Online Graph Exploration on a Restricted Graph Class: Optimal Solutions for Tadpole Graphs

Sebastian Brandt[b], Klaus-Tycho Foerster[a], Jonathan Maurer[b], Roger Wattenhofer[b]

[a] *University of Vienna, Faculty of Computer Science, 1090 Vienna, Austria*
[b] *ETH Zurich, 8092 Zurich, Switzerland*

Abstract

We study the problem of online graph exploration on undirected graphs, where a searcher has to visit every vertex and return to the origin. Once a new vertex is visited, the searcher learns of all neighboring vertices and the connecting edge weights. The goal such an exploration is to minimize its total cost, where each edge traversal incurs a cost of the corresponding edge weight.

We investigate the problem on tadpole graphs (also known as dragons, kites), which consist of a cycle with an attached path. Miyazaki et al. (The online graph exploration problem on restricted graphs, IEICE Transactions 92-D (9), 2009) showed that every online algorithm on these graphs must have a competitive ratio of $2-\varepsilon$, but did not provide upper bounds for non-unit edge weights. We show via amortized analysis that a greedy approach yields a matching competitive ratio of 2 on tadpole graphs, for arbitrary non-negative edge weights.

Keywords: Graph Exploration, Online Algorithms

1. Introduction

Exploring an unknown graph is considered to be one of the fundamental problem of robotics [1, 2]: A searcher has to visit all vertices and return to the origin. As the searcher only has information about the subgraph already explored and the adjacent vertices, the problem is commonly studied from an online perspective. In other words, the goal is to minimize the competitive ratio of the cost of the actually traversed tour versus a tour of minimum cost.

In this paper, we investigate online graph exploration on *tadpole* [3] graphs (also known as *dragons* [4] or *kites* [5], see [6]), which can be visualized as follows: start with a cycle and attach an endpoint of a path to one vertex of the cycle.

Email addresses: brandts@ethz.ch (Sebastian Brandt), klaus-tycho.foerster@univie.ac.at (Klaus-Tycho Foerster), maurerjo@ethz.ch (Jonathan Maurer), wattenhofer@ethz.ch (Roger Wattenhofer)

Background. Rosenkrantz et al. [7] proved that a greedy exploration algorithm has a competitive ratio of $\Theta(\log n)$ on general graphs with n vertices. Even though their result is over 40 years old, it is not known if algorithms with a competitive ratio of $o(\log n)$ exist—the best known lower bound is $2.5 - \varepsilon$ [8]. However, when all edges have unit traversal cost, a depth-first search (DFS) has a competitive ratio of $\frac{2n-2}{n} < 2$, with a matching lower bound of $2 - \varepsilon$ for any exploration algorithm [9, 10]. For the case of k different edge weights, a hierarchical DFS provides a competitive ratio of $2k$, which can be extended to a competitive ratio of $\Theta(\log n)$ for arbitrary edge weights [11]. Moreover, when considering strongly connected directed general graphs, a greedy algorithm achieves an optimal competitive ratio of $n - 1$ [12]. The related problem of searching for a specific vertex was covered by Smula et al. [13, 14].

With the general undirected weighted case being an open problem for many decades now, some articles studied graph exploration on restricted graph classes. Kalyanasundaram and Pruhs [15] showed that their weighted nearest neighbor algorithm yields a competitive ratio of 16 on planar graphs, which was extended to graphs of genus g by Megow et al. [11] with a competitive ratio of $16(2g+1)$—however these weighted nearest neighbor algorithms cannot break the $o(\log n)$ bound on general gaphs. Notwithstanding, Asahiro et al. [16] showed that nearest neighbor algorithms can achieve a competitive ratio of 1.5 on cycles, also giving a lower bound of 1.25. The latter result was improved by Miyazaki et al. [9] by utilizing weighted distance computations to reach a competitive ratio of $\frac{1+\sqrt{3}}{2} \approx 1.366$, with a matching lower bound of $\frac{1+\sqrt{3}}{2} - \varepsilon$.

Motivation. For graphs with different edge weights, there are no results between cycles and planar graphs, leaving a rather large gap in the competitivity landscape between the ratios of 1.366 (cycles) and 16 (planar graphs), where the latter result is also just an upper bound. We thus investigate a natural extension of the cycle, by joining the endpoint of a path to a cycle, also known as tadpole, dragon, or kite graphs [6]. Maybe interestingly, this class of graphs was also implicitly used for the lower bound construction on unit edge weight graphs in [9, 10]. We thus hope that a further investigation of these graphs will lead to a better understanding of graph exploration on special (planar) graphs.

Contribution. We prove that a greedy exploration algorithm achieves a competitive ratio of 2 on tadpole graphs, for non-negative edge weights. As Miyazaki et al. [9] proved a lower bound of $2 - \epsilon$ for unit edge weights, our result is tight.

Organization. Our article is structured as follows. We first provide a formal model in §2, followed by an intuition for the lower bound proof from Miyazaki et al. [9, § 4.2] in §3. We then provide a proof in §4 that a greedy exploration results in a matching competitive ratio of 2 on tadpoles, lastly concluding in §5.

2. Model

Graph model. We consider connected undirected graphs $G = (V, E)$ with $|V| = n$ vertices and $|E| = m$ edges, denoting an edge from u to v as (u, v).

Each edge $e \in E$ is equipped with a positive edge weight $c(e)$.[1] In particular, all investigated graphs will be tadpole graphs, which are graphs where one vertex has degree 1, one vertex has degree 3, and all other vertices have degree 2. Hence, one can imagine that a tadpole graph consists of one cycle with i vertices, to which a path (or, from now on, *stem*) of j vertices is attached. As such, all tadpole graphs can be represented e.g. in the form $T_{i,j}$, with $i \geq 3$ and $j \geq 1$.

Exploration model. We assume that the searcher has unlimited computational power and memory. Each vertex is equipped with an unique identifier (ID), where exploration proceeds as follows [11, 15]: upon arriving at a vertex v, the searcher obtains the IDs of all adjacent vertices, as well as the weight of all incident edges. In order to move to a neighboring vertex u of v, the searcher has to pay the edge weight of (v, u), even if this edge was traversed before. The goal is to perform a closed tour (a closed walk) from the starting vertex $s \in V$ that visits all vertices in V, while minimizing the accumulated cost. We call algorithms that perform such closed tours exploration algorithms.

Competitive ratio. The quality of an exploration algorithm A is rated by its competitive ratio. The competitive ratio $R_A(G)$ on a given graph G is defined as $R_A(G) = \frac{\text{cost}_A(G)}{\text{cost}_{\text{opt}}(G)}$, where $\text{cost}_A(G)$ is the cost of the tour of the algorithm A on G and $\text{cost}_{\text{opt}}(G)$ is the cost for the optimal tour. A competitive ratio of 1 is optimal and given two algorithms, the one with the lower competitive ratio is better. One can find the optimal tour by solving the traveling salesman problem (TSP) [17] on G. Lastly, the competitive ratio of A on some graph class $\mathcal{G}$ is defined as the supremum of its competitive ratio over all graphs $G \in \mathcal{G}$.

3. Lower Bounds for Tadpole Graphs

To provide some first intuition, we briefly recall the lower bound of Miyazaki et al. [9, § 4.2]. We note that Miyazaki et al. state their theorem for general graphs, but the graph construction used is just a tadpole graph:

Theorem 1 (Follows from [9], Section 4.2)**.** *For any positive constant ε, there is no $(2-\varepsilon)$-competitive online algorithm for unit weight tadpole graphs.*

We refer to [9, § 4.2] for the full proof construction and just sketch the idea:

Proof idea. Let the searcher start on a vertex s of degree 2. Now, an adversary will construct the graph on the go, s.t. the graph will look as follows, from the searcher's point of view: Two paths joined at s, and once the junction vertex t with degree 3 is found, like three paths joined at t. The challenge for the searcher is now to find out which of the two paths form the cycle. However, the adversary can "force" the searcher to explore the cycle first, i.e., the searcher needs to go back to the earlier visited junction t. If the adversary appropriately chooses the path lengths according to the searcher's decisions, the additional tour length required can be arbitrarily close to the optimal tour length. □

[1] We can omit edge weights of 0, as these edges can be explored for free in the undirected case, i.e., one might as well "contract" these edges and only consider positive weights.

On the algorithmic side, a depth-first search achieves a matching competitive ratio for unit weight graphs, but has no guarantees for weighted graphs. We show that a greedy approach yields matching upper bounds for weighted graphs.

4. Greedy Exploration of Tadpole Graphs

In this section, we will utilize an amortized analysis to prove that a greedy exploration is at most 2-competitive on tadpole graphs. We first formalize the notion of a greedy exploration algorithm and then present our proof.

Definition 2. *A greedy exploration algorithm proceeds as follows, until every vertex is visited: From all vertices that are known but have not yet been visited, pick the vertex to which the best known path is shortest, and visit via this shortest path. Once all vertices have been visited, return to the starting vertex s along a shortest path.*

Theorem 3. *A greedy exploration algorithm has a competitive ratio of* 2 *on weighted tadpole graphs.*

Proof. We first introduce some preliminaries and then prove the theorem statement by case distinction.

Step by step. It will be useful to introduce the notion of a *step* of the algorithm, where a step is the decision which previously univisited vertex to visit next, plus the actual visit itself. As such, if a graph has n vertices, a greedy online exploration algorithm takes $n-1$ steps and then return to the starting vertex s.

Charging an edge for the next step. Consider the situation that the searcher is currently on a vertex v *with* an unvisited neighbor w. If v has more than one unvisited neighbor, let w be the one that connects to v with the cheapest edge. In the next step, a greedy exploration algorithm currently on a vertex v will either a) visit a yet unvisited neighboring vertex w by traversing $e = (w, v)$ or b) visit a yet unvisited vertex w' by a path P that does not contain (w, v), where not necessarily $w' \neq w$. Due to the greedy nature of the exploration algorithm, in case b) the cost of P will be at most the cost of e, i.e., $c(P) \leq c(e)$ (Fig. 1). Similarly, in case a) the cost $c(e)$ is cheaper than any other path to a yet unvisited vertex. Hence, for an amortized analysis, in such a situation we can say that we *charge* the cost of the next step to the edge (w, v).

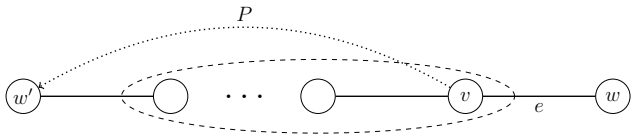

Figure 1: Illustration of the case b) when charging an edge for the next step, where the dashed ellipse marks nodes already visited. When the searcher is on v and decides to visit w' instead of w, then the cost of the dashed path P is at most the cost of the edge $e = (w, v)$.

Charging the actual path. Next, consider the situation that the searcher is currently on a vertex v *without* an unvisited neighbor or that all vertics have already been visited, but $v \neq s$. Then, for the next step or the return to v, we say that we charge the actual taken path to the algorithm, i.e., the cost of each traversed edge.

A first upper cost bound. We now combine both charging ideas for a first simple upper bound, to give some intuition for the remainder of the proof. To this end, we ask the question how often we cannot charge an edge for the next step. By definition, this can only be the case if there is no unvisited neighbor w of the current vertex v. Such a situation can occur only in two cases:

1. The searcher is not on s and at the end of the stem, or
2. the searcher is on the cycle, and all cycle vertices have been visited.

For the first case, if the end of the stem is s, then we will only be on s at the beginning (where we have an unvisited neighbor) or upon completion of the algorithm. For the second case, assume not all cycle vertices have been visited. Denote the current vertex neighbors of v on the cycle by w' and w'': when they were visited for the first time, it was either because they were s or because they were at the end of a step. Similarly, v was visited for the first time at the end of a step. However, in order to visit both w', w'' without visiting v, all vertices between w', w'' on the cycle must have been visited, a contradiction.

Both cases will only occur at most once until the algorithm has completed its exploration, as the searcher will not choose an already explored vertex for the endpoint of a step (note that returning to s was not defined as a step). As thus, charging a path will only happen twice, and for the second time, it is the return to s. Hence, a greedy exploration algorithm will charge every edge at most once, and charge two paths, i.e., the total exploration cost is at most three times all edge weights in the tadpole graph.

Improving the upper cost bound. Our above upper bound is still far from the theorem statement itself: In our amortized analysis, we would like to include the cost of an edge only twice as often as an optimal solution uses it, and furthermore, an optimal solution might skip a very expensive cycle edge. To this end, we first investigate the optimal tour shapes.

Shape of the optimal tour. Observe that the optimal tour for a tadpole graph consists of a) the optimal walk through the cycle [16] and b) twice the attached path, the stem. As such, the optimal tour can have one of the following two shapes:

1. Traverse all the edges in the cycle once and all edges in the stem twice.
2. Traverse all the edges in the cycle, except for one, twice, and the stem twice. The excluded edge e_∞ has a weight of at least the remaining cycle.

To break ties, we say that shape 1 is optimal, in case both have the same cost. As such, we assume that e_∞ has a weight *greater* than the remaining cycle. We start our proof by case distinction with Shape 2.

Shape 2. We can show the theorem statement for Shape 2 by proving that a greedy exploration algorithm will utilize the cost of every edge $e \neq e_\infty$ at most 4 times. Hence, our analysis for Shape 2 would be complete if we can remove the charging of e_∞ from the costs, as every other edge cost is utilized at most 3 times. Observe that instead of charging e_∞, we can also charge the actual path taken in this step, as a greedy exploration algorithm will never traverse e_∞: If the vertex w across e_∞ is unvisited, there will always be a cheaper path to reach a yet unvisited vertex (possibly w), and if all vertices have been visited, then the shortest path to the start s will never include e_∞. In conclusion, in Shape 2, we will include the cost of every edge $e \neq e_\infty$ at most 4 times, at most once by direct charging, and at most 3 times by charging the path taken. It remains to investigate Shape 1.

Shape 1. In this shape, the optimal solution will traverse every cycle edge once and every stem edge twice. We keep the edge charging analysis for this shape, but will fine-tune the charging of the paths. Recall for this argument that we can utilize stem edges 4 times. The goal is thus to utilize every cycle edge only once via path charging, instead of twice, improving the upper bound of 3 to 2.

We start with the case where the searcher hits the end of the stem first, where the end of the stem is not s. Let the cycle edges that are traversed for the next step be $e_1, \ldots, e_k, k \geq 0$ (this set of edges might be empty). For the next steps, until all vertices are visited, we can charge the edges again. It then remains to return to s from some v. If the shortest path from s to v does not utilize $e_1, \ldots, e_k$, then the upper bound of 2 holds for this case. Else, if it does, consider the whole weight of the cycle C, denoted by $c(C)$. We can deduce that $\sum_{i \leq k} c(e_i) \leq 0.5 \cdot c(C)$, else there would have been a cheaper way (the other way "around") to pick these edges initally. Moreover, consider the combined cost of the cycle edges used for the path from v to s: Again, their combined cost may also only be $0.5 \cdot c(C)$ at most. Hence, in total, we charge every cycle edge at most once and in addition at most $2 \cdot 0.5c(C) = c(C)$, i.e., we are 2 competitive in this case as we can utilize stem edges 4 times.

We next cover the case where we do not hit the end of the stem first, but rather that the searcher is on the cycle and has visited all cycle vertices with the latest step. Next, the searcher will visit a vertex v' on the stem, where we recall that we can utilize the cost of each stem edge 4 times. Again, let $e_1, \ldots, e_k$ be the cycle edges traversed along this shortest path to v'. Until the end of the stem is hit, the searcher will now just charge yet unutilized stem edges. Once at the end of the stem, it remains to return to s. We can use the same argument as before, namely that the both the sum of the cost in the cycle back to s and the cost of $e_1, \ldots, e_k$ are at most $0.5 \cdot c(C)$, respectively, i.e., we are 2-competitive.

It remains to cover the case where the searcher starts at the end of the stem. In this case, each edge will only be charged once until all vertices on the cycle have been visited. Afterwards, the searcher will return on the shortest path to s, i.e., the cost of each edge will be utilized at most a second time, finishing the proof construction. □

5. Conclusion and Outlook

We studied the online exploration of tadpole graphs, showing that a greedy exploration achieves a competitive ratio of 2. Our results also hold on weighted graphs, with a matching lower bound of $2 - \varepsilon$ already on unit weight tadpoles. Moreover, we have also presented the first non-trivial graph class where a greedy graph exploration is optimal from a competitive point of view.

We see our work as a first step towards charting the landscape of graph exploration, a problem which continues to puzzle researchers for many decades. Next steps could be the investigation of more broad graph classes, such as unicyclic graphs, lollipop graphs, barbell graphs, or wheel graphs, with the hope that a better understanding of these specific graph structures will bring us closer to bridging the large gap for exploration on general graphs.

Bibliographical note.
A preliminary version of the results in this article appeared in [18].

References

[1] W. Burgard, M. Moors, D. Fox, R. G. Simmons, S. Thrun, Collaborative multi-robot exploration, in: ICRA, IEEE, 2000, pp. 476–481 (2000).

[2] R. Fleischer, G. Trippen, Exploring an unknown graph efficiently, in: ESA, Vol. 3669 of Lecture Notes in Computer Science, Springer, 2005, pp. 11–22 (2005).

[3] K. Koh, D. Rogers, H. Teo, K. Yap, Graceful graphs: some further results and problems, Congr. Numer 29 (1980) 559–571 (1980).

[4] M. Truszczynski, Graceful unicyclic graphs, Demonstatio Mathematica 17 (1984) 377–387 (1984).

[5] S. Kim, J. Y. Park, On super edge-magic graphs, Ars Combinatoria 81 (2006) 113–127 (2006).

[6] J. A. Gallian, A dynamic survey of graph labeling (twenty-first edition, 2018), The Electronic Journal of Combinatorics (2018).
URL `http://www.combinatorics.org/ojs/index.php/eljc/article/viewFile/DS6/pdf`

[7] D. J. Rosenkrantz, R. E. Stearns, P. M. Lewis, II, An analysis of several heuristics for the traveling salesman problem, SIAM journal on computing 6 (3) (1977) 563–581 (1977).

[8] S. Dobrev, R. Královic, E. Markou, Online graph exploration with advice, in: SIROCCO, Vol. 7355 of Lecture Notes in Computer Science, Springer, 2012, pp. 267–278 (2012).

[9] S. Miyazaki, N. Morimoto, Y. Okabe, The online graph exploration problem on restricted graphs, IEICE Transactions 92-D (9) (2009) 1620–1627 (2009).

[10] N. Morimoto, Design and Analysis of Algorithms for Graph Exploration and Resource Allocation Problems and Their Application to Energy Management (Kyoto University), Ph.D. thesis (2014).

[11] N. Megow, K. Mehlhorn, P. Schweitzer, Online graph exploration: New results on old and new algorithms, Theor. Comput. Sci. 463 (2012) 62–72 (2012).

[12] K.-T. Foerster, R. Wattenhofer, Lower and upper competitive bounds for online directed graph exploration, Theor. Comput. Sci. 655 (2016) 15–29 (2016).

[13] D. Komm, R. Královic, R. Královic, J. Smula, Treasure hunt with advice, in: SIROCCO, Vol. 9439 of Lecture Notes in Computer Science, Springer, 2015, pp. 328–341 (2015).

[14] J. Smula, Information content of online problems: advice versus determinism and randomization, Ph.D. thesis, ETH Zurich, Switzerland (2015).

[15] B. Kalyanasundaram, K. Pruhs, Constructing competitive tours from local information, Theor. Comput. Sci. 130 (1) (1994) 125–138 (1994).

[16] Y. Asahiro, E. Miyano, S. Miyazaki, T. Yoshimuta, Weighted nearest neighbor algorithms for the graph exploration problem on cycles, Inf. Process. Lett. 110 (3) (2010) 93–98 (2010).

[17] D. L. Applegate, R. E. Bixby, V. Chvatal, W. J. Cook, The traveling salesman problem: a computational study, Princeton University Press, 2011 (2011).

[18] J. Maurer, Graph Exploration, thesis, ETH Zurich, Switzerland (2015).