{n}{2}+\frac{1}{2}-\frac{1}{n}$	≤ 4
(deterministic) euclidean planar	$n-2-\bar{\epsilon}$	$n-1$	$\leq 1.25+\epsilon$
(randomized) euclidean planar	$\frac{n}{4}-\bar{\epsilon}$	$n-1$	$\leq 4+\epsilon$
(d.) unit weight euclidean planar	$\frac{n}{4}+\frac{1}{2}-\frac{2}{n}$	$\frac{n}{2}+\frac{1}{2}-\frac{1}{n}$	≤ 2
(r.) unit weight euclidean planar	$\frac{n}{8}+\frac{3}{4}-\frac{1}{n}$	$\frac{n}{2}+\frac{1}{2}-\frac{1}{n}$	≤ 4

* also applies to planar graphs and graphs that satisfy the triangle inequality

^{c} also applies to complete graphs and graphs with any diameter from 1 to $n-1$

$^+$ also applies to graphs with any maximum incoming/outgoing degree from 2 to $n-1$ and to graphs with any minimum incoming/outgoing degree from 1 to $n-1$

2 Related Work

The offline variant, i.e. where all information about the graph is available to the algorithm, of directed graph exploration is the asymmetric travelling salesperson problem, where it is allowed to visit nodes multiple times. Unlike the undirected case, there is no known polynomial approximation algorithm with constant approximation ratio [3]. An approximation ratio of $O(\log n)$ was achieved in [22], the constant was improved over time, e.g. [8, 28]; the best result known to us is $\frac{2}{3}\log_2 n$ [19]. There exists a result of $O(\log n/\log\log n)$ for the randomized

case [2]. If only the edge weights 1 and 2 are allowed, it is approximable with a ratio of 17/12 [37], with a NP-hard lower bound of $2805/2804 - \epsilon$ [18]. An online variant of asymmetric TSP is as follows: A searcher knows the graph, but the nodes to visit get determined during the runtime by an adversary [3].

More closely related to the online exploration of all nodes of directed graphs is the online exploration of all nodes of undirected graphs. While a greedy algorithm achieves a competitive ratio of $\Theta(\log n)$ [35], it is not known if a constant competitive ratio for general graphs is possible [31]. For cycles there is an algorithm with a sharp competitive ratio of $\frac{1+\sqrt{3}}{2}$, while for trees depth-first search is optimal [32]. Recently, the best known lower bound for general graphs was improved from $2 - \epsilon$ [32] to $5/2 - \epsilon$ [16]. For planar graphs a sophisticated variant of depth-first search named ShortCut by Kalyanasundaram and Pruhs achieves a competitive ratio of 16 [27]. Their result was recently extended for graphs of genus g to $16(1 + 2g)$ [31]. If there are just k different edge weights, there exists an algorithm with competitive ratio $2k$ [31]. Fleischer et. al. considered the problem of searching just for a node instead of a tour in [23]. They model their searcher as "blind", meaning that it can only sense the outgoing edges, but not any incoming edges or adjacent neighbors. They use the example of a modified clique to show a lower bound on the cost of $\Omega(n^2)$ for unit weights, since a blind searcher might visit nearly all edges.

Another related problem is the exploration of all edges of a strongly connected directed graph. Here the difficulty of the problem depends on another parameter, introduced by Kutten [30]: the eulerian deficiency d of a graph, which is the minimum amount of edges that need to be added to make the graph eulerian. A graph is eulerian, if there exists a path that visits all edges exactly once. If a graph is eulerian, then it can be traversed in an online fashion with at most $2m$ edge traversals [14], which directly implies at most $4m$ edge traversals in the undirected case, see for example [1]. For $d = 1$, a ratio of 4 is optimal [15]. An upper bound only dependent polynomially in d for the directed case was given by Fleischer and Trippen [25], their algorithm is $O(d^8)$-competitive. There exists also a lower bound of $\Omega(d)$-competitivity for the deterministic case and a lower bound of $\Omega(\frac{d}{\log d})$-competitivity for the randomized case [14, 15]. Furthermore, graph exploration has also been considered with restricted memory models or multiple searchers, see for example [4, 6, 12, 13, 17, 20, 21].

There seems to be no known randomized algorithm for the exploration of graphs (wether it be just nodes or edges) that gives better bounds than the known deterministic algorithms. Experimental studies of randomized algorithms for exploring all edges and nodes of a strongly connected directed graph have been done in [24].

The similar sounding term graph searching, which was first discussed by Breisch and Parsons (cf. [5]), stands for another problem: A number of agents has to capture an intruder, or as formulated in the original papers, a party of searchers has to find a person lost in a cave. For an overview of other online navigation tasks we refer to [10].

3 Lower Bounds for General Graphs

We note that in this section we only use the weights 0 and 1 in the weighted case for lower bounds. If only integers of size at least one are allowed as edge weights, then analog results can be achieved by replacing 0 with 1 and 1 with $\lceil 1/\epsilon \rceil$ for arbitrarily small $\epsilon > 0$. Furthermore, the unique names of nodes in the remainder of the paper are just fixed for the convenience of the reader, an adversary can permute them in any way it desires – therefore an online algorithm can derive no further information from just the unique name of an unexplored node. Also the graphs used in the lower bounds are planar and satisfy the triangle inequality.

3.1 Deterministic Online Algorithms

Theorem 1. *No deterministic online algorithm can achieve a better competitive ratio on exploring all nodes of strongly connected directed weighted graphs than* $n-1$.

Proof. Consider the graph in Figure 1. A searcher using any deterministic online algorithm starting at node v_n cannot differentiate between the nodes v_1,v_2,..., v_{n-1}, they all look the same, since it can only see the outgoing edges from v_n and the nodes at the end of these edges. In the worst case, the searcher chooses to visit the node v_{n-1} first, then is forced to go back to v_n, then to visit v_{n-2} and so on, until it visits v_1 and then returns to v_n. The cost of this route is $n-1$, while an optimal tour first visits v_1 and then goes to v_n, inducing a total cost of just 1. This yields a competitive ratio of $n-1$ for any deterministic online algorithm. □

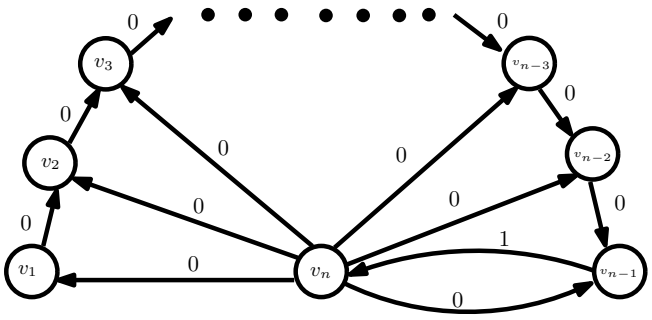

Fig. 1. In this graph the starting node s is v_n in the lower middle of the image. A deterministic algorithm can get tricked into first visiting v_{n-1}, then v_{n-2} and so on.

3.2 Randomized Online Algorithms

A randomized searcher can explore the graph in Figure 1 with much lower expected costs: In average it chooses a node in the "middle" of the so far yet unvisited nodes when being at v_n, therefore visiting the starting node only about $O(\ln(n))$-times. However, we can reach nearly the same lower bounds with the graph from Figure 2 as in the deterministic case:

Theorem 2. *No randomized online algorithm can achieve a better competitive ratio on exploring all nodes of strongly connected directed weighted graphs than* $\frac{n}{4}$.

Proof. Consider the graph in Figure 2 and let the number of nodes n be even. If one wants to consider odd n, then the same results can be achieved by removing the node $v_{\frac{n}{2}}$ and updating the graph accordingly. Let us assume a searcher using any randomized online algorithm starting from v_n visits a node v_i, with $1 \leq i \leq \frac{n}{2} - 2$, for the first time: then it cannot differentiate the two outgoing edges. An adversary can choose the IDs so that a good edge is picked with a probability of at most $p = 0.5$. Thus the decisions at the nodes v_1 to v_{i-1} do not yield any useful information about how to pick the outgoing edges at v_i. Therefore the expected amount of choosing a wrong outgoing edge is $0.5\left(\frac{n}{2} - 2\right)$. A wrongly chosen edge when visiting v_i for the first time induces a cost of 1, since the searcher has to follow the unique way back to v_i, traversing the edge from v_{n-1} to v_n with cost 1. This results in an expected cost of $0.5\left(\frac{n}{2} - 2\right) = \frac{n}{4} - 1$ to explore the node $v_{\frac{n}{2}-1}$. Once reaching the node $v_{\frac{n}{2}-1}$ for the first time, the searcher is forced to go back to v_n, resulting in another cost of 1. Since an optimal tour has a cost of 1, this yields the lower bound of $\frac{n}{4}$. □

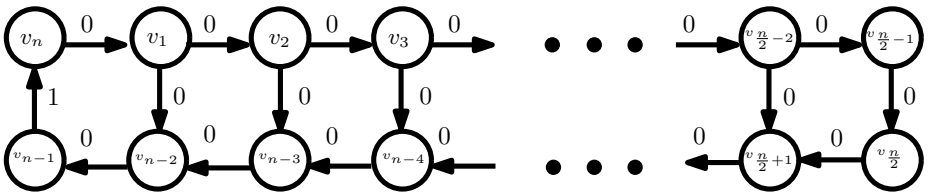

Fig. 2. In this graph the starting node s is v_n in the upper left corner. Upon arriving at each of the nodes $v_1, v_2, \ldots v_{\frac{n}{2}-2}$ for the first time, a randomized algorithm gets tricked into taking the wrong edge with probability at least 0.5. If n is odd, then the lower right node $v_{\frac{n}{2}}$ can be removed to achieve the lower bound.

3.3 Starting Node

While the examples of the graphs in the Figures 1 and 2 lead to a high lower bound for the competitive ratio, this is only true because the online algorithm is

forced to start at the node v_n. Starting at node v_1 in Figure 1 or at node $v_{\frac{n}{2}}$ in Figure 2 leads to a competitive ratio of 1. If the starting node were to be chosen randomly, the expected ratio is $O(\sqrt{n})$ for both cases. This raises the question if a random starting node can lead to a better competitive ratio. However this is not the case, there is still a lower bound of $\Omega(n)$:

Theorem 3. *Even if taking the best result from all possible n starting nodes, no deterministic online algorithm can achieve a better competitive ratio on exploring all nodes of strongly connected directed weighted graphs than $n/4$. The same holds for randomized online algorithms with a competitive ratio of $n/16$.*

Proof. We start with the deterministic case. We again take the graph from Figure 1, but draw it two times as G and G' with their respective starting nodes v_n and v'_n. We now connect these graphs by adding an edge from v_n to v'_n and back, both with weight 0 – resulting in a graph G'' with $2n$ nodes. Without loss of generality we can assume that a starting node from G' is chosen. No deterministic online algorithm can achieve a better worst-case cost on exploring G than $n-1$, since the old graph G can only be entered by the edge from v'_n to v_n. On the other hand, the graph G' can be explored with a cost of just 1. An optimal offline algorithm will just have a cost of 2 for exploring the whole graph, no matter what starting node is chosen. Since the graph has $2n$ nodes, this leads to a lower bound of $n/4$. We can apply the same arguments to the randomized case using the graph in Figure 2, giving a lower bound of $n/16$. □

4 Upper Bounds for General Graphs

In the undirected case, it is not known yet if there is an algorithm with a better competitive ratio than $O(\log n)$ [35]. A greedy approach reaches this competitive ratio of $O(\log n)$ [35], but the same algorithm has a competitivity of $\Omega(\log n)$ even on planar unweighted graphs [26]. Thanks to our strong lower bounds, a greedy algorithm has a sharp competitive ratio in the directed case:

Theorem 4. *A greedy algorithm achieves a competitive ratio of $n-1$ for exploring all nodes of strongly connected directed weighted graphs.*

Proof. Given any graph $G = (V, E)$, let us fix an optimal tour OPT. The tour OPT can be viewed as a concatenation of n paths, that visit the nodes of the graph in the following order: $s = v_0^o, v_1^o, v_2^o, \ldots, v_{n-1}^o, v_n^o = s$. We name the path from v_i^o to v_{i+1}^o as w_{i+1}^o with $0 \leq i \leq n-1$. The walk $W_{i,j}^o$ from v_i^o to v_j^o (with $v_i^o \neq v_j^o$) in OPT consists of the concatenation of $w_{i+1}^o, w_{i+2}^o, \ldots, w_j^o$ for $i < j$ or of $w_{i+1}^o, w_{i+2}^o, \ldots, w_n^o, w_1^o, \ldots, w_{j-1}^o, w_j^o$ for $i > j$. For each $W_{i,j}^o$ with $i \neq j$ it holds that $W_{i,j}^o$ is the concatenation of at most $(n-1)$ different paths w_r^o with $1 \leq r \leq n$. Let us assume the greedy algorithm proceeds as follows: upon reaching a node v_k^g for the first time, find a shortest path w_{k+1}^g from the current node to a unknown node v_{k+1}^g in the outgoing neighborhood of the so far explored nodes. This path w_{k+1}^g has at most the weight of the concatenated paths from

v_k^g to v_{k+1}^g in OPT. Let us assume it has heavier weight: then there is a cheaper path from v_k^g to v_{k+1}^g that also visits another not yet explored node v_q before visiting v_{k+1}^g. However by the choice of v_{k+1}^g, then v_q is the same node as v_{k+1}^g, which leads to a contradiction. If we sum this up for all n paths $w_1^g, \ldots, w_{n-1}^g$ from the greedy algorithm plus the shortest path w_n^g from v_{n-1}^g to $s = v_n^g$, a first simple upper bound is $n \cdot |OPT|$. However, each path w_r^o with $1 \leq r \leq n$ from OPT only gets used at most $(n-1)$ times in the upper bound. This leads to an upper bound of $(n-1) \cdot |OPT|$ on the cost of a tour produced by the greedy algorithm. □

A combination of Theorem 1, 2 and 4 yields the following corollary:

Corollary 1. *The result of Theorem 4 cannot be improved by any other deterministic online algorithm. For randomized online algorithms, only a improvement by a factor of at most 4 is possible.*

Furthermore, the authors of [16] also studied the problem of advice complexity for exploring undirected graphs. They showed that there is a family of graphs where a searcher needs to be given at least $\Omega(n \ln(n))$ bits of information (from an all-knowing outside source before starting) to explore the graphs with optimal cost. We note that a greedy algorithm in any directed or undirected graph can solve the graph exploration problem optimally with $O(n \ln(n))$ bits. We apply the arguments from above and give a list of the nodes from an optimal tour OPT in the order they first appear in OPT to the searcher. Since the ID of every node is of size $O(\ln(n))$ bits, the lower bound of $\Omega(n \ln(n))$ from [16] is a sharp bound of $\Theta(n \ln(n))$ bits for both directed and undirected graphs.

5 Unweighted Graphs

An unweighted graph is a graph where the edges have no edge weights, i.e. the cost is the same for all edges. For our purposes, this is the same as assigning the edge weight 1 to every edge. The lower bounds are lower, but we will see that the upper bounds also go down:

Theorem 5. *No online algorithm can achieve a better competitive ratio on exploring all nodes of strongly connected directed unweighted graphs than* $\frac{n}{2} + \frac{1}{2} - \frac{1}{n}$ *(deterministic) or* $\frac{n}{8} + \frac{3}{4} - \frac{1}{n}$ *(randomized) .*

Proof. Consider the graph in Figure 1 for the deterministic case and assign all edges a weight of 1. A deterministic online algorithm starting at v_n first visits v_{n-1}, then v_{n-2} etc. in the worst case. Exploring v_{n-1} and going back to v_n has a cost of 2, for v_{n-2} it is 3, ..., for v_1 it is n. Summed up this yields $2+3+\ldots+n = \frac{n^2}{2} + \frac{n}{2} - 1$. Since an optimal tour has cost n, this gives a lower bound for the competitive ratio of $\frac{n}{2} + \frac{1}{2} - \frac{1}{n}$.

Consider the graph in Figure 2 for the randomized case and assign all edges an edge weight of 1. Now we can apply the same argument as in the weighted case, but the induced cost by each wrong decision is not 1, but 4 for v_1, 6 for

v_2, ..., $n-2$ for $v_{\frac{n}{2}-2}$. Since the previous decisions are useless for the current decision, each of these wrong decisions happens with a probability of at least 0.5. Furthermore, independently of these decisions, the last exploration tour starting at v_n will visit all nodes exactly once in this example. This gives a lower cost bound of $0.5\left(\left(\frac{n}{2}-2\right)^2+3\left(\frac{n}{2}-2\right)\right)+n=\frac{n^2}{8}+\frac{3n}{4}-1$. An optimal tour has cost n, resulting in a lower bound for the competitive ratio of $\frac{n}{8}+\frac{3}{4}-\frac{1}{n}$. □

Theorem 6. *A greedy algorithm achieves a competitive ratio of $\frac{n}{2}+\frac{1}{2}-\frac{1}{n}$ for exploring all nodes of strongly connected directed unweighted graphs.*

Proof. We prove this upper bound by summing up the costs to reach the first newly explored node, the second newly explored node, ..., the $(n-1)$th (and last) newly explored node. Let us assume that, beside the starting node, we have explored $(k-2)$ additional nodes and have just reached the $(k-1)$th new node v_{k-1} for the first time. Since the graph is strongly connected, there is always at least one new node reachable from the current node in the neighborhood of the so far explored subgraph – unless every node has been visited already. If we pick the new node v_k as a unexplored one we can reach with as few edge-traversals as possible, then we induce a cost of at most k. A shortest path from v_{k-1} to v_k will by definition not include another unexplored node v_u, since then v_u had been chosen as v_k. Furthermore, the path will not include any node twice. This gives an upper bound of k for the length of the path from v_{k-1} to v_k. In order to get back to the starting node once all nodes are explored, a shortest path can again visit at most all other $n-2$ nodes before reaching the starting node, giving an upper bound of $n-1$ for this last path. If we sum this up we get an upper bound of $1+2+3+\ldots+(n-2)+(n-1)+(n-1)=-1+\sum_{i=1}^{n} i=\frac{n^2}{2}+\frac{n}{2}-1$. An optimal tour has cost at least n, giving a competitive ratio of at most $(\frac{n^2}{2}+\frac{n}{2}-1)/n=\frac{n}{2}+\frac{1}{2}-\frac{1}{n}$. □

Combining the results of Theorem 5 and Theorem 6 yields:

Corollary 2. *The result of Theorem 6 cannot be improved by any other deterministic online algorithm. For randomized online algorithms, only a improvement by a factor of at most 4 is possible.*

6 Searching a Node

Instead of generating a tour, one can also change the model, and find just one specific node v and then stop. However an adversary can place this node in such a way that it is found last. The searcher does not need to return to the start, but searching for a node is still costly:

Theorem 7. *Searching for a node in strongly connected directed weighted graphs has an arbitrarily large competitive ratio for any deterministic or randomized online algorithm and can induce arbitrarily large additive costs.*

Proof. We start with the deterministic case. In Figure 1, a node v_{n+1} can be added that is connected to v_1 with two edges of weight 0. Since an optimal algorithm finds this node with cost 0, any deterministic node search algorithm has an arbitrarily bad competitive ratio, since it induces positive costs. The same holds for randomized algorithms if the same construction is applied at node $v_{\frac{n}{2}-1}$ in Figure 2. We can apply the same thought for arbitrarily large additive costs by replacing the edge weight of 1 with an arbitrarily large value. □

If we consider the model of unit weight edges, then the situation changes:

Theorem 8. *Any online algorithm for searching a node in strongly connected directed unweighted graphs has a lower bound of* $\frac{(n-1)^2}{4} - \frac{(n-1)}{4} - \frac{1}{2}$ *(deterministic) or* $\frac{(n-1)}{4} + \frac{1}{2} + \frac{2}{(n-1)}$ *(randomized) for its competitive ratio.*

Proof. An optimal offline algorithm has a cost of 2 to find v_{n+1} in the modified graph from Figure 1 with unweighted edges (v_n to v_1 to v_{n+1}). Any deterministic online algorithm finds v_{n+1} last in the worst case, producing a cost of at least (see the proof of Theorem 5) $\frac{n^2}{2} + \frac{n}{2} - 1 - n$. The searcher does not have to go back to the start, so $(-n)$ is added at the end. Since this graph has $(n+1)$ nodes, a lower bound for the competitive ratio of any deterministic node search algorithm is $\frac{(n-1)^2}{4} - \frac{(n-1)}{4} - \frac{1}{2}$.

For the randomized case we use the modified graph from Figure 2 and search for the node $v_{\frac{n}{2}}$. An optimal algorithm finds $v_{\frac{n}{2}}$ after $\frac{n}{2}$ steps (v_n to v_1 ...to $v_{\frac{n}{2}-1}$ to v_{n+1}). Any randomized algorithm needs at least an expected cost of (see the proof of Theorem 5) $\frac{n^2}{8} - \frac{n}{4} - 1 + \frac{n}{2}$. This leads to a competitive ratio of $\left(\frac{n^2}{8} + \frac{n}{4} - 1\right) / \frac{n}{2} = \frac{n}{4} - \frac{2}{n} + \frac{1}{2}$. □

For an upper bound we can again use the greedy algorithm:

Theorem 9. *A greedy algorithm searching for a node in strongly connected directed unweighted graphs has a competitive ratio of* $\frac{n^2}{4} - \frac{n}{4}$.

Proof. A greedy algorithm finds the searched node last in the worst case with a cost of at most $\frac{n^2}{2} - \frac{n}{2}$ (see the proof of Theorem 6). If the node were to be directly reachable from the starting node, then an online algorithm can find it in one step. Therefore we can use 2 as the minimal cost needed for an offline algorithm when computing an upper bound for the competitive ratio. This leads to a competitive ratio of $\frac{n^2}{4} - \frac{n}{4}$. □

Combining Theorem 8 and Theorem 9 yields the following corollary:

Corollary 3. *Any deterministic online algorithm searching for a node in strongly connected directed unweighted graphs can improve the competitive ratio of a greedy algorithm by a factor of 3 at most.*

Proof. The quotient of the upper and lower bounds from Theorem 9 and 8 for $n \geq 4$ (for $n \leq 3$ any node search takes two steps at most) has a global maximum in the range $n \in [4, \infty)$ at $n = 4$ with value 3. □

Let us now come back to the situation mentioned at the start of our introduction. How expensive can going back to your hotel be? Essentially, it is the same as searching for a node – just that this node is the only one that has an outgoing edge to your hotel. For the deterministic case, we again use the graph from Figure 1 with unweighted edges. We add a hotel-node v_h and add a directed edge from v_h to v_n (the node with the tourist attraction) and one directed edge from v_1 to v_h. Going back to your hotel is now the same as searching the node v_1 with one additional step back. The same construction can be used for the randomized case with the graph from Figure 2 with unweighted edges. We add a hotel-node v_h and add an outgoing edge from v_h to v_n (again, the node with the tourist attraction) and one outgoing edge from $v_{\frac{n}{2}}$ to v_h.

7 Lower Bounds for Special Cases

When we add directed edges with arbitrarily high weights to a given graph, then using these edges in any online algorithm will not improve the weight of an obtained tour. An online algorithm has now more information about the graph (for example about the number of nodes), but we can add these edges in such a way to our lower bound graphs in Figure 1 and 2 that the searcher gains no useful information. For example, if we turn the graph from Figure 1 into a complete graph by adding all missing edges with arbitrarily high weights, then these new edges do not help a searcher on deciding what node to explore next when visiting v_n, since all possibilities look the same except for their ID – unless the searcher decides to use an expensive edge. Due to space constraints, we omit the proofs of the Theorems 10 and 11 in this section:

Theorem 10. *For graphs of any diameter from 1 to $n-1$ or complete graphs with eulerian deficiency of $d = 0$, no online algorithm can achieve a better competitive ratio on exploring all nodes of strongly connected directed weighted graphs than $n-1$ (deterministic) or $n/4$ (randomized).*

We can apply the same line of thought to the graph in Figure 2:

Theorem 11. *For graphs of any maximum (minimum) incoming/outgoing degree from 2(1) to $n-1$, no online algorithm can achieve a better competitive ratio on exploring all nodes of strongly connected directed weighted graphs than $n/4$.*

A graph is called euclidean, if its nodes can be embedded into the euclidean plane with the edge weights being equivalent to the length of the straight edge in the embedding [36].

Theorem 12. *No online algorithm can achieve a better competitive ratio on exploring all nodes of strongly connected directed weighted planar euclidean graphs than $n-2-\epsilon^*$ (deterministic) or $n/4-\epsilon^*$ (randomized) for any $\epsilon^* > 0$.*

Proof. We again consider the graph in Figure 2 for the randomized case. If we replace all edge weights with a fixed $\epsilon^r > 0$, then it can be embedded as a

planar euclidean graph like shown in the figure. To reach the lower bound for competitivity, we replace both edge weights of the incoming and the outgoing edge for v_{n-1} with $1/2$. Now let us consider a circle with radius $\frac{1}{2}$ through the nodes v_n and v_{n-2}, with the nodes v_1 and v_{n-3} not being inside the circle. If we place v_{n-1} in the center of the circle, we have a proper planar euclidean embedding of the constructed graph. By choosing ϵ^r to be small enough, for example $\epsilon^r < \epsilon^*/n^2$, we reach a lower bound of $n/4 - \epsilon^*$.

For the deterministic case we consider the graph in Figure 1. Let us fix a $\epsilon^d > 0$ and construct a cycle of radius ϵ^d with v_n being in the center of the cycle and placing the nodes v_1 to v_{n-1} with distance ϵ^d/n on the cycle. Like in the randomized case, we construct another circle of radius $\frac{1}{2}$ through the nodes v_n and v_{n-2}, with v_1 and v_{n-3} not being inside the circle. We now place v_{n-1} in the middle of that cycle, which means that the edge weights of both the incoming and the outgoing edges are $1/2$. We remove the edge from v_n to v_{v-1}, since it has no longer the same weight than the other outgoing edges from v_n. All other edge weights are now $\leq \epsilon^d$. Notice that a deterministic algorithm can now only be tricked $n-2$ times. If we choose $\epsilon^d \leq \epsilon^*/n^2$, we reach a lower bound of $n - 2 - \epsilon^*$. □

A similar result also holds if all edge weights have to be of unit weight:

Corollary 4. *No online algorithm can achieve a better competitive ratio on exploring all nodes of strongly connected directed unit weight planar euclidean graphs than* $\frac{n}{4} + \frac{1}{2} - \frac{2}{n}$ *(deterministic) or* $\frac{n}{8} + \frac{3}{4} - \frac{1}{n}$ *(randomized).*

Proof. We use the graph from Figure 2 (see Theorem 5) and set all edge weights to 1, which results in a unit weight euclidean planar graph. □

8 Other Exploration Models

Unique Edge Names: Our results also hold if the searcher cannot see the name of nodes at the end of incident outgoing edges, but just the unique name of both incoming and outgoing edges. When two nodes v_i and v_j are visited by the searcher, it knows the name of all incident edges for v_i and v_j, therefore also the subgraph that is spanned by v_i and v_j. If the searcher is at a node v_i and does not know where an incident outgoing edge ends, then the node at the end of that edge has not been explored yet. In other words, the searcher has visited all nodes if and only if it knows where each edge ends and starts. Since our greedy algorithms do not utilize node names when selecting the next node to be explored, but just try to get to a unexplored node as cheap as possible, our upper bounds still apply. This holds as well for our lower bound examples if we use this modified exploration model: every time we trick any online algorithm into making a wrong decision, we give a set of options to choose from that look exactly the same for the online searcher.

Incoming Edges: Let us assume that the searcher does not just see the names

of the nodes at the end of incident outgoing edges, but also the names of the nodes at the other end of incident incoming edges. Our upper bound still applies, since the algorithms can just choose to ignore that additional information. For the lower bound however, we can no longer use the graphs from Figure 1 and Figure 2. For example when starting on the graph in Figure 1, the node v_{n-1} now can be differentiated from the nodes $v_1, \ldots, v_{n-2}$. Also when visiting v_{n-2}, the searcher can now differentiate v_{n-3} from $v_1, \ldots, v_{n-4}$, since there is an edge from v_{n-3} to v_{n-2}. We can fix this problem by hiding this information with adding additional nodes. For the example in Figure 1, we add $n-1$ additional nodes. For $1 \leq i \leq n-1$, remove the edge from v_i to v_{i+1}, add a new node v_i^+ between them and add a edge from v_i to v_i^+ and from v_i^+ to v_{i+1}. The edge weights of the two new edges is one half of the edge weight of the removed edge. This decreases the lower bound by a factor of less than 2. We fix the graph in Figure 2 in a similar way. We add $\frac{n}{2}-3$ nodes between the nodes $v_{\frac{n}{2}+1}$ to v_{n-2}. For $\frac{n}{2}+1 \leq i \leq n-2$, remove the edge from v_i to v_{i+1} add a new node v_i^+ between them and add a edge from v_i to v_i^+ and from v_i^+ to v_{i+1}. The edge weights of the two new edges are one half of the edge weight of the removed edge. This decreases the lower bound by a factor of less than 1.5.

Connectivity: When exploring directed graphs (for both cases of just nodes or nodes and edges), usually only strongly connected variants are considered, see for example [1, 14, 15, 25, 30]. This ensures that every node is reachable from the starting node and that the searcher can return to the starting node from every node. If the directed graph is not strongly connected, then any deterministic online algorithm can already get stuck after visiting the first new node, even though an offline algorithm can visit every other node and just skip this one. Similar graphs can be constructed for the randomized case. Consider a directed cycle, where each node has an outgoing edge to the same node v – which has outgoing degree of 0. The starting point is only reached again by the searcher with a probability of $(0.5)^{n-2}$ for $n \geq 3$. Already for $n = 12$ this gives just a chance of < 0.001 to return to the start.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments.

References

1. Albers, S., Henzinger, M.R.: Exploring Unknown Environments. SIAM J. Comput. 29(4), 1164–1188 (2000)
2. Asadpour, A., Goemans, M.X., Madry, A., Gharan, S.O., Saberi, A.: An O(log n/ log log n)-approximation Algorithm for the Asymmetric Traveling Salesman Problem. In: Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '10), pp. 379–389. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2010)

3. Ausiello, G., Bonifaci, V., Laura, L.: The on-line asymmetric traveling salesman problem. J. Discrete Algorithms 6(2), 290–298 (2008)
4. Baldoni, R., Bonnet, F., Milani, A., Raynal, M.: Anonymous graph exploration without collision by mobile robots. Inf. Process. Lett. 109(2), 98–103 (2008)
5. Flocchini, P., Fraigniaud, P., Santoro, N.: Capture of an intruder by mobile agents. In: Proceedings of the fourteenth annual ACM symposium on Parallel algorithms and architectures (SPAA '02), pp. 200-209. ACM, New York, NY, USA (2002)
6. Bender, M.A., Fernandez, A., Ron, D., Sahai, A., Vadhan, S.P.: The Power of a Pebble: Exploring and Mapping Directed Graphs. Inf. Comput. 176(1), 1–21 (2002)
7. Brass, P., Gasparri, A., Cabrera-Mora, F., Xiao, J.: Multi-robot tree and graph exploration. In: Proceedings of the 2009 IEEE international conference on Robotics and Automation (ICRA'09), pp. 495–500. IEEE Press, Piscataway, NJ, USA (2009)
8. Bläser, M.: A new approximation algorithm for the asymmetric TSP with triangle inequality. ACM Transactions on Algorithms 4(4), 47:1–47:15 (2008)
9. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press, Cambridge, England (1998)
10. Bermann, P.: On-line Searching and Navigation. In: Fiat, A., Woeginger, G.J. (eds.) Online Algorithms: The State of the Art. LNCS, vol. 1442., pp. 232–241. Springer, Heidelberg (1998)
11. Burgard, W., Moors, M., Fox, D., Simmons, R.G., Thrun, S.: Collaborative Multi-Robot Exploration. In: Proceedings of the 2000 IEEE International Conference on Robotics and Automation (ICRA '00), pp. 476–481. IEEE, San Francisco, CA, USA (2000)
12. Chalopin, J., Flocchini, P., Mans, B., Santoro, N.: Network Exploration by Silent and Oblivious Robots. In: Proceedings of the 36th international conference on Graph-theoretic concepts in computer science (WG'10), pp. 208–219. Springer-Verlag, Berlin, Heidelberg (2000)
13. Das, S., Flocchini, P., Kutten, S., Nayak, A., Santoro, N.: Map construction of unknown graphs by multiple agents. Theor. Comput. Sci. 385(1-3), 34–48 (2007)
14. Deng, X., Papadimitriou, C.H.: Exploring an unknown graph (Extended Abstract). In: Proceedings of the 31st Annual Symposium on Foundations of Computer Science (FOCS '90), Volume I, pp. 355–361. IEEE Computer Society, St. Louis, Missouri, USA (1990)
15. Deng, X., Papadimitriou, C.H.: Exploring an unknown graph. J. Graph Theory 32(3), 265–297 (1999)
16. Dobrev, S., Královič, R., Markou, E.: Online Graph Exploration with Advice. In: Even, G., Halldórsson, M.M. (eds.) SIROCCO 2012. LNCS, vol. 7355, pp. 267–278. Springer, Heidelberg (2012)
17. Dynia, M., Łopuszański, J., Schindelhauer, C.: Why Robots Need Maps. In: Prencipe, G., Zaks, S. (eds.) SIROCCO 2007. LNCS, vol 4474, pp. 41–40. Springer, Heidelberg (2007)
18. Engebretsen, L.: An Explicit Lower Bound for TSP with Distances One and Two. Algorithmica 35(4), 301–318 (2003)
19. Feige, U., Singh, M.: Improved Approximation Ratios for Traveling Salesperson Tours and Paths in Directed Graphs. In: Charikar, M., Jansen, K., Reingold, O. Rolim, J.D.P. (eds.) APPROX-RANDOM 2007. LNCS, vol. 4627, pp. 104–118. Springer, Heidelberg (2007)
20. Fraigniaud, P., Ilcinkas, D.: Digraphs Exploration with Little Memory. In: Diekert, V., Habib, M. (eds.) STACS 2004. LNCS, vol. 2996, pp. 246–257. Springer, Heidelberg (2004)

21. Fraigniaud, P., Gasieniec, L., Kowalski, D.R., Pelc, A.: Collective tree exploration. Networks 48(3), 166–177 (2006)
22. Frieze, A.M., Galbiati, G., Maffioli, F.: On the worst-case performance of some algorithms for the asymmetric traveling salesman problem. Networks 12(1), 23–39 (1982)
23. Fleischer, R., Kamphans, T., Klein, R., Langetepe, E., Trippen, G.: Competitive Online Approximation of the Optimal Search Ratio. SIAM J. Comput. 38(3), 881–898 (2008)
24. Fleischer, R., Trippen, G.: Experimental Studies of Graph Traversal Algorithms. In: Jansen, K., Margraf, M., Mastrolilli, M., Rolim, J. (eds.) WEA 2003. LNCS, vol. 2647, pp. 120–133. Springer, Heidelberg (2003)
25. Fleischer, R., Trippen, G.: Exploring an Unknown Graph Efficiently. In: Brodal, G., Leonardi, S. (eds.) ESA 2005. LNCS, vol. 3669, pp. 11–22. Springer, Heidelberg (2005)
26. Hurkens, C.A.J., Woeginger, G.J.: On the nearest neighbor rule for the traveling salesman problem. Oper. Res. Lett. 32(1), 1–4 (2004)
27. Kalyanasundaram, B., Pruhs, K.: Constructing Competitive Tours from Local Information. Theor. Comput. Sci. 130(1): 125–138 (1994)
28. Kaplan, H., Lewenstein, M., Shafrir, N., Sviridenko, M.: Approximation Algorithms for Asymmetric TSP by Decomposing Directed Regular Multigraphs. In: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS '03), pp. 56–65. IEEE Computer Society, Washington, DC, USA (2003)
29. Kuhn, F., Oshman, R.: The Complexity of Data Aggregation in Directed Networks. In: Peleg, D. (ed.) DISC 2011. LNCS, vol. 6950, pp. 416–431. Springer, Heidelberg (2011)
30. Kutten, S.: Stepwise construction of an efficient distributed traversing algorithm for general strongly connected directed networks or: Traversing one way streets with no map. In: Computer Communication Technologies for the 90's, Proceedings of the Ninth International Conference on Computer Communication (ICCC '88), pp. 446–452. International Council for Computer Communication Elsevier (1988)
31. Megow, N., Mehlhorn, K., Schweitzer, P.: Online Graph Exploration: New Results on Old and New Algorithms. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011. LNCS, vol. 6756, pp. 478–489. Springer, Heidelberg (2011)
32. Miyazaki, S., Morimoto, N., Okabe, Y.: The Online Graph Exploration Problem on Restricted Graphs. IEICE Transactions 92-D(9), 1620–1627 (2009)
33. Prakash, R.: Unidirectional links prove costly in wireless ad hoc networks. In: Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications (DIALM '99), pp. 15–22. ACM, New York, NY, USA (1999)
34. Ribeiro, B. F., Wang, P., Murai, F., Towsley, D.: Sampling directed graphs with random walks. In: Proceedings of the IEEE INFOCOM 2012, pp. 1692–1700. IEEE, Orlando, FL, USA (2012)
35. Rosenkrantz, D.J., Stearns, R.E., Lewis, P.M.II: An Analysis of Several Heuristics for the Traveling Salesman Problem. SIAM J. Comput. 6(3), 563–581 (1977)
36. Sedgewick, R., Vitter, J. S.: Shortest Paths in Euclidean Graphs. Algorithmica 1(1), 31–48 (1986)
37. Vishwanathan, S.: An Approximation Algorithm for the Asymmetric Travelling Salesman Problem with Distances One and Two. Inf. Process. Lett. 44(6), 297–302 (1992)