Lower and Upper Competitive Bounds for Online Directed Graph Exploration[☆]

Klaus-Tycho Foerster[a,1,*], Roger Wattenhofer[a]

[a] *ETH Zürich, Gloriastrasse 35, 8092 Zurich, Switzerland*

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. Analogously to the travelling salesman problem, the goal is to minimize the cost of such a tour. In this article, we present upper and lower bounds for the competitive ratio of 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. As it turns out, restricting the diameter, the incoming/outgoing degree, or randomly choosing a starting point does not improve lower bounds beyond a small constant factor. Even supplying the searcher in a planar euclidean graph with the nodes' coordinates does not help. Essentially, exploring a directed graph has a multiplicative overhead linear in the number of nodes. Furthermore, if one wants to search for a specific node in unweighted directed graphs, a greedy algorithm with quadratic multiplicative overhead can only be improved by a small constant factor as well.

Keywords: Graph exploration, Online algorithms, Competitive analysis, Directed graphs

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?!

[☆] A preliminary extended abstract appeared in Proceedings of the 16th International Conference on Principles of Distributed Systems (OPODIS), Springer, 2012 [37].

[*] Corresponding author

Email addresses: foklaus@ethz.ch (Klaus-Tycho Foerster), wattenhofer@ethz.ch (Roger Wattenhofer)

[1] *Tel:* +41 44 63 24776

In this article 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, even with coordinates.

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 [49], routing [57], web crawlers [53], or traversing social networks [58]. Brass et al. [13] 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 [21] 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 [17, 33]. Because of all these applications, directed graph exploration will be the main focus in this article. 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

For ease of notation, in the remainder of our paper a graph $G = (V, E)$ is directed and strongly connected with $|V| = n \geq 6$ nodes and $|E| = m$ edges. We denote an edge from $u \in V$ to $v \in V$ with uv. All nodes $v \in V$ have unique IDs and all edges $uv \in E$ have non-negative weights of weight(uv).

A walk is a concatenation of incident edges $uv, vw, \ldots$, with a path being a walk that visits no node twice. The weight or cost of a walk or a path is the sum of the weights of each edge traversal. The distance from a node u to a node v is the weight of a shortest path from u to v, i.e. of a path with the least cost.

Similarly, the hop-cost of a walk or a path is the number of edges traversed ("hops"), with the hop-distance from a node u to a node v being the hop-cost of a path from u to v with the least hop-cost. The radius of a node u is the largest hop-distance to any other node in V, with the diameter of a graph G being the largest radius of any node $u \in V$.

A searcher that explores a graph via some deterministic or randomized algorithm 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. [46, 52]) of the corresponding nodes at the head of these edges. We call

a graph explored, if a searcher starting from some node s has visited all nodes and returned to s.

We only consider the common model of strongly connected directed graphs [1, 20, 21, 33, 50], since a searcher else might get stuck right away (Section 8). In Section 9, we also include geometric coordinates. Graph exploration is an online problem since only partial information about the graph is available [8, 12]. For other exploration models, e.g., unique edge names, or information about incoming edges, we refer to Section 8 as well.

The cost of such an online exploration tour T is measured by the total sum of the weight of the traversed edges, denoted as $\text{cost}(T)$. 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 c_r of a tour T is measured by the cost of the (online) tour divided by the cost of an (offline[2]) tour of minimum cost OPT, i.e. $c_r(T) = \frac{\text{cost}(T)}{\text{cost}(OPT)}$. Should OPT have a cost of 0 while $\text{cost}(T)$ has a positive value, then we set $c_r(T) = \infty$. We say a deterministic algorithm A has a competitive ratio of $r(n)$ w.r.t. some graph class $\mathcal{G}$, if for every graph $G \in \mathcal{G}$ the computed tour T by A has a competitive ratio of at most $r(n)$, i.e., $r(n) \geq c_r(T)$ for all $G \in \mathcal{G}$. Should A induce a competitive ratio of ∞ for some $G \in \mathcal{G}$, then $r(n)$ is set to ∞ as well. If we do not denote a specific graph class $\mathcal{G}$, then we assume $\mathcal{G}$ to be the class of all strongly connected directed graphs.

We also study graph exploration by randomized algorithms[3], where we study the *expected* costs instead of the worst case. Thus, the competitive ratio of a tour is now calculated by $\mathbb{E}\left[\frac{\text{cost}(T)}{\text{cost}(OPT)}\right]$. Let us consider a small introductory example, where the cost of an optimal tour is 10: If the randomized algorithm chooses a tour with a cost of 80 with a 25% chance and a tour with a cost of 40 with a 75% chance, then the competitive ratio is $(20+30)/10 = 5$. The competitive ratio of a randomized algorithm is analogously $r(n)$ if $r(n) \geq \mathbb{E}\left[\frac{\text{cost}(T)}{\text{cost}(OPT)}\right]$ for all $G \in \mathcal{G}$.

1.2. Results

In this article 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 general weighted and unweighted cases. 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

[2] Offline in the sense that all information about the graph is available to the searcher.

[3] We note that every deterministic algorithm can also be used as a randomized algorithm. Furthermore, while an algorithm designer could choose to, e.g., pick new edges uniformly at random for exploration, any other strategy is possible as well, e.g., pick the next node to explore with a probability inversely proportional to the cost of reaching that node.

type of graph \ competitivity	lower bound	upper bound	mult. gap	Section
(deterministic) general*^{c}	$n-1$	$n-1$	sharp	3, 4
(randomized) general*$^{+c}$	$\frac{n}{4}$	$n-1$	≤ 4	3, 4
(d.) unweighted general*	$\frac{n}{2}+\frac{1}{2}-\frac{1}{n}$	$\frac{n}{2}+\frac{1}{2}-\frac{1}{n}$	sharp	5
(r.) unweighted general*	$\frac{n}{8}+\frac{3}{4}-\frac{1}{n}$	$\frac{n}{2}+\frac{1}{2}-\frac{1}{n}$	≤ 4	5
(d.) euclidean planar	$n-2-\epsilon'$	$n-1$	$1+O(1/n)$	7, 4
(r.) euclidean planar	$\frac{n}{4}-\epsilon'$	$n-1$	$\leq 4+\epsilon$	7, 4
(d.) unit w. euclidean planar	$\frac{n}{4}+\frac{1}{2}-\frac{2}{n}$	$\frac{n}{2}+\frac{1}{2}-\frac{1}{n}$	≤ 2	7, 5
(r.) unit w. euclidean planar	$\frac{n}{8}+\frac{3}{4}-\frac{1}{n}$	$\frac{n}{2}+\frac{1}{2}-\frac{1}{n}$	≤ 4	7, 5
(r.) geom. euclidean planar	$\frac{n}{16}+\frac{5}{8}+\frac{1}{2n}-\epsilon'$	$n-1$	$\leq 16+\epsilon$	9
(d.) searching a node	$\frac{(n-1)^2}{4}-\frac{(n-1)}{4}-\frac{1}{2}$	$\frac{n^2}{4}-\frac{n}{4}$	≤ 3	6
(r.) searching a node	$\frac{n^2}{16}-\frac{n}{8}+1$	$\frac{n^2}{4}-\frac{n}{4}$	< 4.1	6

* also applies to planar graphs and graphs that satisfy the triangle inequality[4]

ϵ, ϵ' denote any fixed value greater than 0

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

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

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.

has a multiplicative overhead of $\Theta(n)$. Perhaps surprisingly, even allowing to see the nodes' coordinates in planar euclidean graphs will not help a searcher beyond a constant factor.

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.

Before our work, sharp results regarding deterministic and randomized exploration of directed graphs have not yet been published. We summarize our main results in Table 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 [42], the constant was improved over time, e.g. [9, 47]; the best

[4]The triangle inequality states that for every $v, u, w \in V$ holds: If $vu, uw, vw \in E$, then weight(vu) + weight$(uw) \geq$ weight(vw).

result known to us is $\frac{2}{3}\log_2 n$ [28]. 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 [63], with a NP-hard lower bound of 2805/2804 $-\epsilon$ [27]. 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 $\Theta(\log n)$ [59], it is not known if a constant competitive ratio for general graphs is possible [52]. For cycles there is an algorithm with a sharp competitive ratio of $\frac{1+\sqrt{3}}{2}$, while for trees and unweighted graphs depth-first search is optimal [54]. Recently, the best known lower bound for general graphs was improved from $2-\epsilon$ [54] to $5/2-\epsilon$ [23]. For planar graphs a sophisticated variant of depth-first search named ShortCut by Kalyanasundaram and Pruhs achieves a competitive ratio of 16 [46]. Their result was recently extended for graphs of genus g to $16(1+2g)$ [52]. If there are just k different edge weights, there exists an algorithm with competitive ratio $2k$ [52]. Fleischer et al. considered the problem of searching just for a node instead of a tour in [31]. 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 [50]: 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 walk 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 [20], 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 [21]. An upper bound only dependent polynomially in d for the directed case was given by Fleischer and Trippen [33], 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 [20, 21].

There seems to be no known randomized algorithm for the exploration of graphs (whether 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 [32].

Graph exploration has also been considered with restricted memory models or multiple searchers, see for example [4, 7, 14, 18, 19, 22, 24, 38, 39]. In the context of biologically inspired algorithms, Feinerman et al. [29, 30] proposed to let agents without communication search the plane to find a treasure, where the

time needed to find the treasure is roughly quadratic in its distance divided by the number of agents. This idea has also been considered under the assumption that the ants are modeled as finite state machines by Emek et al. [25, 26]: Comparable bounds (to turing-machine agents) can be reached when considering even a very small constant numbers of ants, but communication is necessary in this case. Work by Fraigniaud et al. [40] showed that if one uses only a single searcher, then it needs memory in the order of diameter times maximum degree to succeed.

Historically, the oldest known online exploration/navigation problem studied formally on graphs is graph searching, which was first discussed by Breisch [15] and Parson [55, 56] (cf. [6, 16]). It stands for a closely related problem: A number of agents has to capture an intruder, or as formulated in the original paper [15], a rescue party has to search for a person wandering aimlessly in a particular cave. The offline problem of determining the minimum number of searchers is NP-complete, but can be solved in linear time for trees [51]. For a bibliography on graph searching see [36], with more recent work in, e.g., [5, 10, 11, 34, 35, 62].

For an overview of other online navigation tasks we refer to [8, 43, 44].

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 analogous results can be achieved by replacing 0 with 1 and 1 with $\lceil 1/\epsilon \rceil$ for arbitrarily small $\epsilon > 0$, cf. Subsection 8.4. 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. □

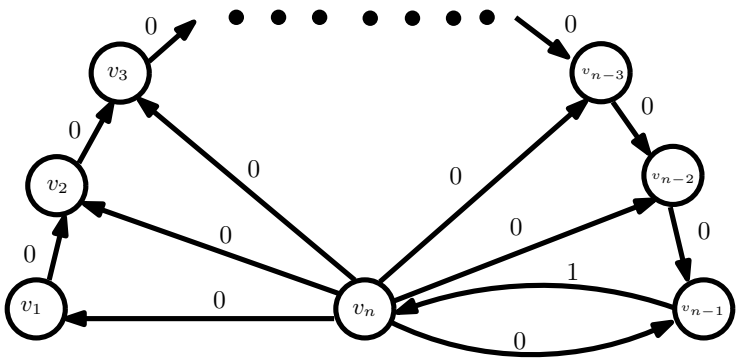

Figure 1: In this graph the starting node s is v_n in the lower middle of the image. A deterministic algorithm can get tricked into first visiting v_{n-1}, then v_{n-2} and so on.

3.2. Randomized Online Algorithms

The construction in Figure 1 relied on a worst case analysis for deterministic algorithms. When looking at the expected exploration costs of a well-designed randomized algorithm, the situation changes: The searcher starting at v_n can pick each of the yet unvisited nodes with the same probability, meaning it will on average choose a node in the "middle" of the so far yet unvisited nodes, therefore visiting the starting node only about $O(\ln(n))$-times in expectation. 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. 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 $\frac{n}{4}$. □

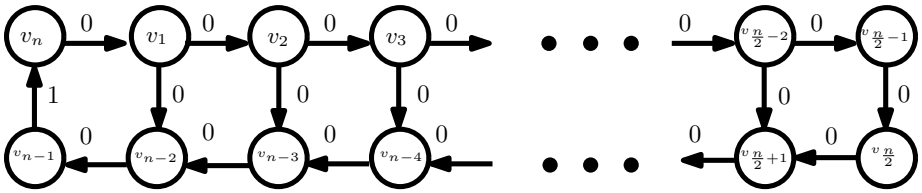

Figure 2: In this graph the starting node s is v_n in the upper left corner. Upon arriving at each of the nodes $v_1, v_2, \ldots v_{\frac{n}{2}-2}$ for the first time, a randomized algorithm gets tricked into taking the wrong edge with probability 0.5. If n is odd, then the lower right node $v_{\frac{n}{2}}$ 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' will be explored with a cost of at least 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 $n/16$. □

This technique can also be applied to the geometric search with coordinates and the problem of searching for a node which are both covered later.

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 $O(\log n)$ [59]. A greedy approach reaches this

competitive ratio of $O(\log n)$ [59], but the same algorithm has a competitivity of $\Omega(\log n)$ even on planar unweighted graphs [45].

We define the greedy approach (or also, greedy algorithm) analogously to the nearest neighbor algorithm in [59] for the case of directed graphs:

Greedy algorithm (for directed graph exploration)
Let V_{known} be set the set of known, but not yet visited nodes. Initially this is just the set of neighbors of s. Select a node $v \in V_{known}$ that can be reached with a path P of the least cost[5] *in the so far known subgraph and traverse the path P until v is reached. Repeat the last step until $V_{known} = \emptyset$. Then, choose a path P of least cost to s and traverse P to s.*

Thanks to our strong lower bounds, this 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 $1 \leq r \leq n$.

The greedy algorithm proceeds as follows: Upon reaching a node v_k^g for the first time, it selects 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. We are now going to show by contradiction that the path w_{k+1}^g has at most the combined cost of the concatenated paths from v_k^g to v_{k+1}^g in OPT, even though the greedy algorithm has no knowledge of OPT.[6] Let us assume it has greater cost: 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:

[5] Note that this does not have to be a neighbor of the current node!

[6] Note that v_k^g and v_{k+1}^g could appear in a totally different order in OPT, e.g., v_{k+1}^g could be the third new node v_3^o to be visited and v_k^g could be v_{n-1}^o.

Corollary 5. *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.*

5. Unweighted Graphs

An unweighted graph is a graph where the edges have no 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 6. *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 $2 + 3 + \cdots + n = \frac{n^2}{2} + \frac{n}{2} - 1$. 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 7. *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 $1+2+3+\cdots+(n-2)+(n-1)+(n-1) = -1+\sum_{i=1}^{n} i = \frac{n^2}{2}+\frac{n}{2}-1$. 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 6 and Theorem 7 yields:

Corollary 8. *The result of Theorem 7 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 9. *Searching for a node in strongly connected directed weighted graphs has a competitive ratio of ∞ 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 a competitive ratio of ∞, 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 10. *No online algorithm for searching a node in strongly connected directed unweighted graphs can achieve a better competitive ratio than $\frac{(n-1)^2}{4}-\frac{(n-1)}{4}-\frac{1}{2}$ (deterministic) or $\frac{n^2}{16}-\frac{n}{8}+1$ (randomized).*

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 6) $\frac{n^2}{2}+\frac{n}{2}-1-n$. 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 consider a star with one center starting node s, with an outgoing edge to each of the k leaves, and one edge going back from

each of the k leaves to s. If we now hide the searched node v behind one of the leaves v_i, then the optimal offline solution would have a cost of two (s to v_i to v). On the other hand, a randomized algorithm needs to visit $\frac{k+1}{2}$ different leaves in expectation until the node v can be found, leading to an expected cost of $k+1$. By setting $k = n-2$, this construction leads to a lower bound of $\frac{n-1}{2}$ for the competitive ratio of any randomized algorithm.

However, we can improve this construction: Instead of letting the searcher go back directly from each of the leaves (which induces a total cost of two for visiting each leaf and going back), we remove the back-edge from each leaf and connect each of these leaves to a new node v'. An exception is the leaf connected to v: There only the outgoing edge from v changes to point at v'. We refer to Figure 3 for an illustration.

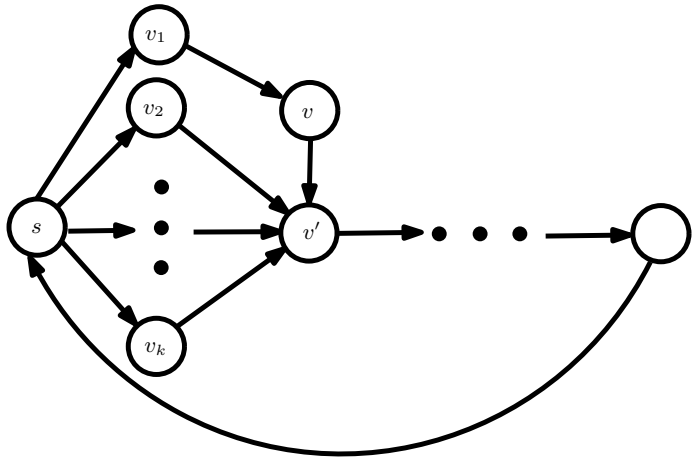

Figure 3: When the searcher starts at node s and searches for the node v, there are k different nodes adjacent to s to choose from. However, not choosing v_1 will induce a large cost, as the only path back to s from v_i, with $2 \leq i \leq k$, is along $k' = n-k-2$ further nodes.

From the node v', the only way back to s from v' is via a path of length k' along $k'-1$ further nodes. Now, the optimal path to v still has a cost of two, but going to one of the k leaves and back to s now induces a cost of $k'+2$. I.e., the expected cost for any randomized online algorithm will be at least $\frac{(k'+2)(k-1)}{2}+2$, leading to a lower bound on the competitive ratio of $\frac{(k'+2)(k-1)}{4}+1$. If we set $k' = \frac{n}{2}-2$ and $k = \frac{n}{2}$, we have (with the starting node s and the searched node v) n nodes in total, leading to a lower bound of $\frac{\left(\frac{n}{2}\right)\left(\frac{n}{2}-1\right)}{4}+1 = \frac{n^2}{16}-\frac{n}{8}+1$. □

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

Theorem 11. *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 7). 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 10 and Theorem 11 yields the following corollary:

Corollary 12. *Searching for a node in strongly connected directed graphs has a competitive ratio of $\Theta(n^2)$, for both deterministic and randomized algorithms.*

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 idea can be applied for the randomized case with the construction from the proof of Theorem 10.

7. Lower Bounds for Special Graph Classes

When we add directed edges with an arbitrarily high weight W to a given graph, then using these edges in any online algorithm is not beneficial, as already traversing them once sets the cost of the obtained tour to be at least W. However, 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.

Theorem 13. *For graphs of any diameter from 1 to $n-1$, no online algorithm can achieve a better competitive ratio on exploring all nodes of strongly connected directed weighted graphs than $n-1$ (deterministic) or $n/4$ (randomized).*

Proof. We start with the deterministic case: Consider the graph from Figure 1. It has a diameter of $n-1$, as the nodes v_1, v_2 have a radius of $n-1$, the node v_3 has $n-2$, the node v_4 has $n-3$, …, v_{n-1} has 2, and lastly, v_n has a radius of 1. If we add edges with arbitrarily large weight from v_1 to v_3 and from v_2 to v_4, then only the radius of v_1, v_2 changes from $n-1$ to $n-2$, i.e., the diameter is now $n-2$.

To achieve a diameter of $n-3$, the nodes v_1, v_2, v_3 need to have a radius of $n-3$. Adding an edge from v_3 to v_5 reduces the radius of v_3 to $n-3$. Similarly, adding edges from v_2 to v_5 and from v_1 to v_4 reduces their radius to $n-3$ respectively, resulting in a diameter of $n-3$.

The general idea for adding these edges is that, except for v_1, the radius of each node is determined by its edge distance to v_n plus one, while for v_1 it is the edge distance to v_n. Thus, to achieve a diameter of $n-1 \geq i \geq 2$, one would reduce the radius of the nodes $v_1, v_2, \ldots, v_{n-i+1}$ to i. In all cases, choosing any of the new edges with arbitrarily large weight will not improve the result and the searcher gains no new additional information about the nodes v_1 to v_{n-1} when being at v_n, unless the nodes were visited beforehand.

For a diameter of 1, we add all possible missing edges with arbitrarily large weight to make the graph a complete graph. Again, the searcher cannot use any of the new edges without violating the competitive ratio. When being at a node v_i with $1 \leq i \leq n-1$, the searcher now sees outgoing edges to all nodes v_j with $j < i$, but this information is of no use to the searcher.

We now look at the randomized case. Similar to above, we consider the graph from Figure 2, but to avoid case analysis we consider just graphs where n is divisible by 12. The graph has a diameter of $n-1$ as well: The edge distance from $v_{\frac{n}{2}+1}$ to $v_{\frac{n}{2}}$ is $n-1$, also from $v_{\frac{n}{2}}$ to $v_{\frac{n}{2}-1}$ and $v_{\frac{n}{2}-1}$ to $v_{\frac{n}{2}-2}$, as the shortest path each time is via the starting node v_n. However, there is no node with a radius of just 1, the nodes with the smallest radius of $1 + n/3$ are the two nodes $v_{-1+2n/12}$ and $v_{2n/12}$. For the $2n/3$ nodes from $v_{\frac{n}{2}+1}$ to $v_{-1+2n/12}$, the radius decreases by 1 from $n-1$ to $1+n/3$, while for the remaining nodes, the radius increases from $1+n/3$ to $n-1$ by 2 the further one is away from the starting node v_n.

The used construction scheme is a bit different to the deterministic case, we depict both main ideas in Figure 4:

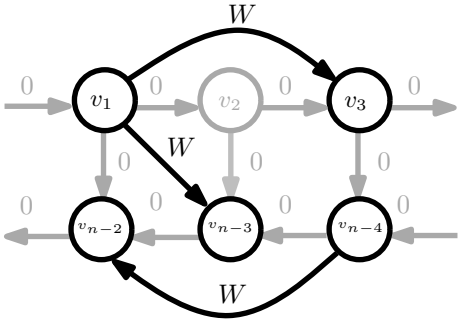

Figure 4: An example of the two techniques how to reduce the diameter of the graph in the randomized case by adding edges with arbitrarily high weight W. The new lower edge from v_{n-4} to v_{n-2} decreases the diameter by one and the two new outgoing edges from v_1 decrease the diameter by one as well. In the latter case, two edges have to be introduced to not allow the searcher to break symmetry when deciding whether to continue at v_3 or v_{n-3}.

When an edge is introduced from v_1 to v_4, then an edge from v_1 to v_{n-4} has to be introduced as well, as else the searcher could break the symmetry between v_4 and v_{n-4} when visiting the node v_3 for the first time. However, an edge with arbitrarily large weight from one of the nodes $v_{n/2-1}, \ldots, v_{n/2-1+i}$ with $0 \leq i \leq n/2$ to any of the nodes in the range $v_{n/2+i}, \ldots, v_n$ does not help

the searcher, because when this edge is seen for the first time, the searcher just needs to return to v_n and is done with the exploration. Thus, it is possible to reduce the radius of all the nodes $v_{n/2-1}, \ldots, v_n$ to any value of at most $n/2$. In a similar way, adding an edge with arbitrarily large weight from any node to the node $v_{n/2-1}$ does not help the searcher with exploring the graph. The same holds for adding edges with arbitrarily large weights from $v_{n/2-1}$ to any other node. Combining these techniques allows to reduce the diameter from $n-1$ to any value down to 2, at which point all nodes would have an edge to $v_{n/2-1}$ and $v_{n/2-1}$ would have an edge to all other nodes.

To reach a diameter of 1, one would turn the graph into a complete graph by adding all possible missing edges, again with an arbitrarily large weight: Now, if there would be, e.g., an edge from v_1 to v_4, there is also an edge from v_1 to v_{n-4}, as the graph is complete. Like before, the searcher cannot use any of these new edges without violating the competitive ratio. □

Corollary 14. *For graphs of any maximum (respectively minimum) incoming/outgoing degree from 2 (respectively 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, cf. [60].

Theorem 15. *No online algorithm can achieve a better competitive ratio on exploring all nodes of strongly connected directed 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 and replace all edge weights with a fixed ϵ^r, with $1/100 > \epsilon^r > 0$ and $\epsilon^*/n^2 > \epsilon^r$, and embed it as a planar euclidean graph like shown in the figure, except for the node v_{n-1} – which will be placed next. 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 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. We place v_{n-1} in the center of the circle and obtain a proper planar euclidean embedding of the constructed graph. As we chose ϵ^r to be small enough, we reach a lower bound of $n/4-\epsilon^*$.

For the deterministic case we consider the graph in Figure 1 and proceed in a similar fashion. Let us fix a ϵ^d with $1/100 > \epsilon^d > 0$, $\epsilon^*/n^2 > \epsilon^d$ and construct a circle of radius ϵ^d with v_n being in the center of the circle and placing the nodes v_1 to v_{n-2} 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. As we chose ϵ^d to be small enough, 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 16. *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).*

8. Other Exploration Models

In the following subsections we discuss five variations of the previously used exploration model: Unique edge names in Subsection 8.1, seeing identifiers from incoming edges in Subsection 8.2, deviating from strongly connected directed graphs in Subsection 8.3, positive integer edge weights in Subsection 8.4, and giving advice in Subsection 8.5. For the special case of embedding a planar graph into a plane and letting the searcher see coordinates, we refer to Section 9.

8.1. 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.

8.2. 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.

8.3. 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, 20, 21, 33, 50]. 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 exploration is not feasible (for example if two nodes have outgoing degree of 0).

8.4. Positive Integer Edge Weights

In Section 5 we studied the exploration of unweighted graphs, opposed to weighted graphs before – where we just used the edge weights 0 and 1 for the lower bounds in Section 3. One could also change the model to just allow positive integer edge weights ranging from 1 to some $W \in \mathbb{N}_{>0}$, to prevent *i*) any edge from having a weight of 0, and *ii*) to introduce a limit of W on how much more costly traversing one edge can be compared to another.

We start with the deterministic case. Consider the graph from Figure 1, but replace all edge weights of 0 with 1 and assign the single edge back to the start v_n with weight 1 a weight of W. The optimal tour will use the edge with weight W just once, while a deterministic exploration algorithm will use it $n-1$ times in the worst case. An optimal tour has now a cost of $n - 1 + W - 1$, and any deterministic algorithm can incur a cost of $\frac{n^2}{2} + \frac{n}{2} - 1 + (n-1)(W-1)$.

For the randomized case, we consider the graph from Figure 3.2, and proceed in an analogous fashion: I.e., we replace all edge weights of 0 with 1 and assign the edge back to the start v_n with weight 1 a weight of W. The optimal tour has a cost of $n + W - 2$ as well, and the expected costs for a randomized algorithm are at least $\frac{n^2}{8} + \frac{3n}{4} - 1 + \frac{n}{4}(W-1)$.

Thus, if we consider the asymptotic behavior of our lower bounds, they converge to the undirected case for any fixed W with $n \to \infty$ and towards the weighted case for any fixed n with $W \to \infty$.

8.5. Advice Complexity

The authors of [23] studied the problem of advice complexity for exploring undirected graphs[7]. They showed that there is a family of graphs where a

[7]Tree exploration was studied in [41] and searching for a node was studied in [48, 61].

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 $O(n \log(n))$ bits: We 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 can be uniquely represented in size $O(\log(n))$ bits, the lower bound of $\Omega(n \log(n))$ from [23] is a sharp bound of $\Theta(n \ln(n))$ bits for both directed and undirected graphs.

9. Adding Geometry

In the previous sections, the searcher could easily be fooled by multiple options, out of which only one in some sense was correct. In Figure 1, the searcher had to guess the right node out of $n-1$ nodes when being at the starting node, while in Figure 2, the searcher had to choose the right node out of two options a linear number of times. As all options were indistinguishable except for the nodes' identifiers, it was essentially a guessing game. Various changes of the model have been discussed in this article up to this point, though no non-artifical ones offered a way out of a multiplicative linear overhead. One model change that could offer a way out is to embed the nodes of a planar graph into the euclidean plane, set the edge weights to the euclidean distance between the respective nodes and to supply the searcher with the coordinates of all nodes at outgoing incident edges. Then, in the counterexamples of Figure 1 and Figure 2, the searcher could easily achieve better competitive ratios. Nonetheless, we will show in this section that even such a geometric model has a linear multiplicative overhead.

Thus, we will increase the power of the searcher in this section compared to the model defined in Subsection 1.1 (changes in bold): A searcher that explores an **euclidean** graph via some deterministic or randomized algorithm 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 to the searcher: all outgoing incident edges including their weight, plus the IDs **and the coordinates** of the corresponding nodes at the head of these edges.

We indicate this more powerful searcher by denoting that the searcher has access to the coordinates of each known node.

Theorem 17. *No randomized online algorithm can achieve a better competitive ratio on exploring all nodes of strongly connected directed planar euclidean graphs than* $\frac{n}{16} + \frac{5}{8} + \frac{1}{2n} - \epsilon \in \Omega(n)$ *for any* $\epsilon > 0$*, even if the searcher has access to the coordinates of each known node.*

The proof idea is essentially as follows: Even with geometric information, the searcher can still only guess in some cases what the right way to go would be. Following the idea of the proofs of Theorem 2 and Theorem 6, we can construct a family of graphs where the searcher has to go back to the start a linear number of times in expectation, while an optimal tour only revisits nodes

a constant number of times, resulting in a linear multiplicative overhead for any randomized or deterministic algorithm.

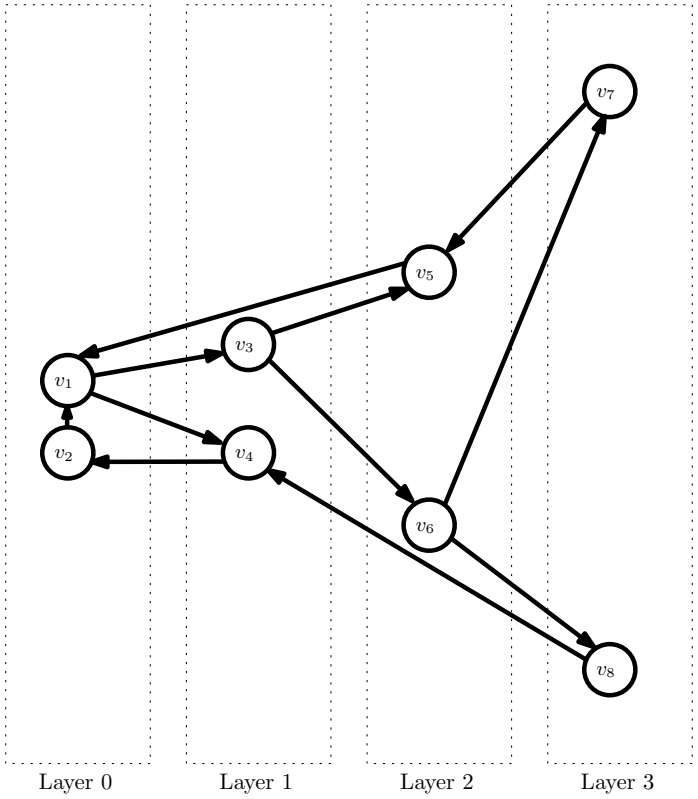

Figure 5: In this graph the starting node s is v_1 on the left side of the image. When the searcher has never visited layer 1 before, it is unknown if v_3 or v_4 advances to layer 2. Even with the knowledge of the coordinates of v_3 or v_4, both nodes could be the correct ones to choose. While the correct node in layer 1 in this example is the top node v_3, it might as well have been the bottom node (as it is the case in layer 2). Each time the searcher chooses the wrong node, it has to go back to the start. An all-knowing algorithm however only needs to go back to v_1 once before finishing the tour, by traversing the graph in the sequence $v_1, v_3, v_6, v_7, v_5, v_1, v_3, v_6, v_8, v_4, v_2, v_1$.

Proof. We construct a family of planar euclidean graphs with an even amount of $n \geq 6$ nodes $v_1, v_2, \ldots, v_n$, which consists exactly of all graphs which fit the following description: The nodes are organized in $n/2$ layers, with the nodes v_1, v_2 being layer 0, the nodes v_3, v_4 being layer 1, ..., and the nodes v_{n-1}, v_n being layer $n/2 - 1$.

For the remainder of the proof, we will tag each node being either a "correct" or a "wrong" node, with the idea that the correct nodes in the intermediate layers are those that need to be visited to advance to the next layer: For each layer from 1 to $n/2 - 2$, one node is the correct node and the other node is the

wrong node. In these $n/2-2$ layers, the correct node has two outgoing edges to both nodes of the next layer, with the node v_1 having two outgoing edges to the nodes v_3, v_4 as well. For the sake of completeness of tagging of all nodes being correct or wrong, the first layer 0 has two wrong nodes and the last layer $n/2-1$ has also just two wrong nodes. Furthermore, all nodes with odd index will be denoted as top nodes and all nodes with even index as bottom nodes. In each layer i from 1 to $n/2-1$, each wrong node can be top or bottom. Note that the layers from 1 to $n/2-2$ each only have one wrong node. If it is a top node, it has an outgoing edge to the top wrong node with the largest layer smaller than i. The same holds analogously if the wrong node is a bottom node. Lastly, the node v_2 has one outgoing edge to v_1. Thus, the layers 0 to $n/2-2$ have each three outgoing edges and the layer $n/2-1$ has exactly two outgoing edges, resulting in $m = 3(n/2-1)+2 = 1.5n-1$ edges. The starting node for the searcher is set as the node v_1.

The only thing needed to complete the description of each graph is the actual embedding, i.e., the two-dimensional coordinates of each node. Let $0 < \epsilon' < 1$. Each top node in layer i has the coordinates $(i, i^2 \cdot \epsilon')$, while each bottom node in layer $i > 0$ has the coordinates $(i, -i^2 \cdot \epsilon')$. The missing bottom node v_2 in layer 0 has the coordinates $(0, -\epsilon')$. We note that with to the choice of the y-coordinates (with $i^2 \cdot \epsilon'$), the graph is planar. An example with $n = 8$ nodes can be found in Figure 5.

We first describe an optimal tour[8]: The searcher starts by advancing from layer to layer via the correct nodes, then picking the top node in the last layer, and visits all wrong top nodes on the way back to the start. This is repeated once more, but on the way back, all wrong bottom nodes are visited. The cost of each edge from a layer $i-1$ to a layer i (or from a layer i to a layer $i-1$) can be upper bounded by $\sqrt{1^2 + (2i^2 \cdot \epsilon')^2} = \sqrt{1 + 4i^4 \cdot \epsilon'^2} < 1 + 4i^4 \cdot \epsilon' < 1 + 4(n/2)^4 \cdot \epsilon' = 1 + n^4/4 \cdot \epsilon'$. I.e., the unique path from v_1 to v_{n-1} or v_n has a cost of at most $n/2 + n/2 \cdot n^4/4 \cdot \epsilon'$. Due to the triangle inequality and the upper bound above, the cost of the unique path from layer $n/2-1$ to layer 0 along the top or bottom wrong nodes can be upper bounded by the cost of $n/2 + n/2 \cdot n^4/4 \cdot \epsilon'$ as well. If we add the cost of going from v_2 to v_1, this results in a total cost of $\epsilon' + 4 \cdot \left(n/2 + n/2 \cdot n^4/4 \cdot \epsilon'\right) = 2n + \epsilon' \cdot \left(2n \cdot n^4/4\right) + \epsilon' = 2n + \epsilon' \left(1 + n^5/2\right)$.

We now consider the behavior of any randomized online algorithm starting from v_1. Each time the algorithm wants to move from layer i to layer $i+1$ for the first time, it has to decide with some probability if the top or bottom node is chosen. Note that in the example of Figure 2, the searcher could always choose the top node and explore the graph in an optimal fashion. However, for each strategy chosen and for each $n \geq 6$, there is at least one graph where the searcher will pick the wrong node with a probability of at least 0.5. For ease of readability, we will decrease the weight of edges in the following calculations. This will only improve the cost of an algorithm. Note that the actual strategy

[8]We note that any actual tour of all nodes can be used as an upper bound for an optimal tour that visits all nodes and returns to the start.

stays untouched. Each edge from a layer i to a layer $i+1$ will have a weight of exactly one, while the unique tour from a wrong node in layer i to layer 0 will have a cost of i. The edge from v_2 to v_1 will have a cost of 0.

We can now reason analogously as in the proofs of Theorem 2 and Theorem 6: The cost for choosing the wrong edge in layer 0 is 2, for layer 1 is 4, for layer 2 is 6, ..., for layer $n/2-3$ is $n-4$. Thus, before the searcher starts from v_1 and knows a correct way to the layer $n/2-1$ for the first time, an expected cost of $0.5\,(2+4+6+\cdots+n-4) = 0.5\left((n/2-2)^2+(n/2-2)\right) = n^2/8-3n/4+1$ is accrued. Afterwards, the graph can be explored with a cost of 2n, resulting in a total expected cost of at least $n^2/8+5n/4+1$ for any randomized or deterministic online algorithm.

Recall that the cost of an optimal tour can be bounded from above by $2n+\epsilon'\left(1+n^5/2\right)$. Thus, for every $\epsilon''>0$ and every $n\geq 6$ one can choose $\epsilon'>0$ small enough s.t. no randomized algorithm can achieve a better competitive ratio than $\frac{n}{16}+\frac{5}{8}+\frac{1}{2n}-\epsilon''\in\Omega(n)$. □

Corollary 18. *Exploring all nodes of strongly connected directed planar euclidean graphs has a competitive ratio of $\Theta(n)$, even when considering randomized algorithms where the searcher has access to the coordinates of each known node.*

Lastly, we note that the above corollary holds even when supplying the searcher with the coordinate of every node in the graph before starting the exploration, as long as the searcher is not informed about the edges beforehand.

10. Concluding Remarks

We studied the online exploration of all nodes in directed graphs by a single searcher, both deterministic and randomized. As it turns out, this problem deviates strongly from the corresponding problem in undirected graphs. Apart from rather artificial scenarios, it seems to us that there is no way to escape a multiplicative linear overhead in the competitive ratio, even if coordinates of the nodes are available to the searcher.

Acknowledgements

We would like to thank the anonymous reviewers of the 16th International Conference on Principles of Distributed Systems (OPODIS) for their helpful comments on our preliminary extended abstract [37]. The first author would also like to thank Cristina Pérez Arranz for supplying him with a copy of [15]. Furthermore, we would like to thank the anonymous reviewers of this journal article for their helpful comments as well, among many other things they allowed us to improve the lower bound of the competitive ratio for searching a node with randomized algorithms from $\Omega(n)$ to $\Omega(n^2)$.

11. References

[1] Susanne Albers and Monika Rauch Henzinger. Exploring unknown environments. *SIAM J. Comput.*, 29(4):1164–1188, 2000.

[2] Arash Asadpour, Michel X. Goemans, Aleksander Madry, Shayan Oveis Gharan, and Amin Saberi. An o(log n/ log log n)-approximation algorithm for the asymmetric traveling salesman problem. In Moses Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 379–389. SIAM, 2010.

[3] Giorgio Ausiello, Vincenzo Bonifaci, and Luigi Laura. The on-line asymmetric traveling salesman problem. *J. Discrete Algorithms*, 6(2):290–298, 2008.

[4] Roberto Baldoni, François Bonnet, Alessia Milani, and Michel Raynal. Anonymous graph exploration without collision by mobile robots. *Inf. Process. Lett.*, 109(2):98–103, 2008.

[5] Lali Barrière, Paola Flocchini, Fedor V. Fomin, Pierre Fraigniaud, Nicolas Nisse, Nicola Santoro, and Dimitrios M. Thilikos. Connected graph searching. *Inf. Comput.*, 219:1–16, 2012.

[6] Lali Barrière, Paola Flocchini, Pierre Fraigniaud, and Nicola Santoro. Capture of an intruder by mobile agents. In *Proceedings of the Fourteenth Annual ACM Symposium on Parallel Algorithms and Architectures, SPAA 2002, Winnipeg, Manitoba, Canada, August 11-13, 2002*, pages 200–209. ACM, 2002.

[7] Michael A. Bender, Antonio Fernández, Dana Ron, Amit Sahai, and Salil P. Vadhan. The power of a pebble: Exploring and mapping directed graphs. *Inf. Comput.*, 176(1):1–21, 2002.

[8] Piotr Berman. On-line searching and navigation. In Amos Fiat and Gerhard J. Woeginger, editors, *Online Algorithms, The State of the Art*, volume 1442 of *Lecture Notes in Computer Science*, pages 232–241. Springer, 1996.

[9] Markus Bläser. A new approximation algorithm for the asymmetric TSP with triangle inequality. *ACM Transactions on Algorithms*, 4(4):47:1–47:15, 2008.

[10] Lélia Blin, Janna Burman, and Nicolas Nisse. Exclusive graph searching. In Hans L. Bodlaender and Giuseppe F. Italiano, editors, *Algorithms - ESA 2013 - 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings*, volume 8125 of *Lecture Notes in Computer Science*, pages 181–192. Springer, 2013.

[11] Anthony Bonato and Richard J. Nowakowski. *The game of cops and robbers on graphs*, volume 61 of *Student Mathematical Library*. American Mathematical Society, Providence, RI, 2011.

[12] Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 1998.

[13] Peter Brass, Flavio Cabrera-Mora, Andrea Gasparri, and Jizhong Xiao. Multirobot tree and graph exploration. *IEEE Transactions on Robotics*, 27(4):707–717, 2011.

[14] Peter Brass, Ivo Vigan, and Ning Xu. Improved analysis of a multirobot graph exploration strategy. In *13th International Conference on Control Automation Robotics & Vision, ICARCV 2014, Singapore, December 10-12, 2014*, pages 1906–1910. IEEE, 2014.

[15] Richard Breisch. An intuitive approach to speleotopology. *Southwestern cavers*, 6(5):72–78, 1967.

[16] Richard Breisch. *Lost in a Cave: applying graph theory to cave exploration*. National Speleological Society, 2012.

[17] Wolfram Burgard, Mark Moors, Dieter Fox, Reid G. Simmons, and Sebastian Thrun. Collaborative multi-robot exploration. In *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 2000, April 24-28, 2000, San Francisco, CA, USA*, pages 476–481. IEEE, 2000.

[18] Jérémie Chalopin, Paola Flocchini, Bernard Mans, and Nicola Santoro. Network exploration by silent and oblivious robots. In Dimitrios M. Thilikos, editor, *Graph Theoretic Concepts in Computer Science - 36th International Workshop, WG 2010, Zarós, Crete, Greece, June 28-30, 2010 Revised Papers*, volume 6410 of *Lecture Notes in Computer Science*, pages 208–219, 2010.

[19] Shantanu Das, Paola Flocchini, Shay Kutten, Amiya Nayak, and Nicola Santoro. Map construction of unknown graphs by multiple agents. *Theor. Comput. Sci.*, 385(1-3):34–48, 2007.

[20] Xiaotie Deng and Christos H. Papadimitriou. Exploring an unknown graph (extended abstract). In *31st Annual Symposium on Foundations of Computer Science, STOC 1990, St. Louis, Missouri, USA, October 22-24, 1990, Volume I*, pages 355–361. IEEE Computer Society, 1990.

[21] Xiaotie Deng and Christos H. Papadimitriou. Exploring an unknown graph. *Journal of Graph Theory*, 32(3):265–297, 1999.

[22] Krzysztof Diks, Pierre Fraigniaud, Evangelos Kranakis, and Andrzej Pelc. Tree exploration with little memory. *J. Algorithms*, 51(1):38–63, 2004.

[23] Stefan Dobrev, Rastislav Královic, and Euripides Markou. Online graph exploration with advice. In Guy Even and Magnús M. Halldórsson, editors, *Structural Information and Communication Complexity - 19th International Colloquium, SIROCCO 2012, Reykjavik, Iceland, June 30-July 2, 2012, Revised Selected Papers*, volume 7355 of *Lecture Notes in Computer Science*, pages 267–278. Springer, 2012.

[24] Miroslaw Dynia, Jakub Lopuszanski, and Christian Schindelhauer. Why robots need maps. In Giuseppe Prencipe and Shmuel Zaks, editors, *Structural Information and Communication Complexity, 14th International Colloquium, SIROCCO 2007, Castiglioncello, Italy, June 5-8, 2007, Proceedings*, volume 4474 of *Lecture Notes in Computer Science*, pages 41–50. Springer, 2007.

[25] Yuval Emek, Tobias Langner, David Stolz, Jara Uitto, and Roger Wattenhofer. How many ants does it take to find the food? In Magnús M. Halldórsson, editor, *Structural Information and Communication Complexity - 21st International Colloquium, SIROCCO 2014, Takayama, Japan, July 23-25, 2014. Proceedings*, volume 8576 of *Lecture Notes in Computer Science*, pages 263–278. Springer, 2014.

[26] Yuval Emek, Tobias Langner, Jara Uitto, and Roger Wattenhofer. Solving the ANTS problem with asynchronous finite state machines. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II*, volume 8573 of *Lecture Notes in Computer Science*, pages 471–482. Springer, 2014.

[27] Lars Engebretsen. An explicit lower bound for TSP with distances one and two. *Algorithmica*, 35(4):301–318, 2003.

[28] Uriel Feige and Mohit Singh. Improved approximation ratios for traveling salesperson tours and paths in directed graphs. In Moses Charikar, Klaus Jansen, Omer Reingold, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 10th International Workshop, APPROX, and 11th International Workshop, RANDOM, Princeton, NJ, USA, August 20-22, 2007, Proceedings*, volume 4627 of *Lecture Notes in Computer Science*, pages 104–118. Springer, 2007.

[29] Ofer Feinerman and Amos Korman. Memory lower bounds for randomized collaborative search and implications for biology. In Marcos K. Aguilera, editor, *Distributed Computing - 26th International Symposium, DISC 2012, Salvador, Brazil, October 16-18, 2012. Proceedings*, volume 7611 of *Lecture Notes in Computer Science*, pages 61–75. Springer, 2012.

[30] Ofer Feinerman, Amos Korman, Zvi Lotker, and Jean-Sébastien Sereni. Collaborative search on the plane without communication. In Darek Kowalski and Alessandro Panconesi, editors, *ACM Symposium on Principles of Distributed Computing, PODC 2012, Funchal, Madeira, Portugal, July 16-18, 2012*, pages 77–86. ACM, 2012.

[31] Rudolf Fleischer, Thomas Kamphans, Rolf Klein, Elmar Langetepe, and Gerhard Trippen. Competitive online approximation of the optimal search ratio. *SIAM J. Comput.*, 38(3):881–898, 2008.

[32] Rudolf Fleischer and Gerhard Trippen. Experimental studies of graph traversal algorithms. In Klaus Jansen, Marian Margraf, Monaldo Mastrolilli, and José D. P. Rolim, editors, *Experimental and Efficient Algorithms, Second International Workshop, WEA 2003, Ascona, Switzerland, May 26-28, 2003, Proceedings*, volume 2647 of *Lecture Notes in Computer Science*, pages 120–133. Springer, 2003.

[33] Rudolf Fleischer and Gerhard Trippen. Exploring an unknown graph efficiently. In Gerth Stølting Brodal and Stefano Leonardi, editors, *Algorithms - ESA 2005, 13th Annual European Symposium, Palma de Mallorca, Spain, October 3-6, 2005, Proceedings*, volume 3669 of *Lecture Notes in Computer Science*, pages 11–22. Springer, 2005.

[34] Fedor V. Fomin, Pierre Fraigniaud, Nicolas Nisse, and Dimitrios M. Thilikos. Report on GRASTA 2014. Research report, 6th Workshop on GRAph Searching, Theory and Applications, 2014.

[35] Fedor V. Fomin, Pierre Fraigniaud, and Dimitrios M. Thilikos, editors. *Special issue on graph searching*, volume 399(3). Theor. Comput. Sci., 2008.

[36] Fedor V. Fomin and Dimitrios M. Thilikos. An annotated bibliography on guaranteed graph searching. *Theor. Comput. Sci.*, 399(3):236–245, 2008.

[37] Klaus-Tycho Förster and Roger Wattenhofer. Directed graph exploration. In Roberto Baldoni, Paola Flocchini, and Binoy Ravindran, editors, *Principles of Distributed Systems, 16th International Conference, OPODIS 2012, Rome, Italy, December 18-20, 2012. Proceedings*, volume 7702 of *Lecture Notes in Computer Science*, pages 151–165. Springer, 2012.

[38] Pierre Fraigniaud, Leszek Gasieniec, Dariusz R. Kowalski, and Andrzej Pelc. Collective tree exploration. *Networks*, 48(3):166–177, 2006.

[39] Pierre Fraigniaud and David Ilcinkas. Digraphs exploration with little memory. In Volker Diekert and Michel Habib, editors, *STACS 2004, 21st Annual Symposium on Theoretical Aspects of Computer Science, Montpellier, France, March 25-27, 2004, Proceedings*, volume 2996 of *Lecture Notes in Computer Science*, pages 246–257. Springer, 2004.

[40] Pierre Fraigniaud, David Ilcinkas, Guy Peer, Andrzej Pelc, and David Peleg. Graph exploration by a finite automaton. *Theor. Comput. Sci.*, 345(2-3):331–344, 2005.

[41] Pierre Fraigniaud, David Ilcinkas, and Andrzej Pelc. Tree exploration with advice. *Inf. Comput.*, 206(11):1276–1287, 2008.

[42] Alan M. Frieze, Giulia Galbiati, and Francesco Maffioli. On the worst-case performance of some algorithms for the asymmetric traveling salesman problem. *Networks*, 12(1):23–39, 1982.

[43] Shmuel Gal. *Search Games*. Academic Press, 1980.

[44] Bruce Golden, S. Raghavan, and Edward Wasil, editors. *The Vehicle Routing Problem: Latest Advances and New Challenges*. Springer US, 2008.

[45] Cor A. J. Hurkens and Gerhard J. Woeginger. On the nearest neighbor rule for the traveling salesman problem. *Oper. Res. Lett.*, 32(1):1–4, 2004.

[46] Bala Kalyanasundaram and Kirk Pruhs. Constructing competitive tours from local information. *Theor. Comput. Sci.*, 130(1):125–138, 1994.

[47] Haim Kaplan, Moshe Lewenstein, Nira Shafrir, and Maxim Sviridenko. Approximation algorithms for asymmetric TSP by decomposing directed regular multigraphs. *J. ACM*, 52(4):602–626, 2005.

[48] Dennis Komm, Rastislav Královic, Richard Královic, and Jasmin Smula. Treasure hunt with advice. In Christian Scheideler, editor, *Structural Information and Communication Complexity - 22nd International Colloquium, SIROCCO 2015, Montserrat, Spain, July 14-16, 2015, Post-Proceedings*, volume 9439 of *Lecture Notes in Computer Science*, pages 328–341. Springer, 2015.

[49] Fabian Kuhn and Rotem Oshman. The complexity of data aggregation in directed networks. In David Peleg, editor, *Distributed Computing - 25th International Symposium, DISC 2011, Rome, Italy, September 20-22, 2011. Proceedings*, volume 6950 of *Lecture Notes in Computer Science*, pages 416–431. Springer, 2011.

[50] Shay Kutten. Stepwise construction of an efficient distributed traversing algorithm for general strongly connected directed networks or: Traversing one way streets with no map. In *ICCC*, pages 446–452, 1988.

[51] Nimrod Megiddo, S. Louis Hakimi, M. R. Garey, David S. Johnson, and Christos H. Papadimitriou. The complexity of searching a graph. *J. ACM*, 35(1):18–44, 1988.

[52] Nicole Megow, Kurt Mehlhorn, and Pascal Schweitzer. Online graph exploration: New results on old and new algorithms. *Theor. Comput. Sci.*, 463:62–72, 2012.

[53] Seyed M. Mirtaheri, Mustafa Emre Dinçtürk, Salman Hooshmand, Gregor von Bochmann, Guy-Vincent Jourdan, and Iosif-Viorel Onut. A brief history of web crawlers. In James R. Cordy, Krzystof Czarnecki, and Sang-Ah Han, editors, *Center for Advanced Studies on Collaborative Research, CASCON 2013, Toronto, ON, Canada, November 18-20, 2013*, pages 40–54. IBM / ACM, 2013.

[54] Shuichi Miyazaki, Naoyuki Morimoto, and Yasuo Okabe. The online graph exploration problem on restricted graphs. *IEICE Transactions*, 92-D(9):1620–1627, 2009.

[55] Torrence D. Parsons. Pursuit-evasion in a graph. In Yousef Alavi and DonR. Lick, editors, *Theory and Applications of Graphs*, volume 642 of *Lecture Notes in Mathematics*, pages 426–441. Springer Berlin Heidelberg, 1978.

[56] Torrence D. Parsons. The search number of a connected graph. In *Proc. 9th southeast. Conf. on Combinatorics, graph theory, and computing*, 1978.

[57] Ravi Prakash. 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 (DIAL-M 1999), Seattle, Washington, USA, August 20, 1999*, pages 15–22. ACM, 1999.

[58] Bruno F. Ribeiro, Pinghui Wang, Fabricio Murai, and Don Towsley. Sampling directed graphs with random walks. In Albert G. Greenberg and Kazem Sohraby, editors, *Proceedings of the IEEE INFOCOM 2012, Orlando, FL, USA, March 25-30, 2012*, pages 1692–1700. IEEE, 2012.

[59] Daniel J. Rosenkrantz, Richard Edwin Stearns, and Philip M. Lewis II. An analysis of several heuristics for the traveling salesman problem. *SIAM J. Comput.*, 6(3):563–581, 1977.

[60] Robert Sedgewick and Jeffrey Scott Vitter. Shortest paths in euclidean graphs. *Algorithmica*, 1(1):31–48, 1986.

[61] Jasmin Smula. *Information Content of Online Problems: Advice versus Determinism and Randomization.* PhD thesis, ETH Zürich, Zürich, 2015.

[62] Dimitrios M. Thilikos, Fedor V. Fomin, Pierre Fraigniaud, and Stephan Kreutzer, editors. *Special Issue on Theory and Applications of Graph Searching Problems*, volume 463. Theor. Comput. Sci., 2012.

[63] Sundar Vishwanathan. An approximation algorithm for the asymmetric travelling salesman problem with distances one and two. *Inf. Process. Lett.*, 44(6):297–302, 1992.