Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Advice Complexity of Online Algorithms

Master's Thesis

2014

Bc. Peter Fulla

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Advice Complexity of Online Algorithms

Master's Thesis

Study programme:	Computer Science
Field of study:	2508 Computer Science
Department:	Department of Computer Science
Supervisor:	prof. RNDr. Rastislav Královič, PhD.

Bratislava, 2014

Bc. Peter Fulla

22188809

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	Bc. Peter Fulla
Študijný program:	informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor:	9.2.1. informatika
Typ záverečnej práce:	diplomová
Jazyk záverečnej práce:	anglický

Názov: Advice complexity of online algorithms
On-line algoritmy s pridanou informáciou

Cieľ: Nájsť horné a dolné odhady na veľkosť rady vo vybraných on-line problémoch.

Vedúci:	prof. RNDr. Rastislav Královič, PhD.
Katedra:	FMFI.KI - Katedra informatiky
Vedúci katedry:	doc. RNDr. Daniel Olejár, PhD.
Dátum zadania:	08.11.2012

Dátum schválenia: 28.11.2012 prof. RNDr. Branislav Rovan, PhD.
garant študijného programu

....
študent vedúci práce

I would like to thank my supervisor Rastislav Královič for advice; he presented me with a most compelling subject to research.

I am also extremely grateful to my family and friends for their support, help, and distractions.

Abstract

In the online graph exploration problem, an agent explores an unknown weighted graph aiming to visit each vertex at least once and return to the starting node while minimizing the total cost of its walk. The topology of the graph is revealed to the agent gradually: After visiting a node, the agent learns all edges incident to it. We provide some additional information about the input to the agent in the form of advice, and study the trade-off between the amount of provided advice and the competitive ratio (with respect to an optimal offline algorithm knowing the graph beforehand). In this thesis, we focus on the exploration problem on two restricted classes of graphs. We give asymptotically tight bounds of $\Theta(\lg n)$ and $\Theta(n \lg n)$ on advice necessary to yield an optimal solution on cycles and unweighted graphs respectively. Furthermore, we propose a new algorithm that explores cycles achieving the competitive ratio of $1 + 3/2^{2^k}$ (for any constant k) while requiring advice of size $O(k)$ bits. Finally, we provide a lower bound of $\Omega(n)$ on advice necessary to achieve a competitive ratio better than $1 + \frac{1}{\ln 16 - 1} \approx 1.564$ on unweighted graphs.

Keywords: online graph exploration, competitive analysis, advice complexity, online algorithms

Abstrakt

V probléme prehľadávania grafu online sa agent pohybuje po neznámom ohodnotenom grafe s cieľom navštíviť každý jeho vrchol aspoň raz a vrátiť sa do štartovacieho vrcholu, pričom sa snaží minimalizovať celkovú prejdenú vzdialenosť. Agent sa topológiu grafu dozvedá postupne: Až keď navštívi vrchol po prvýkrát, prezradíme mu, aké hrany z neho vedú. Agentovi navyše poskytneme informáciu o grafe vo forme rady. Predmetom nášho skúmania je vzťah medzi množstvom poskytnutej rady a dosiahnuteľným kompetitívnym pomerom v porovnaní s optimálnym offline algoritmom (t.j. takým, ktorý prehľadávaný graf pozná vopred). V tejto diplomovej práci sa zameriame na dve triedy grafov: cykly a neohodnotené grafy. Ukážeme, že optimálny agent pre ne potrebuje $\Theta(\lg n)$ resp. $\Theta(n \lg n)$ bitov rady. Okrem toho predstavíme nový algoritmus, ktorý prehľadáva cykly s kompetitívnym pomerom $1 + 3/2^{2^k}$ (pre ľubovoľnú konštantu k), pričom vyžaduje radu veľkosti $O(k)$ bitov. Nakoniec ukážeme, že ľubovoľný algoritmus prehľadávajúci neohodnotené grafy s kompetitívnym pomerom menším ako $1 + \frac{1}{\ln 16 - 1} \approx 1.564$ musí prečítať $\Omega(n)$ bitov rady.

Kľúčové slová: prehľadávanie grafu online, kompetitívna analýza, množstvo pridanej informácie, online algoritmy

Contents

Introduction

In the field of computational complexity, we usually consider algorithms restricted to use only a limited amount of certain resources, e.g. the number of steps taken or the size of the allocated memory. Online problems provide a similar challenge: An algorithm is supposed to produce a partial output before it finishes reading the input, therefore it is generally unable to find an optimal solution. In the case of online problems, an algorithm is limited in its access to information about the future input.

There have been numerous attempts to examine the impact of additional information provided to an online algorithm on its performance. Traditionally, this information is presented in a qualitative manner, for example as a promise of certain properties of the input instance. On the other hand, the study of advice complexity offers a quantitative approach: An algorithm receives advice that has been specially tailored for a particular input instance. The advice is in the form of a binary sequence; by limiting its length we can control the amount of information an agent obtains.

In this thesis, we focus exclusively on the online graph exploration problem, in which a mobile agent explores an unknown undirected weighted graph. Starting in a designated node, the agent travels through the edges of the graph aiming to visit every vertex at least once and to return to the initial vertex. For every edge traversal, the agent incurs the cost equal to the length of that edge. Naturally, the objective is to find an exploration walk with the minimum total cost.

The question whether there is an algorithm solving the online graph exploration problem with a constant competitive ratio (i.e. incurring at most a constant multiple of the optimal cost) remains open. Several authors have also studied variants of the problem, mainly exploring graphs from restricted classes. We contribute to this work by investigating the advice complexity of exploring cycles and unweighted graphs. We prove tight lower and upper bounds on the amount of advice necessary to yield an optimal solution on these classes of graphs. We also devise a new algorithm on cycles that utilizes a constant number of advice bits and substantially improves its efficiency. Moreover, we prove a lower bound on the competitive ratio of agents exploring unweighted graphs with a sublinear amount of advice.

In the first chapter, we present an overview of online algorithms in general with a special focus on advice complexity and the online graph exploration problem. We summarize related work, known results, and open problems. The following two chapters consist of our results. In the second chapter, we consider a special case of the online graph exploration problem, when the given graph is a cycle. In the third chapter, we look at the exploration of unweighted graphs.

Chapter 1

Overview

In this chapter, we present the reader with an overview of online algorithms, competitive analysis, and advice complexity. We define basic concepts used in this thesis, especially the online graph exploration problem. Finally, we discuss related work and known results, and summarize our contribution.

1.1 Online Algorithms

In online problems, an algorithm receives the input in several parts and is also required to produce output in such a fashion. Formally, an input instance of an online problem is a sequence $I = (x_1, x_2, \ldots, x_n)$. An online algorithm A computes the output sequence $A(I) = (y_1, y_2, \ldots, y_n)$, where $y_i = f(x_1, x_2, \ldots, x_i)$ for some function f.[1] Specifically, A has no information about $x_{i+1}, \ldots, x_n$ at the moment it outputs y_i. The output is then evaluated by a cost function; the goal of the problem may be to minimize or maximize the cost. We refer the reader to [3] for further information on online algorithms.

An extremely simple example of an online problem is SkiRental (see [3]), in which a skier is repeatedly asked to choose between buying a pair of skis and renting them in order to attend a series of ski trips. The cost of renting is lower, but the skier obtains the gear only temporarily – for the length of one trip. Buying, on the other hand, settles all subsequent queries, as the skis remain forever in the skier's possession. Finding the optimal solution is easy if we know the number of trips. However, we lack this information in the online setting, which prevents us from achieving the optimal cost.

The performance of an online algorithm is often measured by *competitive ratio* (introduced in [11]) when we compare the cost of the algorithm's solution with that of an optimal solution. In the case of a minimization online problem, an algorithm A is called r-competitive ($r \geq 1$) if there exists some constant $q \geq 0$ such that for any input instance I,

$$C(A(I)) \leq r \cdot C(\text{opt}(I)) + q$$

where C is the cost function and $\text{opt}(I)$ is an optimal solution of instance I. If we allow for randomized algorithms, we require the bound holding for the

[1] In the case of randomized online algorithms, the elements of the output sequence are also a function of the random bits used so far.

expected value of the incurred cost over all random strings:

$$E[C(A(I))] \leq r \cdot C(\mathrm{opt}(I)) + q$$

Analogous definitions apply to maximization problems.

Let us consider our toy example, the SkiRental problem, with the cost of renting a pair of skis equal to 1 and the cost of buying them equal to some constant $c > 1$ which is a part of the input. An algorithm that responds to the first query by buying a pair of skis regardless of their price is not r-competitive for any constant r: The sequence of trips may end right after the first one, in which case the optimal action would be renting, and the algorithm incurs c times greater cost than the optimal one. Similarly, an algorithm that never buys skis cannot be r-competitive, as the length of the input sequence is unbounded. However, there is a 2-competitive algorithm: We start by renting the skis and continue doing so until the cumulative cost of renting exceeds c. At that moment, we buy a pair of skis instead of renting them. Clearly, we pay at most twice the optimum cost in such a case. If the input sequence ends before us buying the skis, we have found an optimal solution.

Another well-studied example of an online problem is Paging: We maintain a buffer of k items called pages. The input is a sequence of page requests. If the requested page occurs in the buffer, nothing needs to be done; otherwise we must choose a victim – a page in the buffer that will be replaced by the requested page. A request of the latter type is called a *page fault*. The goal of a paging algorithm is to minimize the number of page faults. As was shown in [11], no deterministic algorithm can be better than k-competitive, and a k-competitive algorithm exists. However, there is a randomized algorithm with the competitive ratio of $\Theta(\lg k)$ (see [5]).

1.2 Advice Complexity

An online algorithm is forced to make irreversible decisions at the time it has seen only a part of the input, even though the unknown part may have a profound impact on the total cost of the algorithm's output. The notion of competitive ratio is used to capture the inherent loss of performance (in comparison with a hypothetical optimal solution) associated with a particular online problem. An interesting question is what amount of information about the input does an online algorithm actually lack: If it knew the entire input in advance, it would be able to produce an optimal solution,[2] but this may be an unnecessary amount. For example, in the SkiRental problem, an algorithm does not need to know the exact number of ski trips in the input in order to produce an optimal solution. Instead, one bit of information is sufficient: should it buy a pair of skis immediately, or should it rent them every time?

The concept of advice complexity enables us to investigate the trade-off between the amount of additional information and the achievable competitive ratio. Formally, we provide an algorithm A with a binary string ϕ called *advice*. Let us denote by $A^{\phi}(I)$ the output produced by A on an input instance I given advice ϕ. Algorithm A is r-competitive with advice complexity $s(n)$ if there

[2]Let us note that we are not concerned with the computational complexity of the problem: An algorithm is not restricted in the number of steps or the size of memory it use.

exists some constant $q \geq 0$ such that for any n and any input instance I of size n, there exists some advice ϕ such that

$$C(A^{\phi}(I)) \leq r \cdot C(\mathrm{opt}(I)) + q$$

and at most $s(n)$ bits of ϕ have been accessed during the computation of $A^{\phi}(I)$. For more detailed information, we refer the reader to [2].

As we mentioned earlier, one bit of advice is sufficient for an algorithm to find the optimal solution of the SkiRental problem. Let us consider the Paging problem now. To achieve optimality, we could encode an optimal sequence of victims into the advice. However, just one bit per page request is sufficient. On the other hand, an amortized constant number of advice bits per request is necessary for an algorithm with a competitive ratio less than 1.25. (For these results, see [2].)

1.3 Online Graph Exploration Problem

The problem was introduced in [7] as an online variant of the *travelling salesman problem* (TSP). In the online setting, there is a mobile agent exploring an unknown undirected weighted graph. It starts in a designated vertex s and travels through the edges of the graph. For every edge traversal, the agent incurs the cost equal to the weight of the traversed edge. The topology of the graph is not known to the agent beforehand; instead, it is revealed gradually as the agent visits new vertices. Upon entering a vertex for the first time, the agent learns the set of edges incident to that vertex together with their lengths and endpoints. Every vertex in the graph is assigned a unique label, so the agent is able to discern two vertices even if it has not visited them yet. The goal of the agent is to visit every vertex of the graph at least once, and return to the starting node s, while minimizing the total cost incurred.

Known local heuristics for the TSP can be also used in this online variant, but none of them achieves a constant competitive ratio. For example, the *nearest neighbour* (NN) algorithm, which repeatedly chooses the nearest yet unvisited vertex and travels to it, is in the worst case $\Theta(\lg n)$-competitive even on planar graphs with unit weights (see [8, 6]). The existence of an algorithm with a constant competitive ratio remains an open question. The best known lower bound on the competitive ratio of a deterministic agent is $5/2 - \varepsilon$ (see [4]). For graphs of genus at most g, a $16(1 + 2g)$-competitive algorithm was devised in [8] (i.e. a 16-competitive algorithm exploring planar graphs).

There have also been results concerning the online graph exploration problem on cycles. In [1], it is shown that the NN algorithm attains the competitive ratio of 1.5. A tight bound is given in [9], where the authors present an algorithm with the competitive ratio of $\frac{1+\sqrt{3}}{2} \approx 1.366$ and show that no algorithm with a lower competitive ratio exists.

In another variant of the graph exploration problem, the explored graph is unweighted (or equivalently, all edges have the same weight). As was shown in [9], the depth-first search (DFS) algorithm is optimal with its competitive ratio of 2.

In [4], the authors study the advice complexity of the online graph exploration problem on general graphs. Namely, they show that $\Omega(n \lg n)$ bits of advice are necessary for an optimal agent. Moreover, they devise a deterministic algorithm that uses $O(n)$ bits of advice and achieves a constant competitive ratio.

Our Results

In this thesis, we examine the advice complexity of the online graph exploration problem on two classes of graphs: cycles and unweighted graphs.

We show that $\lg n + O(\lg\lg n)$ bits of advice[3] are sufficient for an agent to explore a cycle of n vertices with the minimum total cost. On the other hand, we prove that any optimal agent requires at least $\lg n + O(1)$ advice bits. Moreover, we devise an algorithm able to utilize a constant number of bits and substantially improve its performance (in comparison with the NN algorithm on which it is based). For any constant k, our algorithm reads $2k+1$ advice bits and explores cycles attaining the competitive ratio of $1 + 3/2^{2^k+1}$.

When exploring unweighted graphs, $O(n \lg n)$ bits of advice are sufficient to find an optimal solution. If we consider only graphs with the maximum degree bounded from above by Δ, our algorithm requires $O(n \lg \Delta)$ bits. We also present a different optimal algorithm with the advice complexity of $O(m)$, where m is the number of edges. Conversely, we prove that $\Omega(n \lg n)$ advice bits are necessary to explore unweighted graphs optimally, thus generalizing the lower bound in [4]. In addition, we show that any algorithm with a competitive ratio less than $1 + \frac{1}{\ln 16 - 1} \approx 1.564$ must obtain $\Omega(n)$ bits of advice.

[3]Throughout the thesis, we use $\lg n$ as a notation of the binary logarithm of n.

Chapter 2

Exploring Cycles

At first glance, the problem of exploring a cycle appears to be trivial. However, if the length of the longest edge is greater than the sum of remaining lengths, no optimal solution uses the longest edge.

To capture this fact formally, let us denote the length of the longest edge by ℓ_m and the sum of lengths of all edges by L. The walk from the starting node s around the cycle and back to s uses every edge exactly once and therefore costs L. An alternative walk starts in s, leads to one of the longest edge's endpoints, then turns back and continues through the rest of the cycle to visit the opposite endpoint, and finally returns to s. The cost of this walk is $2(L - \ell_m)$, as it uses all edges except the longest one exactly twice. It can be easily shown that the first walk is optimal when $\ell_m \leq L/2$, otherwise the second walk is optimal.

In this chapter, we show that $\Theta(\lg n)$ bits of advice are both necessary and sufficient to explore a cycle of n vertices optimally. Moreover, we present an algorithm based on the nearest neighbour heuristic that attains a much lower competitive ratio using only a constant number of advice bits.

2.1 Optimal Solution

In this section, we give asymptotically matching lower and upper bounds on the amount of advice necessary to yield an optimal solution on a cycle.

2.1.1 Upper Bound on Advice

Whether it is optimal to use every edge of the cycle or to skip the longest one, we can communicate it to the agent in a single bit of advice. In the second case, the agent must be able to recognize the longest edge, even before seeing the complete cycle. Therefore we encode to the advice the number of edges the agent can traverse until it reaches the longest edge.

Let n be the number of nodes of the cycle. We need to encode an integer from the range $0, \ldots, n-1$, which can be done using $\lceil \lg n \rceil$ bits. As the agent does not know the number of nodes in advance, we use additional $2\lceil \lg \lceil \lg n \rceil \rceil$ bits to encode the value of $\lceil \lg n \rceil$ in a prefix-free code, interleaving the binary digits with ones and appending a zero after the last digit. Hence the total advice given to the agent is $\lg n + O(\lg \lg n)$ bits.

2.1.2 Lower Bound on Advice

For any $k \geq 2$ let us define a cycle C_k with the following sequence of edge lengths (starting and ending in the node s): $1, 3, 9, \ldots, 3^{k-1}, 3^k, 3^{k-1}, \ldots, 9, 3, 1$.

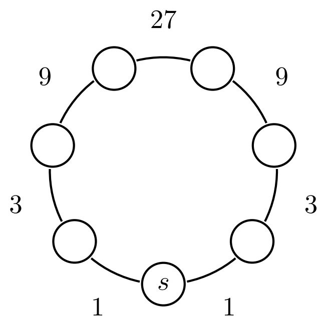

Figure 2.1: Graph C_3

The graph C_k consists of $2k+1$ edges, the sum of their lengths is $2 \cdot 3^k - 1$ and the greatest length is 3^k. As $3^k > (2 \cdot 3^k - 1)/2$, no optimal walk uses the longest edge.

Note that a deterministic agent cannot behave optimally on both graphs C_i and C_j for $i < j$. The graphs are indistinguishable after traversing the first i edges, hence in the next step the agent moves identically on both of them. However, on the graph C_j the agent must traverse the next edge of length 3^i, but on the graph C_i it must not.

Let us consider all cycles C_k with at most n nodes, i.e. the graphs C_2, C_3, $\ldots$, $C_{\lfloor (n-1)/2 \rfloor}$. An optimal agent cannot receive the same advice string for any two of them, thus it requires at least

$$\lceil \lg(\lfloor (n-1)/2 \rfloor - 1) \rceil = \lg n + O(1)$$

bits of advice.

2.2 Using Small Advice

The algorithm NN (nearest neighbour) always visits the unvisited node closest to the current node. It was shown by [1] that on cycles, NN is 1.5-competitive. In this section, we present an algorithm based on NN that makes use of a constant number of advice bits to achieve a lower competitive ratio. Again, we focus only on the case when $\ell_m > L/2$, since otherwise we can ensure the optimal behaviour of the agent with a single advice bit.

2.2.1 Algorithm

If we are not allowed to traverse the longest edge, then for every node v there is a unique path from s to v. We call an edge *heavy* if it is longer than the path from s to it. Note that the longest edge of the cycle is heavy (both of its corresponding paths are shorter than ℓ_m). Just like NN, our algorithm does not traverse the longest edge and hence can recognize a heavy edge in the moment it visits its first endpoint.

Let r be an integer greater than 1. For a given input, we count the heavy edges on each of the two paths connecting s and the endpoints of the longest edge. We encode to advice these two values modulo r. (To do that, $\lceil 2 \lg r \rceil$ bits are sufficient.)

Our algorithm acts like NN with one exception. When the nearest node is not adjacent to the current one, there is an untraversed heavy edge e incident to the current node. If the number of heavy edges on the path from s to e is not congruent modulo r to the value given in advice, the agent can be sure that e is not the longest edge. If that is the case, the agent diverges from the nearest neighbour strategy and continues by traversing e; otherwise it moves as the NN algorithm.

After visiting all the nodes, the agent returns to s following the shortest path.

2.2.2 Competitive Analysis

There are two paths connecting s and the longest edge; we call them A and B in an arbitrary order. Let us define D_A as the shortest subpath of A starting in an endpoint of the longest edge and containing r heavy edges. If there are fewer than r heavy edges in A, we define $D_A = A$. Let T_A be the subpath of A consisting of all remaining edges. Analogously, we define subpaths D_B and T_B of B. Let T be the concatenation of T_A and T_B.

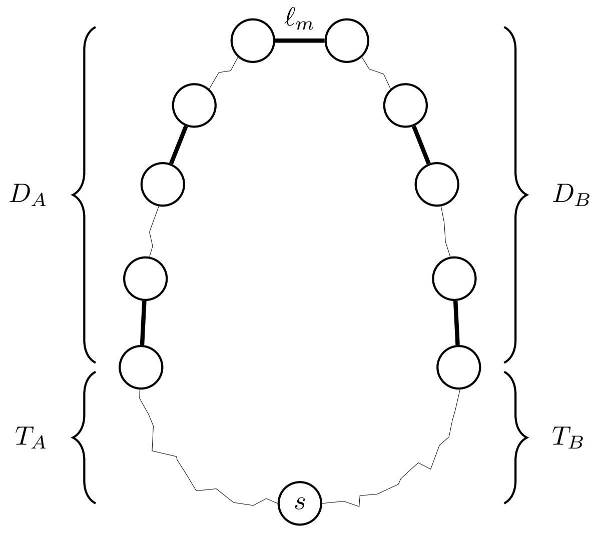

Figure 2.2: An illustration of the notation used in the analysis for $r = 2$. The thick edges are heavy, the thin squiggly lines represent paths.

We show that the path D_A (D_B) forms a substantial part of A (B).

Lemma 1. *For any cycle,* $|D_A| \geq (2^r - 1) \cdot |T_A|$ *and* $|D_B| \geq (2^r - 1) \cdot |T_B|$.

Proof. If there are fewer than r heavy edges in A, the path T_A is empty and therefore the claim holds.

Let there be r heavy edges in A, we consider them in order of their distance from s. The first heavy edge (nearest to s) is, by definition, longer than T_A. The second heavy edge is longer than the concatenation of T_A and the first heavy edge, thus its length is greater than $2 \cdot |T_A|$. By induction, the length of the i-th heavy edge is greater than $2^{i-1} \cdot |T_A|$. These inequalities together imply:

$$|D_A| \geq |T_A| + 2 \cdot |T_A| + \cdots + 2^{r-1} \cdot |T_A| = (2^r - 1) \cdot |T_A|$$

The proof of the other claim is analogous. □

Now we give an upper bound on the length of the agent's walk.

Lemma 2. *For any cycle, the agent's cost on it is at most* $2(|D_A|+|D_B|)+5\cdot|T|$.

Proof. Let us divide the agent's walk into two phases. During the first phase the agent moves only using the edges of T; the second phase starts with a traversal of an edge from $D_A \cup D_B$.

To bound the cost of the first phase, we place on every edge of T a number of coins equal to the length of the edge. In addition, we place $|T|$ coins on both edges from $D_A \cup D_B$ incident to T. We claim that these $3 \cdot |T|$ coins are sufficient to pay for the cost of the first phase. If the agent traverses the edge e incident to the current node, we pay for this move with coins placed on e. Otherwise, the agent moves to a node that is nearer than the opposite endpoint of e, thus the coins on e are also sufficient to pay for this.

Without loss of generality, let us assume that the second phase starts with a traversal of an edge from D_A. Because of the check of congruency modulo r, our algorithm continues traversing all edges of D_A until it reaches the cycle's longest edge. After that, it turns back to visit the remaining nodes. It continues traversing in the same direction until it reaches the other endpoint of the longest edge (as $\ell_m > L/2$, the shortest path to a node does not use the longest edge) and then returns to s. The cost of the second phase is therefore at most $2(|D_A| + |D_B|) + 2 \cdot |T|$. □

An optimal walk has cost $2(L-\ell_m) = 2(|D_A|+|D_B|+|T|)$. The competitive ratio of our algorithm is therefore at most

$$\frac{2(|D_A| + |D_B|) + 5|T|}{2(|D_A| + |D_B| + |T|)}$$

From Lemma 1 we have a lower bound on $|D_A|$ and $|D_B|$; applying it we get the competitive ratio of

$$1 + \frac{3}{2} \cdot \frac{|T|}{(2^r - 1) \cdot (|T_A| + |T_B|) + |T|} = 1 + \frac{3}{2^{r+1}}$$

using $1 + \lceil 2 \lg r \rceil$ bits of advice.

Chapter 3

Exploring Unweighted Graphs

The edges of a graph to explore may have in general different lengths. However, if we restrict the class of possible inputs to graphs whose edges are of the same length, we obtain a simpler problem. The cost of a walk in such a graph is uniquely determined by the number of edges it consists of, therefore the actual length of edges becomes irrelevant to the competitive analysis. We call such graphs unweighted, although it is technically incorrect.

It was shown by [9] that on unweighted graphs, the depth-first search algorithm (DFS) is optimal with the competitive ratio of 2. In this chapter, we prove that any agent with a competitive ratio less than $1 + \frac{1}{\ln 16 - 1} \approx 1.564$ requires $\Omega(n)$ bits of advice. We also establish a tight bound of $\Theta(n \lg n)$ on advice necessary for an agent to find an optimal solution. If the graph's maximum degree is limited to Δ, the bound becomes $\Theta(n \lg \Delta)$. Furthermore, we describe another algorithm that incurs the minimum possible cost while reading $O(m)$ advice bits, where m is the number of edges of the explored graph.

3.1 Optimal Solution

In this section, we give asymptotically matching lower and upper bounds on the amount of advice necessary to yield an optimal solution on an unweighted graph.

3.1.1 Upper Bound on Advice

Let us denote the maximum degree of a given graph by Δ. We are going to describe an optimal agent A with advice complexity of $O(n \lg \Delta)$, but first we prove an upper bound on the length of an optimal exploration walk.

Lemma 3. *Let G be an unweighted graph with n vertices. Then the length of an optimal exploration walk on G does not exceed $2(n-1)$.*

Proof. We show that the length of the exploration walk on G found by the DFS algorithm equals $2(n-1)$.

The DFS algorithm constructs a spanning tree T of the explored graph. It traverses exclusively the edges belonging to T, each of them exactly twice. As any spanning tree of a connected graph with n vertices consists of $n-1$ edges, the DFS algorithm's cost depends only on the value of n and equals $2(n-1)$. This gives us an upper bound on the length of an optimal walk. □

We provide agent A with advice that completely specifies an arbitrary optimal walk. For every edge in the walk, we need to encode its index in the list of edges incident to the current vertex. As the degree of any vertex does not exceed Δ, we can encode the index in $\lceil \lg \Delta \rceil$ bits. Altogether, $2(n-1)\lceil \lg \Delta \rceil = O(n \lg \Delta)$ bits of advice are sufficient to represent the entire optimal walk.

In general, the maximum degree of a graph may reach $n-1$, therefore we proved the upper bound of $O(n \lg n)$.

3.1.2 Upper Bound on Advice for Sparse Graphs

The previous algorithm requires only a linear amount of advice if we restrict the class of input instances to graphs with the maximum degree bounded by a constant. However, there are sparse graphs (i.e. graphs with $O(n)$ edges) with a high maximum degree, for example a star. To establish a tighter upper bound on sparse graphs in general, we devise a different algorithm using $O(m)$ bits of advice.

Again, we choose an arbitrary optimal walk and describe it to the agent using the advice. This time, however, the agent may not be able to replicate the walk exactly, although it will still find a walk of the minimum length.

Lemma 4. *Let G be an unweighted graph and ω an optimal exploration walk of G. Then no directed arc $x \to y$ occurs in ω more than once.*

Proof. We prove the claim by contradiction. As ω is a closed walk with at least two occurrences of some directed arc $x \to y$, we may partition it into these two occurrences and the two remaining open walks α and β. Both α and β lead from vertex y to vertex x. However, by reversing one of them and concatenating it with the other we create a shorter closed walk passing through the same set of vertices as ω. This contradicts the assumption that ω is an optimal exploration walk. □

This lemma implies that every directed arc of G is either not used in the chosen optimal walk or is used exactly once. We write one advice bit per arc to indicate which possibility occurred, i.e. $2m$ bits altogether. With this advice, the agent will ignore the arcs not belonging to the optimal solution. The remaining arcs form a directed Eulerian graph, because every vertex has an equal in-degree and out-degree. Moreover, the graph is connected.

We define the *exit arc* of a vertex as the last arc in the optimal walk leading from that vertex. The starting node is an exception: No arc leading from it is marked as an exit arc. To brief the agent about exit arcs, we use one bit per arc in the optimal walk, i.e. at most $2(n-1)$ bits (by lemma 3). Altogether, the agent obtains $O(m)$ advice bits.

The agent follows a simple algorithm: In each step, it travels through any outgoing arc that has not been used yet (with the exception of the ignored arcs). However, it chooses the exit arc only when all the other outgoing arcs from the current vertex were already traversed.

We shall prove that the agent as we defined it travels through every arc of the optimal solution exactly once and finishes at the starting vertex, hence it finds an exploration walk with the optimal length.

Firstly, let us note that at the moment the agent obeying its algorithm cannot move any more, it is located at the starting node. If it were at a different node, it would have entered and left this node the same number of times, which is not possible.

Secondly, let us assume that the algorithm finished its walk, but it did not travel through some exit arc $x \to y$. As the agent did not use all arcs leading to y, it also did not travel through the exit arc of y. By repeating this argument we obtain a sequence of exit arcs, none of them being used by the agent. From the definition of the exit arc, no vertex occurs in this sequence more than once, therefore it ends in the starting node. However, this contradicts our assumption: If there is an unused arc leading to the starting node, there must be one leading out of it and the agent's walk is not finished.

In conclusion, the agent explores a given graph in the optimal number of steps while reading $O(m)$ advice bits.

The advice complexity could be improved: As at most $2(n-1)$ arcs out of $2m$ occur in the optimal walk, we could encode them in less than $2m$ bits. To be precise, at most

$$\left\lceil \lg \binom{2m}{2(n-1)} \right\rceil$$

bits are sufficient if $2(n-1) \leq m$. For instance, this bound becomes $O(n \lg n)$ on dense graphs.

3.1.3 Lower Bound on Advice

For any $\Delta \geq 2$, we describe a graph W_Δ on 2Δ vertices with the maximum degree Δ. We consider a slightly altered problem, in which an agent explores a graph starting in a node s and ending in a (possibly different) node t. For that setting, we show that to yield an optimal solution, any agent requires at least $\lceil \lg(\Delta!) \rceil$ advice bits. Then we combine an arbitrary number of copies of W_Δ to obtain an input instance of our original problem. Finally, we show a lower bound on advice for an optimal agent on that particular instance.

Graph W_Δ consists of 2Δ vertices labelled $0, 1, 2, \ldots, 2\Delta-1$. For any vertices i and j $(i < j)$, there is an edge connecting them iff at least one of the following conditions is satisfied:

- $j = i + 1$
- i is even and j is odd

Clearly, no vertex has more than Δ neighbours.

Lemma 5. *For any $\Delta \geq 2$, the path $(0, 1, 2, \ldots, 2\Delta-1)$ is the only Hamiltonian path in W_Δ connecting vertices 0 and $2\Delta - 1$.*

Proof. We show by induction that for any i satisfying $1 \leq i \leq \Delta - 1$, every Hamiltonian path connecting vertices 0 and $2\Delta - 1$ contains edges $(0, 1)$, $(1, 2)$, $(2, 3)$, …, $(2i - 1, 2i)$.

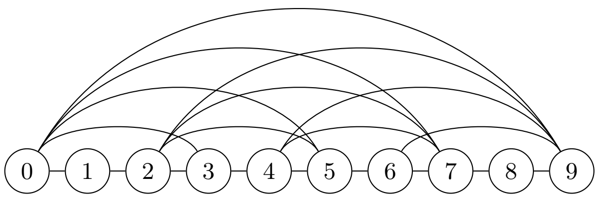

Figure 3.1: Graph W_5

For $i = 1$, the statement holds, as any such Hamiltonian path must comprise both edges incident to vertex 1, i.e. edges $(0, 1)$ and $(1, 2)$.

Let $i > 1$. Vertex $2i-1$ is odd, hence it is adjacent to vertices $0, 2, 4, \dots, 2i-2$ and $2i$. By the induction hypothesis, vertices $0, 2, 4, \dots, 2i-4$ cannot be connected with $2i-1$ in any such Hamiltonian path. Therefore, edges $(2i-2, 2i-1)$ and $(2i-1, 2i)$ must be used.

Finally, by the same reasoning, edge $(2\Delta-2, 2\Delta-1)$ must be used in the Hamiltonian path. □

To simplify our analysis, we use numbers $0, 1, 2, \dots, 2\Delta-1$ to identify vertices of W_Δ. However, if these were revealed to the agent, it could easily find the optimal walk. Therefore, we label the vertices with a permutation of $0, 1, 2, \dots, 2\Delta-1$ and show the agent only the labels.

The following lemma states that after this obfuscation, an agent cannot find the optimal walk without receiving a considerable amount of advice.

Lemma 6. *For any $\Delta \geq 2$ and agent A reading strictly fewer than $\lceil \lg(\Delta!) \rceil$ advice bits when solving an instance of W_Δ, there is a permutation of node labels for which A does not find the Hamiltonian path in W_Δ connecting vertices 0 and $2\Delta-1$.*

Proof. We permute the labels of odd and even vertices separately, so we get $(\Delta!)^2$ different inputs. As agent A can receive at most $2^{\lceil \lg(\Delta!) \rceil - 1} < \Delta!$ different advice strings, there are at least $\Delta! + 1$ inputs with the same advice. Among these, there exist at least two inputs such that all even vertices are labelled consistently in both of them; an analogous statement holds for odd vertices (with a different pair of inputs).

Assume that the agent starts in vertex 0. Let W_Δ^1 and W_Δ^2 be two inputs with the same advice string, identically labelled even vertices, and unequally labelled odd vertices. Let us denote by v the first odd vertex on the Hamiltonian path connecting nodes 0 and $2\Delta-1$ that is labelled differently in W_Δ^1 and W_Δ^2. If A follows the Hamiltonian path all the way from 0 to $v-1$, the inputs W_Δ^1 and W_Δ^2 are indistinguishable for it in the moment it reaches $v-1$. In particular, the set of neighbours' labels in that moment is identical in W_Δ^1 and W_Δ^2. Agent A deterministically chooses a label of adjacent node and moves to it. As the labels of v in W_Δ^1 and W_Δ^2 differ, at least in one of the inputs the agent fails to follow the optimal path.

If the agent starts in vertex $2\Delta-1$, the situation is symmetric – we simply swap the roles of odd and even vertices. □

Theorem 7. *For any $\Delta \geq 3$, $k \geq 1$ and optimal agent A, there is an unweighted graph G on $2k(\Delta - 1) + 1$ vertices with the maximum degree Δ, such that A exploring it reads at least $\lceil k \lg(\Delta - 1)! \rceil$ advice bits.*

Proof. We join k copies of $W_{\Delta-1}$ and a starting node s to create graph G. Apart from the edges in the copies, we add an edge connecting vertex $2(\Delta - 1) - 1$ of i-th copy and vertex 0 of $(i + 1)$-th copy for every $1 \leq i < k$. Moreover, we connect vertex s with vertex 0 of first copy and vertex $2(\Delta - 1) - 1$ of k-th copy. Clearly, graph G has $k \cdot 2(\Delta - 1) + 1$ vertices and the maximum degree of Δ.

Every part of G (node s and the copies of $W_{\Delta-1}$) is connected with exactly two other in a circular manner, hence any Hamiltonian cycle in G consists of Hamiltonian paths in the copies of $W_{\Delta-1}$ and the connecting edges. According to Lemma 5, there is a unique Hamiltonian path connecting vertices 0 and $2(\Delta - 1) - 1$ of any $W_{\Delta-1}$. Therefore, there is a unique Hamiltonian cycle in G.

We permute the labels of odd and even vertices in each copy of $W_{\Delta-1}$ separately, so we get $((\Delta - 1)!)^{2k}$ different inputs. Following the same reasoning as in the proof of Lemma 6, any optimal agent A must read at least $\lceil k \lg(\Delta - 1)! \rceil$ bits of advice. $\square$

The previous theorem implies that any optimal agent exploring an unweighted graph on n vertices with the maximum degree Δ requires at least

$$\left\lceil \frac{n-1}{2(\Delta - 1)} \lg(\Delta - 1)! \right\rceil = \Omega(n \lg \Delta)$$

advice bits. In particular, taking $k = 1$ we get a lower bound on advice for graphs of unbounded degree: $\Omega(n \lg n)$.

3.2 Competitive Ratio $1.564 - \varepsilon$

In this section, we establish that any agent achieving competitive ratio less than $1 + \frac{1}{\ln 16 - 1} \approx 1.564$ on unweighted graphs must read $\Omega(n)$ bits of advice. As the proof is quite long, we provide a brief outline of it first.

3.2.1 Outline

We will focus on a class of unweighted graphs called sun graphs, which are particularly difficult to explore efficiently. A sun graph is essentially a cycle with paths called rays connected to it (see fig. 3.2 for an example). An agent traversing the cycle of a sun graphs faces a sequence of challenges: Every time it enters a junction of the cycle and a ray, it must choose which path will it follow first. It detects whether the chosen path lies on the ray or on a segment of the cycle only after several steps. If it is the segment, the agent must eventually return to visit the ray, which negatively impacts its performance. Therefore, an efficient agent has to receive a bit of advice for at least a fraction of all junctions in order to take the right decisions in them.

Firstly, we show that the total cost of an agent's exploration walk can be estimated by considering only its behaviour on individual challenges. Secondly, we look at how an agent acts on problem instances that are similar to each other, thus establishing an upper bound on the number of instances for which an agent performs satisfactorily well. This leads to a linear lower bound on advice for such an agent. However, the bound applies only to a subset of sun graphs parameterized by a ratio of the number of rays of different lengths. Finally, we find parameters that maximize the competitive ratio for which we can obtain a linear lower bound on advice.

3.2.2 Elementary Definitions

Definition 1. A *sun graph* is an undirected graph that consists of a cycle and a number of non-trivial paths called *rays*. Exactly one vertex of each ray lies on the cycle, and each vertex of the cycle lies on at most one ray.

A vertex lying both on the cycle and on a ray is called a *junction*. A path between two junctions that does not pass through any other junction is called a *sun segment*.

We will show that sun graphs are quite difficult to explore efficiently, i.e. no agent with sublinear advice can achieve competitive ratio less than a certain constant. However, in our analysis we will be considering only a subset of sun graphs that is defined below. The subset is parameterized by two numbers k and ℓ. The value of the minimum achievable competitive ratio on the graphs in the subset depends on these two parameters; we obtain the main result of this section by taking the limit as they approach infinity.

Definition 2. For any $k \geq 1, \ell \geq 3$, a sun graph is of *type* k and *order* ℓ if it satisfies all the following conditions:

- it contains exactly ℓ rays,
- no ray is longer than k,
- one of the sun segments has length $2k(2k+1)$, all the others have length k.

We will refer to the longer sun segment as the *penalty segment*. The starting node s can be chosen arbitrarily from the inner vertices of the penalty segment.

The set of sun graphs of type k and order ℓ is denoted by $\mathcal{S}_{k,\ell}$.

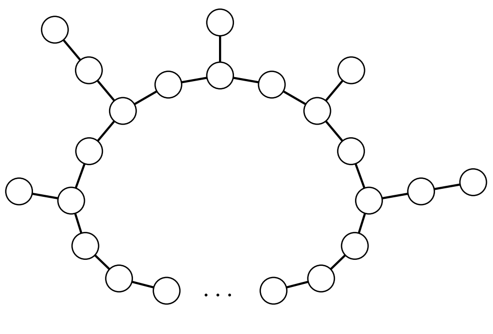

Figure 3.2: A sun graph of type 2 and order 5. The penalty segment is depicted only partially; it consists of 21 vertices, including the junctions on its endpoints.

Lemma 8. *Let $G \in \mathcal{S}_{k,\ell}$ be a sun graph with r_i rays of length i. Then G has*

$$n = k(\ell + 4k + 1) + \sum_{i=1}^{k} ir_i$$

vertices and the optimal cost of exploring it offline is

$$C(opt(G)) = n + \sum_{i=1}^{k} ir_i$$

Proof. The cycle consists of $k(\ell - 1) + 2k(2k + 1)$ vertices, as there are $\ell - 1$ segments of length k and the penalty segment. A ray of length i contributes another i vertices to the total count.

Any exploration walk must enter each vertex at least once. When returning from the last vertex of a ray of length i, the walk must visit at least i vertices for the second time. Clearly, there is a walk meeting this cost. $\square$

We are going to analyse the behaviour of a fixed deterministic agent on graphs in $\mathcal{S}_{k,\ell}$ and to estimate the cost it incurs. To simplify the analysis, we partition the agent's walk into several stages, each of them starting when the agent visits a junction for the first time. In such a situation the agent does not know which one of the two untraversed edges incident with the junction lies on the ray. It would prefer to visit all vertices on the ray first and then continue along the cycle – but it may also visit the next junction before finishing the ray, in which case it needs to return and visit the ray vertices it missed (not necessarily right away).

To formalize this problem the agent faces, we define a concept of a *challenge.*

Definition 3. A *challenge* is an induced subgraph of a sun graph $G \in \mathcal{S}_{k,\ell}$ which consists of a non-penalty sun segment and a ray incident to that segment. The *entrance* of a challenge is the junction lying on the ray; the *exit* is the other junction. The *length* of a challenge is the length of the ray.

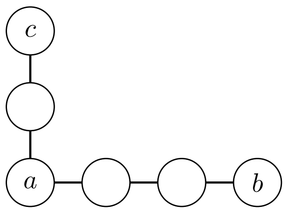

Figure 3.3: A challenge of length 2 from a sun graph of type 3. Vertices a, b are junctions; a is the entrance and b is the exit. Vertex c is the last vertex of the ray.

Let us note that a challenge is not uniquely determined by its entrance. A junction can be visited for the first time using either of the two cycle edges incident to it, which gives rise to two potential challenges with entrance in that junction. Therefore, the set of actually faced challenges depends on the behaviour of the analysed agent.

Apart from a couple of degenerated challenges at the final stages of the agent's walk, an important invariant holds: Right before entering a challenge, the agent has not seen any vertex of it except for the entrance. Hence, the agent can neither distinguish which direction from the entrance leads to the exit, nor it can ascertain the length of the challenge. The situation changes when the agent reaches the exit or the last vertex on the ray. At such a moment, the agent has *completed* the challenge.

Let us continue by defining a *response* to a challenge, a concise notation of the agent's progress on the challenge.

Definition 4. A *response* to a challenge is a string of letters `R` and `C` denoting the order in which an agent visits new vertices of the challenge. An occurrence of letter `R` (`C`) corresponds with entering the next yet unvisited vertex on the ray (on the cycle). If the challenge is of length i and lies in a sun graph of type k, the response must end immediately after the i-th occurrence of letter `R` or the k-th occurrence of letter `C`, whichever comes first.[1]

Definition 5. The *cost* of a response ρ is the minimum cost an agent starting at the entrance of a challenge can incur by visiting vertices in the order specified by ρ, then visiting the rest of vertices, and ending at the exit.

We will denote the cost of a response ρ to a challenge of length i lying in a sun graph of type k by $\text{cost}_k(\rho, i)$.

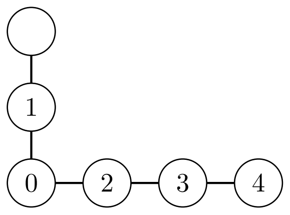

Figure 3.4: A challenge of length 2 from a sun graph of type 3, and the order in which an agent applying response `RCCC` discovers its vertices. The cost of this response is 15.

Lemma 9. *The cost of any response ρ to a challenge in a sun graph of type k is at most $2k(2k + 1)$.*

Proof. Starting at the entrance, an agent needs to move to at most $2k + 1$ vertices: k vertices of the segment, at most k vertices of the ray, and finally to the exit. The distance between any two vertices of the challenge is at most $2k$, therefore the total cost does not exceed $2k(2k + 1)$. □

Lemma 10. *Let ρ be a response to a challenge of length i lying in a sun graph of type k. If ρ ends with letter `C`, then the minimum cost an agent starting at the entrance of the challenge can incur by visiting vertices in the order specified by ρ is $\text{cost}_k(\rho, i) - 2k - 2i$.*

[1]This requirement ensures that by visiting the last vertex in the response the agent completes the challenge.

Proof. If response ρ ends with letter `C`, an agent applying ρ completes the challenge by visiting the exit and leaves the last vertex of the ray unvisited. Let us denote the number of steps it has taken so far by t. After that, the agent moves to the last vertex of the ray and then back to the exit, which requires at least $2k + 2i$ steps. By the definition we have $\text{cost}_k(\rho, i) \leq t + 2k + 2i$ and therefore $t \geq \text{cost}_k(\rho, i) - 2k - 2i$. Clearly, there is a walk meeting this bound. $\square$

3.2.3 Estimating the Total Cost from Responses

We cannot require that the agent remains in a challenge until it visits all vertices, hence our definition of the cost of a response seems too restrictive. However, we will show in the next lemma how the cost as we defined it can be used to bound the total length of the agent's walk.

To keep our analysis simple, we do not want the agent to participate in two opened challenges at the same time. This may happen when the agent enters a challenge and before reaching the exit or the last vertex of the ray, it travels back through the starting vertex to the other side of the cycle and enters a new challenge there. In such a situation, we say the agent has *given up* the first challenge and it will never be completed. Let us note that the agent pays for giving up by traversing the penalty segment.

As we mentioned earlier, in the final stages of the agent's exploration, there are a couple of corrupted challenges for which we cannot ensure that the agent has not seen any vertex of them before entering.[2] For this reason, we consider only the first $\ell - 2$ challenges in our counts (and disregard the remaining two).

Lemma 11. *Let A be a deterministic agent and $G \in \mathcal{S}_{k,\ell}$ a sun graph with r_i rays of length i. Let us denote the number of challenges A gives up while exploring G by g, and the number of completed challenges of length i for which A applies response ρ by $e_{i,\rho}$. Then*

$$C(A(G)) \geq \min\left(\sum_{i,\rho} cost_k(\rho, i)\, e_{i,\rho} + 2k(2k+1)g + k, C(opt(G)) + k(\ell - 3)\right)$$

Proof. If agent A does not travel through every edge of G, then it avoids an edge of the cycle and therefore effectively explores a tree. This cannot be done in less than $2(n-1)$ steps (where n is the number of nodes of G). From lemma 8 we obtain the following bound:

$$\begin{aligned} C(A(G)) &\geq 2(n-1) \\ &\geq C(\text{opt}(G)) + k(\ell + 4k + 1) - 2 \\ &\geq C(\text{opt}(G)) + k(\ell - 3) \end{aligned}$$

In the rest of the proof we will look at the case when the agent travels through every edge at least once.

We partition the agent's walk into stages: the j-th one starts when A enters j-th junction for the first time. We will refer to that junction as the *current*

[2] Let us consider the last segment to be visited. The agent may have seen both its junctions, because the adjacent segments are already visited. Therefore, neither of the junctions is eligible to act as the exit of a challenge.

junction of the stage. The *current path* is the visited path between the starting node and the current junction.

Each of stages $1, 2, \ldots, \ell-2$ involves exploration of an uncorrupted challenge; with the rest of the walk we will deal separately. We are going to follow agent A's progress and maintain several counts (each of them applies to the moment of entering stage j):

- $t^{(j)}$ – the number of steps A has taken,
- $a^{(j)}$ – the number of junctions on the current path,
- $b^{(j)}$ – the number of remaining visited junctions (i.e. on the other side of the cycle). Clearly, $a^{(j)} + b^{(j)} = j$.
- $g^{(j)}$ – the number of challenges A has given up,
- $e^{(j)}_{i,\rho}$ – the number of completed challenges of length i for which A has applied response ρ,
- $f^{(j)}_i$ – the number of *forgotten* rays of length i. A *forgotten* ray is the ray in a challenge that has been completed or given up, but the last vertex of the ray has not yet been visited.
- $c^{(j)}$ – the number of forgotten rays on the current path. As the ray incident to the current junction cannot be forgotten at the beginning of stage j, we have $c^{(j)} \leq a^{(j)} - 1$.

Let us prove by induction that the following invariant holds for all $j \in \{1, 2, \ldots, \ell - 1\}$:

$$t^{(j)} + \sum_i 2if^{(j)}_i + k\left(b^{(j)} + 2c^{(j)}\right) \geq \sum_{i,\rho} \mathrm{cost}_k(\rho, i)\, e^{(j)}_{i,\rho} + 2k(2k+1)g^{(j)}$$

The base case ($j = 1$): All counters on the right-hand side of the invariant are equal to zero, as the agent has just come to its first junction. The left-hand side is clearly non-negative, thus the inequality holds.

The inductive step: Let us assume that the invariant holds for stage $j - 1$, we shall prove it for stage j, where $2 \leq j \leq \ell - 1$. During stage $j - 1$, the agent was facing a challenge H of length i; let us consider the possible outcomes of it.

If the agent initiates stage j by visiting the exit of H, then challenge H was definitely completed and we can tell what response ρ did the agent apply. The appropriate counter $e^{(j-1)}_{i,\rho}$ increments by one, which is compensated by the increase of counters $t^{(j-1)}$, $f^{(j-1)}_i$, and $c^{(j-1)}$: If the agent visited all vertices of the challenge (i.e. ρ ends with `R`), it took at least $\mathrm{cost}_k(\rho, i)$ steps; otherwise (ρ ends with `C`) it took at least $\mathrm{cost}_k(\rho, i) - 2k - 2i$ steps (by lemma 10) and incurred one forgotten ray of length i. Meanwhile, the agent could visit some forgotten rays from earlier stages, thus decreasing counters $f^{(j-1)}_{i'}$ and $c^{(j-1)}$. However, to visit a forgotten ray of length i' and return back, agent A took at least $2k + 2i'$ steps.

In the second case, the agent initiates stage j by visiting the junction on the other side of the cycle. Challenge H is either completed or given up; in both cases the increase of the right-hand side is compensated by agent A's traversal through the penalty segment (by lemma 9, $\mathrm{cost}_k(\rho, i) \leq 2k(2k+1)$). Let us now consider other changes on the left-hand side of the inequality. Agent A moved from the $a^{(j-1)}$-th junction on the current path to the $(b^{(j-1)}+1)$-th junction on the other side, which took at least $k\left(a^{(j-1)} + b^{(j-1)} - 1\right)$ steps (excluding the penalty segment). Term $kb^{(j-1)}$ changed to $kb^{(j)} = ka^{(j-1)}$. Counter $c^{(j-1)}$ might have decreased, but not more than by its value, which we can bound from above by $a^{(j-1)} - 1$. Altogether these changes add up to an increase of at least

$$k\left(\left(a^{(j-1)} + b^{(j-1)} - 1\right) + \left(a^{(j-1)} - b^{(j-1)}\right) - 2\left(a^{(j-1)} - 1\right)\right) = k$$

The agent could also visit some forgotten rays on its way, but the decrease of counters $f_{i'}^{(j-1)}$ is compensated by additional steps it took on those rays.

That concludes the proof by induction, and we have the invariant holding at the beginning of stage $\ell - 1$. Now, the agent needs to visit the last junction, all forgotten rays, and return back to the starting node. While doing so, if it travels through every edge on the current path, we get a simple lower bound on the length of its entire exploration walk: At least $k\left(a^{(\ell-1)} - 1 + b^{(\ell-1)} - 1\right) = k(\ell-3)$ edges of the cycle and all edges of the rays were traversed at least twice.[3] Therefore, agent A took at least

$$k(\ell + 4k + 1) + k(\ell - 3) + \sum_{i=1}^{k} 2ir_i = C(\mathrm{opt}(G)) + k(\ell - 3)$$

steps.

We are left with the second case: While completing its walk, the agent does not travel through some edge on the current path. At the beginning of stage $\ell - 1$, agent A has taken $t^{(\ell-1)}$ steps. Since then it needs at least $\sum_i 2if_i^{(\ell-1)}$ steps to visit forgotten rays, $2kc^{(\ell-1)}$ steps to get to those rays on the current path and back, and $2k + k\left(b^{(\ell-1)} - 1\right)$ steps to return to the starting node. Assuming $e_{i,\rho}^{(\ell-1)} = e_{i,\rho}$, $g^{(\ell-1)} = g$, and that the invariant holds for $j = \ell - 1$, we obtain the following bound:

$$C(A(G)) \geq \sum_{i,\rho} \mathrm{cost}_k(\rho, i)\, e_{i,\rho} + 2k(2k+1)g + k$$

We showed that in any case, the cost of the agent's walk is at least $C(\mathrm{opt}(G)) + k(\ell - 3)$, or at least $\sum_{i,\rho} \mathrm{cost}_k(\rho, i)\, e_{i,\rho} + 2k(2k+1)g + k$. □

3.2.4 Strategies

The previous lemma enables us to disregard the actual edge traversals an agent makes and consider only its responses to challenges. In the following part, we are going to look at how an agent performs on a whole class of graphs.

[3]The claim for the $k\left(a^{(\ell-1)} - 1\right)$ edges follows from the previous assumption. As for the $k\left(b^{(\ell-1)} - 1\right)$ edges, the agent must have traversed them twice in order to move from that side of the cycle to the current one. The claim also holds in the special case of $b^{(\ell-1)} = 0$ when the agent did not visit any junctions on the other side.

A response to a challenge is an observable behaviour of the analysed agent. However, the agent itself cannot be sure what its response is going to be, because it also depends on the challenge's characteristics unknown to the agent. We define a set of similar challenges such that the agent cannot distinguish which member of the set it is exploring until it completes the challenge. For such a fixed set, the agent seems to have a common *strategy* from which we can infer its response to any challenge in the set.

Definition 6. A *family of challenges* is a set $\{H_1, H_2, \ldots, H_k\}$ such that for any $i \in \{1, 2, \ldots, k\}$, H_i is a challenge of length i lying in a graph of type k. Moreover, for any pair of challenges H_i, H_j $(i < j)$ in a family, the labels of corresponding vertices are identical (i.e. the sequence of labels on the path from the entrance to the exit of one challenge is equal to that of the other challenge; the same holds for the labels of the first i vertices on the rays).

Definition 7. The *transpose* of a family of challenges $F = \{H_1, H_2, \ldots, H_k\}$ is a family of challenges $F^\intercal = \{H_1^\intercal, H_2^\intercal, \ldots, H_k^\intercal\}$ such that the sequence of labels on the segment of H_k is equal to the sequence of labels on the ray of $H_k^\intercal$, and vice versa (in all cases the sequence starts with the label of the entrance).

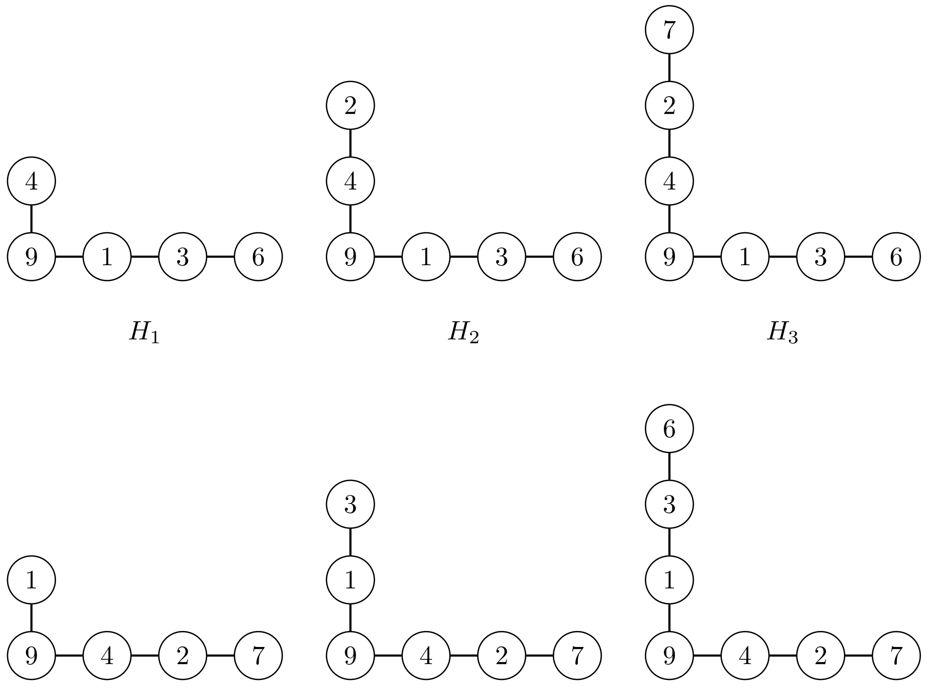

Figure 3.5: A family of challenges and its transpose.

We now define a concept of a *strategy* that captures the behaviour of an agent on all challenges in a family and its transpose.

Definition 8. A *strategy* for a family of challenges $F = \{H_1, H_2, \ldots, H_k\}$ and its transpose $F^\intercal = \{H_1^\intercal, H_2^\intercal, \ldots, H_k^\intercal\}$ is a string of length $2k$, consisting of k digits `1` and k digits `2`, and starting with `1`. An agent applying strategy σ on a challenge $H \in F \cup F^\intercal$ visits new vertices of H in the order specified by σ until it completes the challenge. In the beginning, the agent chooses one of the two paths leading from the entrance and visits its first vertex, thus declaring it the first path. An occurrence of digit `1` (`2`) then corresponds with entering the next yet unvisited vertex on the first (second) path.

We can find out the agent's strategy directly from its response to challenge H_k or $H_k^\intercal$. Conversely, from a strategy and the agent's first move (whether it visits a vertex on the cycle or a vertex on the ray) we are able to determine its response to any challenge $H \in F \cup F^\intercal$.[4] For instance, if an agent exploring H_i visits the first vertex of the ray in its first move, we obtain its response by substituting `R` for every `1`, `C` for every `2`, and taking the longest prefix containing at most i occurrences of `R` and at most k occurrences of `C`.

Let us denote the mapping from strategies to responses by $\text{response}_k(\sigma, i, \delta)$, where k is the type of a sun graph, σ is a strategy, i is the length of a challenge, and $\delta \in \{\mathtt{C}, \mathtt{R}\}$ is the first move of an agent. To shorten our notation, we define $\text{cost}_k(\sigma, i, \delta)$ as $\text{cost}_k(\text{response}_k(\sigma, i, \delta), i)$.

3.2.5 Basic Lower Bound on Advice

In the following lemma, we obtain a lower bound on advice for efficient agents exploring graphs from a certain subset of $\mathcal{S}_{k,\ell}$. For now, the bound is expressed in terms of a complicated function f (defined in the lemma below).

Lemma 12. *Let $\mathcal{G} \subseteq \mathcal{S}_{k,\ell}$ be the set of sun graphs with r_i rays of length i (for some fixed values of r_i satisfying $\sum_{i=1}^k r_i = \ell$). Let A be an agent with advice such that for any graph $G \in \mathcal{G}$, agent A reads at most b bits of advice and takes at most t steps while exploring G, where $t < C(opt(G)) + k(\ell - 3)$.*[5] *Then*

$$b \geq \ell - 2 + \lg \binom{\ell}{r_1, r_2, \ldots, r_k} - \lg f((r_1, r_2, \ldots, r_k), t + 8k^2 + 3k)$$

where

$$\binom{\ell}{r_1, r_2, \ldots, r_k} = \frac{\ell!}{r_1! r_2! \cdots r_k!}$$

is the multinomial coefficient, and $f : \mathbb{Z}^k \times \mathbb{R} \to \mathbb{Z}$ is the function defined as follows: For any $\vec{x} = (x_1, x_2, \ldots, x_k), y$ such that $(x_1, x_2, \ldots, x_k, y \geq 0) \wedge (\exists i\ x_i > 0)$,

$$f(\vec{x}, y) = \max_\sigma \sum_{i=1}^{k} \big(f(\vec{x} - \vec{\tau_i}, y - cost_k(\sigma, i, \mathtt{C})) + f(\vec{x} - \vec{\tau_i}, y - cost_k(\sigma, i, \mathtt{R})) \big)$$

where $\vec{\tau_i}$ stands for the standard basis vector with a one at the i-th position and zeros elsewhere. For any $y \geq 0$,

$$f(\vec{0}, y) = 1$$

[4]Let us note that an agent makes the same first move for all challenges in a family, and the opposite first move for all challenges in the family's transpose.

[5]All graphs in $\mathcal{G}$ have the same optimal exploration cost.

In all other cases, i.e. $(\exists i \ x_i < 0) \vee (y < 0)$,

$$f(\vec{x}, y) = 0$$

Proof. An instance of the graph exploration problem is a graph G together with a labelling of its vertices and a designated starting node. As we are now primarily interested in labels, for any $G \in \mathcal{G}$ we place the start in the same vertex on the penalty segment (we can choose arbitrary one).

Let us introduce an auxiliary concept of a *pseudo-instance*, which is an extension of an instance. A regular instance specifies a sequence of i labels for the vertices of a ray of length i. A pseudo-instance always specifies a sequence of k labels for a ray, regardless of its length (although only the first i labels are actually used if the ray has length i; we could say the rest are labels of pseudo-vertices). Clearly, we obtain an instance from a pseudo-instance by omitting unused labels.

A pseudo-instance of a graph in $\mathcal{S}_{k,\ell}$ contains $2k\ell + 4k^2 + k$ labels. To the vertices of the penalty segment we will always assign the same labels, so there are $2k\ell - k - 1$ labels left and $(2k\ell - k - 1)!$ ways how to assign them. We consider only pseudo-instances with r_i rays of length i, hence there are $\binom{\ell}{r_1, r_2, \ldots, r_k}$ ways how to choose the lengths of rays. Altogether we have a set $\mathcal{I}$ of

$$(2k\ell - k - 1)!\binom{\ell}{r_1, r_2, \ldots, r_k}$$

pseudo-instances.

Agent A reads at most b bits of advice, so it can receive at most 2^b different advice strings. Therefore, there is a subset $\mathcal{I}' \subseteq \mathcal{I}$ such that A receives the same advice for any pseudo-instance from $\mathcal{I}'$, and

$$|\mathcal{I}'| \geq \frac{(2k\ell - k - 1)!\binom{\ell}{r_1, r_2, \ldots, r_k}}{2^b} \tag{3.1}$$

Now, let us categorize pseudo-instances in $\mathcal{I}'$ following the agent's behaviour on them.[6] Agent A starts somewhere in the penalty segment, and continues in a unified way on all pseudo-instances until it enters the first challenge.

Firstly, we are going to partition $\mathcal{I}'$ into several groups according to the labels in this challenge: k vertices on the segment and k (pseudo-)vertices on the ray have not been assigned fixed labels yet, and we have $2k\ell - k - 1$ labels available. Thus, there are

$$\frac{(2k\ell - k - 1)^{\underline{2k}}}{2}$$

ways how to label the challenge,[7] if we disregard the information about which path is the ray and which is the segment. We create a separate group for each such labelling. Let us note that the first challenges of instances from the same group belong to a family or its transpose, so the agent applies the same strategy for each of them.

[6]Let us note that the agent is fully deterministic on pseudo-instances in $\mathcal{I}'$, as it receives the same advice for all of them.

[7]$x^{\underline{k}}$ is the k-th falling factorial power of x; $x^{\underline{k}} = x(x-1)(x-2)\cdots(x-k+1)$.

Secondly, we subdivide each group from previous step into $2k$ groups according to the length of the challenge and the agent's first move (`C` or `R`). Pseudo-instances in such a group are indistinguishable to the agent until it enters the second challenge, so we can repeat the whole process.

In general, we have a set of pseudo-instances that are indistinguishable until agent A visits the j-th challenge. At first, we partition them into

$$\frac{(2k(\ell - j) + k - 1)^{\underline{2k}}}{2}$$

groups according to the labels in the j-th challenge; then we subdivide each group into $2k$ parts according to the length of the j-th challenge and the agent's first move on it.

This process is repeated for $\ell - 2$ times.[8] Each of the groups obtained by the last iteration contains at most $2 \cdot (3k-1)!$ pseudo-instances, because there are only $3k - 1$ (pseudo-)vertices without fixed labels left and at most two different lengths of rays to choose from. The whole procedure creates a categorization tree; in its leaves we have individual groups from the last iteration and its internal nodes represent subdivisions.

Let us estimate the maximum possible size of $\mathcal{I}'$ given the constraint t on the length of agent A's exploration walk. From lemma 11 we have the following bound on the cost $C(A(G))$:

$$C(A(G)) \geq \min\left(\sum_{i,\rho} \mathrm{cost}_k(\rho, i)\, e_{i,\rho} + 2k(2k+1)g + k, C(\mathrm{opt}(G)) + k(\ell - 3)\right)$$

As $t < C(\mathrm{opt}(G)) + k(\ell - 3)$, for any $G \in \mathcal{I}'$ it must hold

$$t \geq C(A(G)) \geq \sum_{i,\rho} \mathrm{cost}_k(\rho, i)\, e_{i,\rho} + 2k(2k+1)g + k$$

Therefore, pseudo-instances from $\mathcal{I}'$ may end up only in certain leaves of our categorization tree: The sum of response costs on the path to such a leaf may not exceed $t - k$. As the cost of giving up a challenge is greater or equal than the cost of any response (by lemma 9), we will ignore the possibility of giving up in our estimation.

The labels of challenges are irrelevant when summing response costs,[9] and the number of ways how to assign labels in the j-th challenge depends only on the value of j. Consequently, we can count ways of labelling and sequences of responses separately, and multiply the counts in the end.

There are

$$\frac{(2k(\ell - j) + k - 1)^{\underline{2k}}}{2}$$

ways how to assign labels in the j-th challenge for $j \in \{1, 2, \ldots, \ell - 2\}$, and $(3k-1)!$ labellings of the remaining vertices. The total number of possibilities

[8] The $(\ell - 1)$-th challenge may be corrupted, i.e. its exit may have been seen by the agent.

[9] They might be relevant to the agent, as it can choose different strategies depending on the labels. However, in our analysis we will assume the agent chooses the optimal strategy (i.e. the strategy maximizing the number of pseudo-instances in given subtree), which is independent of the labels.

is

$$(3k-1)!\prod_{j=1}^{\ell-2}\frac{(2k(\ell-j)+k-1)^{\underline{2k}}}{2}=\frac{(2k\ell-k-1)!}{2^{\ell-2}}$$

Now, let us estimate the maximum possible number of sequences of responses. To simplify the analysis, we append to each sequence such two responses that the sequence contains exactly r_i responses to challenges of length i.[10] Furthermore, we increase the limit on the total cost by $4k(2k+1)$ as a compensation for the additional responses. We are going to show that the maximum number of sequences containing x_i responses to challenges of length i with the sum of response costs not exceeding y is equal to $f((x_1,x_2,\ldots,x_k),y)$, where f is the function defined in the statement of this lemma. The proof is by induction on the sum of x_i's.

If some of the arguments are negative, no such sequence of responses exists. If for all i, $x_i=0$, only the empty sequence satisfies the conditions (assuming $y\geq 0$), therefore $f(\vec{0},y)=1$ holds. Otherwise, any valid sequence starts with a response to a challenge from some family F or its transpose $F^{\intercal}$. If an agent applies strategy σ, we can determine the cost of response to any challenge in $F\cup F^{\intercal}$, and also the number of sequences starting with that response. For example, if the challenge is of length i and the agent's first move is C, there are $f(\vec{x}-\vec{\tau}_i,y-\mathrm{cost}_k(\sigma,i,\mathtt{C}))$ such sequences. Altogether the maximum obtainable number of sequences when applying strategy σ equals

$$\sum_{i=1}^{k}\bigl(f(\vec{x}-\vec{\tau}_i,y-\mathrm{cost}_k(\sigma,i,\mathtt{C}))+f(\vec{x}-\vec{\tau}_i,y-\mathrm{cost}_k(\sigma,i,\mathtt{R}))\bigr)$$

We choose a strategy which maximizes this sum.

Finally, we obtain an upper bound on the size of $\mathcal{I}'$:

$$|\mathcal{I}'|\leq\frac{(2k\ell-k-1)!}{2^{\ell-2}}\cdot f((r_1,r_2,\ldots,r_k),t+8k^2+3k)\tag{3.2}$$

By combining (3.1) and (3.2) we get

$$\frac{(2k\ell-k-1)!}{2^{\ell-2}}\cdot f((r_1,r_2,\ldots,r_k),t+8k^2+3k)\geq\frac{(2k\ell-k-1)!\binom{\ell}{r_1,r_2,\ldots,r_k}}{2^b}$$

$$\begin{aligned}2^b&\geq\frac{2^{\ell-2}\binom{\ell}{r_1,r_2,\ldots,r_k}}{f((r_1,r_2,\ldots,r_k),t+8k^2+3k)}\\ b&\geq\ell-2+\lg\binom{\ell}{r_1,r_2,\ldots,r_k}-\lg f((r_1,r_2,\ldots,r_k),t+8k^2+3k)\qquad\square\end{aligned}$$

Function f gives a precise upper bound on the number of different response sequences an efficient agent can make, but it is extremely cumbersome to work with. In section 3.2.7, we are going to derive a simple upper bound on f, which is tight for our purposes. However, this bound depends on the ratio of numbers $r_1,r_2,\ldots,r_k$ being fixed.

[10]This adjustment also takes care of the outstanding factor of 2 in the estimate of the number of pseudo-instances in a leaf of our categorization tree.

3.2.6 Graphs with a Fixed Distribution of Ray Lengths

Definition 9. A *ray lengths distribution* is a vector $\vec{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_k)$ such that for any $i \in \{1, 2, \ldots, k\}$, $\alpha_i \in \mathbb{Q}^+$, and $\sum_{i=1}^{k} \alpha_i = 1$. A sun graph $G \in \mathcal{S}_{k,\ell}$ with ray lengths distribution $\vec{\alpha}$ has $\alpha_i \ell$ rays of length i.

From now on we will consider only sun graphs with a fixed ray lengths distribution.

Definition 10. The *average optimal cost* of a challenge lying in a sun graph of type k with a ray lengths distribution $\vec{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_k)$ is denoted by $\operatorname{avg}_k(\vec{\alpha})$ and equals

$$\operatorname{avg}_k(\vec{\alpha}) = k + \sum_{i=1}^{k} 2\alpha_i i$$

As we show in the next lemma, the average optimal cost can be used to determine the total optimal cost of a graph exploration.

Lemma 13. *Let $G \in \mathcal{S}_{k,\ell}$ be a sun graph with a ray lengths distribution $\vec{\alpha}$. Then the optimal cost of exploring G offline is*

$$C(opt(G)) = \ell \operatorname{avg}_k(\vec{\alpha}) + 4k^2 + k$$

Proof. From lemma 8 we have

$$C(\operatorname{opt}(G)) = k(\ell + 4k + 1) + 2\sum_{i=1}^{k} i r_i$$

where r_i is the number of rays of length i. Graph G has $\alpha_i \ell$ rays of length i, therefore

$$C(\operatorname{opt}(G)) = k\ell + 4k^2 + k + \ell \sum_{i=1}^{k} 2\alpha_i i = \ell \operatorname{avg}_k(\vec{\alpha}) + 4k^2 + k \qquad \square$$

Definition 11. The *average cost of a strategy* σ with respect to a ray lengths distribution $\vec{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_k)$ is denoted by $\operatorname{avg}_k(\sigma, \vec{\alpha})$ and equals

$$\operatorname{avg}_k(\sigma, \vec{\alpha}) = \sum_{i=1}^{k} \frac{\alpha_i}{2} \left(\operatorname{cost}_k(\sigma, i, \mathtt{C}) + \operatorname{cost}_k(\sigma, i, \mathtt{R})\right)$$

Definition 12. A ray lengths distribution $\vec{\alpha}$ is called *demanding* if the following condition holds for any strategy σ:

$$\operatorname{avg}_k(\sigma, \vec{\alpha}) \geq k + \operatorname{avg}_k(\vec{\alpha})$$

Lemma 14. *Let $\vec{\alpha}$ be a demanding ray lengths distribution. Then*

$$\min_{\sigma} \operatorname{avg}_k(\sigma, \vec{\alpha}) = k + \operatorname{avg}_k(\vec{\alpha})$$

Proof. We show that the average cost of strategy $\sigma' = \mathtt{1}^k\mathtt{2}^k$ equals $k + \mathrm{avg}_k(\vec{\alpha})$.

For any $i \in \{1, 2, \dots, k\}$ we have $\mathrm{cost}_k(\sigma', i, \mathtt{R}) = k + 2i$, because an agent applying σ' completes a challenge of length i by visiting the last vertex of the ray. That takes i steps; additional $i + k$ steps are necessary for moving to the exit. Similarly, $\mathrm{cost}_k(\sigma', i, \mathtt{C}) = 3k + 2i$, because in this case an agent completes a challenge by visiting the exit in k steps, then it takes $k + i$ steps to visit the last vertex of the ray and $i + k$ steps to return back.

Thus the average cost of σ' is

$$\begin{aligned}\mathrm{avg}_k(\sigma', \vec{\alpha}) &= \sum_{i=1}^{k} \frac{\alpha_i}{2}\left((3k + 2i) + (k + 2i)\right) \\ &= 2k + \sum_{i=1}^{k} 2\alpha_i i \\ &= k + \mathrm{avg}_k(\vec{\alpha})\end{aligned}$$

□

The average cost of a strategy is its most important characteristic. The following lemma states that an agent employing the same strategy σ for every challenge it encounters achieves competitive ratio

$$\frac{\mathrm{avg}_k(\sigma, \vec{\alpha})}{\mathrm{avg}_k(\vec{\alpha})}$$

using only one bit of advice to determine which of the two adjacent unvisited nodes it should visit first upon entering a challenge.

On the other hand, we will show in lemma 18 that to perform better than by using the strategy with minimum average cost, any agent must read $\Omega(n)$ bits of advice (provided the ray lengths distribution is demanding).

Lemma 15. *Let $\vec{\alpha}$ be a ray lengths distribution and σ a strategy. Let $\mathcal{G}$ be the set of all sun graphs of type k with ray lengths distribution $\vec{\alpha}$. Then there is an agent A exploring any graph $G \in \mathcal{G}$ and reading one bit of advice, such that*

$$C(A(G)) \leq \frac{\mathrm{avg}_k(\sigma, \vec{\alpha})}{\mathrm{avg}_k(\vec{\alpha})} C(opt(G))$$

Proof. We design a simple deterministic agent B based on the given strategy σ: Agent B travels around the cycle, and at each junction, it arbitrarily chooses one of the two adjacent unvisited nodes,[11] moves to it and continues to explore the current challenge according to strategy σ. If the agent reaches the exit before visiting all vertices of the challenge,[12] it immediately turns back to visit the last node of the ray and returns to the exit, thus entering a new challenge. At the end, the agent travels along the penalty segment to the starting node.

To calculate the cost of agent B's exploration walk, we denote by x_i the number of times it started its response on a challenge of length i by visiting the first node of the cycle. Similarly, we denote by y_i the number of times it started

[11]The choice may be arbitrary, but it must follow a deterministic rule, e.g. selecting the node with the smaller label.

[12]In the case of the last (ℓ-th) challenge, there is no exit to reach; instead, the agent recognizes that it is on the penalty segment when the k-th vertex since the last junction is neither a junction nor a leaf.

in such a situation by visiting the first node of the ray. Clearly, $x_i + y_i = \alpha_i \ell$. In addition to the challenges, the agent takes $2k(2k+1) - k = 4k^2 + k$ steps on the penalty segment (the $-k$ term emerges because the agent visits a part of the penalty segment during the last challenge). In total, the cost of agent B's walk is

$$C(B(G)) = \sum_{i=1}^{k} (x_i \operatorname{cost}_k(\sigma, i, \mathtt{C}) + y_i \operatorname{cost}_k(\sigma, i, \mathtt{R})) + 4k^2 + k$$

Let $B^{\intercal}$ be an agent behaving in the same manner as agent B, but always choosing the opposite option when deciding its first move in a challenge. Consequently, agent $B^{\intercal}$ makes on challenges of length i exactly y_i responses starting with C and x_i responses starting with R. Its total cost is

$$C(B^{\intercal}(G)) = \sum_{i=1}^{k} (y_i \operatorname{cost}_k(\sigma, i, \mathtt{C}) + x_i \operatorname{cost}_k(\sigma, i, \mathtt{R})) + 4k^2 + k$$

The mean cost of B's and $B^{\intercal}$'s walks equals

$$\begin{aligned}
\frac{C(B(G)) + C(B^{\intercal}(G))}{2} &= \sum_{i=1}^{k} \frac{x_i + y_i}{2} (\operatorname{cost}_k(\sigma, i, \mathtt{C}) + \operatorname{cost}_k(\sigma, i, \mathtt{R})) + 4k^2 + k \\
&= \sum_{i=1}^{k} \frac{\alpha_i \ell}{2} (\operatorname{cost}_k(\sigma, i, \mathtt{C}) + \operatorname{cost}_k(\sigma, i, \mathtt{R})) + 4k^2 + k \\
&= \ell \operatorname{avg}_k(\sigma, \vec{\alpha}) + 4k^2 + k
\end{aligned}$$

Therefore,

$$\min(C(B(G)), C(B^{\intercal}(G))) \le \ell \operatorname{avg}_k(\sigma, \vec{\alpha}) + 4k^2 + k$$

We define agent A now: It simulates B or $B^{\intercal}$ depending on the bit of advice it receives. Obviously, the oracle provides the bit such that A simulates the agent with smaller cost on the given graph.

Finally, we give an upper bound on the competitive ratio of A (by lemma 13, $C(\operatorname{opt}(G)) = \ell \operatorname{avg}_k(\vec{\alpha}) + 4k^2 + k$):

$$\begin{aligned}
C(A(G)) &\le \ell \operatorname{avg}_k(\sigma, \vec{\alpha}) + 4k^2 + k \\
&\le \frac{\ell \operatorname{avg}_k(\sigma, \vec{\alpha}) + 4k^2 + k}{\ell \operatorname{avg}_k(\vec{\alpha}) + 4k^2 + k} C(\operatorname{opt}(G)) \\
&\le \frac{\operatorname{avg}_k(\sigma, \vec{\alpha})}{\operatorname{avg}_k(\vec{\alpha})} C(\operatorname{opt}(G))
\end{aligned}$$

□

An another characteristic of a strategy, the second-order average cost, emerges during our derivation of the upper bound on f.

Definition 13. The *second-order average cost of a strategy* σ with respect to a ray lengths distribution $\vec{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_k)$ is denoted by $\operatorname{avg}_k^{(2)}(\sigma, \vec{\alpha})$ and equals

$$\operatorname{avg}_k^{(2)}(\sigma, \vec{\alpha}) = \sum_{i=1}^{k} \frac{\alpha_i}{2} \left(\binom{\operatorname{cost}_k(\sigma, i, \mathtt{C}) + 1}{2} + \binom{\operatorname{cost}_k(\sigma, i, \mathtt{R}) + 1}{2} \right)$$

Lemma 16. *Let $\vec{\alpha} = (\alpha_1, \alpha_2, \dots, \alpha_k)$ be a ray lengths distribution and σ a strategy. Then*

$$\operatorname{avg}_k^{(2)}(\sigma, \vec{\alpha}) > \binom{\operatorname{avg}_k(\sigma, \vec{\alpha}) + 1}{2}$$

Proof. We will show that

$$\sum_{i=1}^{k} \frac{\alpha_i}{2} \left(\binom{x_i + 1}{2} + \binom{y_i + 1}{2} \right) \geq \frac{1}{2} \left(\sum_{i=1}^{k} \frac{\alpha_i (x_i + y_i)}{2} \right) \left(\sum_{i=1}^{k} \frac{\alpha_i (x_i + y_i)}{2} + 1 \right)$$

holds for any positive real numbers $x_1, x_2, \dots, x_k, y_1, y_2, \dots, y_k$. Then we will substitute $\operatorname{cost}_k(\sigma, i, \mathtt{C})$ and $\operatorname{cost}_k(\sigma, i, \mathtt{R})$ for x_i and y_i respectively, and discuss why is in our case the inequality strict.

Let us start with the weighted inequality of quadratic and arithmetic means (QM-AM inequality):

$$\sqrt{\frac{\sum_{i=1}^{k} \alpha_i (x_i + y_i)^2}{\sum_{i=1}^{k} \alpha_i}} \geq \frac{\sum_{i=1}^{k} \alpha_i (x_i + y_i)}{\sum_{i=1}^{k} \alpha_i}$$

$$\sum_{i=1}^{k} \alpha_i (x_i + y_i)^2 \geq \left(\sum_{i=1}^{k} \alpha_i (x_i + y_i) \right)^2 \tag{3.3}$$

From another QM-AM inequality we obtain

$$\sqrt{\frac{x_i^2 + y_i^2}{2}} \geq \frac{x_i + y_i}{2}$$

$$2(x_i^2 + y_i^2) \geq (x_i + y_i)^2$$

which we substitute into (3.3):

$$2 \sum_{i=1}^{k} \alpha_i (x_i^2 + y_i^2) \geq \left(\sum_{i=1}^{k} \alpha_i (x_i + y_i) \right)^2$$

$$2 \sum_{i=1}^{k} \alpha_i (x_i (x_1 + 1) + y_i (y_1 + 1)) \geq \left(\sum_{i=1}^{k} \alpha_i (x_i + y_i) \right)^2 + 2 \sum_{i=1}^{k} \alpha_i (x_i + y_i)$$

$$\sum_{i=1}^{k} \frac{\alpha_i}{2} \left(\binom{x_i + 1}{2} + \binom{y_i + 1}{2} \right) \geq \frac{1}{2} \left(\sum_{i=1}^{k} \frac{\alpha_i (x_i + y_i)}{2} \right) \left(\sum_{i=1}^{k} \frac{\alpha_i (x_i + y_i)}{2} + 1 \right)$$

The equality holds if and only if $x_1 = x_2 = \cdots = x_k = y_1 = y_2 = \cdots = y_k$. This condition is not satisfied in our case: For any strategy σ, $y_1 = \operatorname{cost}_k(\sigma, 1, \mathtt{R}) = k + 2$, as an agent applying the strategy completes the challenge in its first move and then makes $k + 1$ steps to reach exit. However, $x_1 = \operatorname{cost}_k(\sigma, 1, \mathtt{C}) \geq k + 4$, because after its first move, the agent makes at least 2 steps in order to visit the last vertex of the ray and then $k + 1$ steps to finish at the exit. □

Continuing our preparations for lemma 18, we prove a lower bound on the multinomial coefficient from the statement of lemma 12.

Lemma 17. *Let $\vec{\alpha} = (\alpha_1, \alpha_2, \dots, \alpha_k)$ be a ray lengths distribution and ℓ such a number that $\alpha_i \ell$ is an integer for all $i \in \{1, 2, \dots, k\}$. Then*

$$\lg \binom{\ell}{\alpha_1\ell, \alpha_2\ell, \dots, \alpha_k\ell} \geq -\ell \sum_{i=1}^{k} \alpha_i \lg \alpha_i + O(\lg \ell)$$

Proof. Let us begin by reminding well-known lower and upper bounds of the value of $\lg n! = \lg 1 + \lg 2 + \cdots + \lg n$:

$$\lg n! = \sum_{i=1}^{n} \lg i \geq \int_1^n \lg x \,\mathrm{d}x = n \lg n - \frac{n-1}{\ln 2}$$
$$\lg n! = \sum_{i=1}^{n} \lg i \leq \int_1^{n+1} \lg x \,\mathrm{d}x = (n+1)\lg(n+1) - \frac{n}{\ln 2}$$

To obtain a lower bound on the multinomial coefficient, we rewrite it in terms of factorials and apply the aforementioned bounds:

$$\begin{aligned}
\lg \binom{\ell}{\alpha_1\ell, \alpha_2\ell, \dots, \alpha_k\ell} &= \lg \ell! - \sum_{i=1}^{k} \lg(\alpha_i\ell)! \\
&\geq \ell \lg \ell - \frac{\ell - 1}{\ln 2} - \sum_{i=1}^{k} \left((\alpha_i\ell + 1)\lg(\alpha_i\ell + 1) - \frac{\alpha_i\ell}{\ln 2} \right) \\
&\geq \ell \lg \ell + \frac{1}{\ln 2} - \sum_{i=1}^{k} (\alpha_i\ell \lg(\alpha_i\ell + 1) + \lg(\alpha_i\ell + 1)) \\
&\geq \ell \lg \ell - \ell \sum_{i=1}^{k} \alpha_i \lg \left(\alpha_i\ell \left(1 + \frac{1}{\alpha_i\ell} \right) \right) + O(\lg \ell) \\
&\geq -\ell \sum_{i=1}^{k} \alpha_i \lg \alpha_i - \ell \sum_{i=1}^{k} \alpha_i \lg \left(1 + \frac{1}{\alpha_i\ell} \right) + O(\lg \ell)
\end{aligned}$$

We use the Taylor series of function $\lg(1+x)$ to eliminate term $\lg\left(1 + \frac{1}{\alpha_i\ell}\right)$:

$$\begin{aligned}
\lg \binom{\ell}{\alpha_1\ell, \alpha_2\ell, \dots, \alpha_k\ell} &\geq -\ell \sum_{i=1}^{k} \alpha_i \lg \alpha_i - \ell \sum_{i=1}^{k} \alpha_i \left(\frac{1}{\alpha_i\ell \ln 2} + O\left(\ell^{-2}\right) \right) + O(\lg \ell) \\
&\geq -\ell \sum_{i=1}^{k} \alpha_i \lg \alpha_i + O(\lg \ell)
\end{aligned}$$

□

3.2.7 Linear Lower Bound on Advice

Finally, we present a complementary result to lemma 15: We prove that a linear number of advice bits is necessary to outperform an agent using the strategy with the minimum average cost and a single bit of advice.

However, our proof works only for demanding distributions. This restriction can be interpreted as ruling out distributions that under-represent longer rays and therefore decrease the effective type of a sun graph. We will discuss it in more detail later.

Lemma 18. *Let $k \geq 1$ be a type of sun graphs and $\vec{\alpha}$ a demanding ray lengths distribution. Then any agent exploring sun graphs of type k with ray lengths distribution $\vec{\alpha}$ that achieves competitive ratio*

$$\frac{\min_\sigma \operatorname{avg}_k(\sigma, \vec{\alpha})}{\operatorname{avg}_k(\vec{\alpha})} - \varepsilon = \frac{k + \operatorname{avg}_k(\vec{\alpha})}{\operatorname{avg}_k(\vec{\alpha})} - \varepsilon$$

(for $\varepsilon > 0$), requires $\Omega(n)$ bits of advice.

Proof. Let A be an agent such that for any sun graph G of type k,

$$C(A(G)) \leq \left(\frac{k + \operatorname{avg}_k(\vec{\alpha})}{\operatorname{avg}_k(\vec{\alpha})} - \varepsilon\right) C(\operatorname{opt}(G)) + O(1)$$

where ε is infinitesimally small, but greater than 0. Although our proof works only for sufficiently small values of ε, the claim for larger ε follows from it. We will show that agent A must read $\Omega(\ell) = \Omega(n)$ bits of advice in order to efficiently explore every graph $G \in \mathcal{S}_{k,\ell}$ with ray lengths distribution $\vec{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_k)$, where ℓ is sufficiently large. Let us note that due to $\alpha_i \in \mathbb{Q}^+$, there are graphs satisfying these requirements with order greater than any given threshold.

We will make use of the lower bound on advice from lemma 12. However, before using this lemma, we must ensure that the following condition holds:

$$\left(\frac{k + \operatorname{avg}_k(\vec{\alpha})}{\operatorname{avg}_k(\vec{\alpha})} - \varepsilon\right) C(\operatorname{opt}(G)) + O(1) < C(\operatorname{opt}(G)) + k(\ell - 3)$$

$$\left(\frac{k}{\operatorname{avg}_k(\vec{\alpha})} - \varepsilon\right) C(\operatorname{opt}(G)) + O(1) < k\ell$$

From lemma 13 we obtain

$$\left(\frac{k}{\operatorname{avg}_k(\vec{\alpha})} - \varepsilon\right) (\ell \operatorname{avg}_k(\vec{\alpha}) + 4k^2 + k) + O(1) < k\ell$$

$$(k - \varepsilon \operatorname{avg}_k(\vec{\alpha}))\ell + O(1) < k\ell$$

$$O(1) < \varepsilon \operatorname{avg}_k(\vec{\alpha})\, \ell$$

where the constant $O(1)$ depends on the agent, $\vec{\alpha}$, and k. Clearly, the condition holds when ℓ is sufficiently large.

To make subsequent derivations briefer, let us denote

$$\min_\sigma \operatorname{avg}_k(\sigma, \vec{\alpha}) = k + \operatorname{avg}_k(\vec{\alpha})$$

by g, and

$$\max \left\{ \operatorname{avg}_k^{(2)}(\sigma, \vec{\alpha}) \;\middle|\; \operatorname{avg}_k(\sigma, \vec{\alpha}) = g \right\}$$

by $g^{(2)}$.

Let us denote by $b(\ell)$ the maximum number of bits agent A reads on graphs in $\mathcal{S}_{k,\ell}$. Now we can use the bound from lemma 12:

$$b(\ell) \geq \ell - 2 + \lg \binom{\ell}{\alpha_1 \ell, \alpha_2 \ell, \ldots, \alpha_k \ell} -$$
$$- \lg f\left(\ell \vec{\alpha}, \left(\frac{k + \operatorname{avg}_k(\vec{\alpha})}{\operatorname{avg}_k(\vec{\alpha})} - \varepsilon\right) C(\operatorname{opt}(G)) + O(1)\right)$$

Again, we substitute the expression from lemma 13 for $C(\mathrm{opt}(G))$. Moreover, we apply the lower bound on the multinomial coefficient from lemma 17.

$$\begin{aligned} b(\ell) &\geq \ell - \ell \sum_{i=1}^{k} \alpha_i \lg \alpha_i + O(\lg \ell) - \\ &\qquad - \lg f\left(\ell\vec{\alpha}, \left(\frac{g}{\mathrm{avg}_k(\vec{\alpha})} - \varepsilon\right)(\ell\,\mathrm{avg}_k(\vec{\alpha}) + 4k^2 + k) + O(1)\right) \\ b(\ell) &\geq \ell - \ell \sum_{i=1}^{k} \alpha_i \lg \alpha_i - \lg f\big(\ell\vec{\alpha}, \ell(g - \varepsilon\,\mathrm{avg}_k(\vec{\alpha})) + O(1)\big) + O(\lg \ell) \qquad (3.4) \end{aligned}$$

We are going to prove an upper bound on the values of $f(\vec{x}, y)$ (where $\vec{x} = (x_1, x_2, \ldots, x_k)$) in the form

$$f(\vec{x}, y) \leq (1 + \lambda\varepsilon)^y \prod_{i=1}^{k} \left(\frac{2\psi}{\alpha_i}\right)^{x_i} \qquad (3.5)$$

where λ and ψ are constants that we will determine later ($\lambda, \psi > 0$).

The right-hand side of (3.5) is always positive, therefore we focus only on cases when f is positive too. We prove the claim by induction on the sum of x_i's. For any $y \geq 0$,

$$f(\vec{0}, y) = 1 \leq (1 + \lambda\varepsilon)^y$$

If some of the x_i's are non-negative, by the definition of f we have

$$f(\vec{x}, y) = \max_{\sigma} \sum_{i=1}^{k} \big(f(\vec{x} - \vec{\tau}_i, y - \mathrm{cost}_k(\sigma, i, \mathtt{C})) + f(\vec{x} - \vec{\tau}_i, y - \mathrm{cost}_k(\sigma, i, \mathtt{R}))\big)$$

By applying the inductive hypothesis, we obtain an upper bound on $f(\vec{x}, y)$:

$$\max_{\sigma} \sum_{i=1}^{k} \left(\left((1 + \lambda\varepsilon)^{y - \mathrm{cost}_k(\sigma,i,\mathtt{C})} + (1 + \lambda\varepsilon)^{y - \mathrm{cost}_k(\sigma,i,\mathtt{R})}\right) \frac{\alpha_i}{2\psi} \prod_{j=1}^{k} \left(\frac{2\psi}{\alpha_j}\right)^{x_j} \right)$$

It suffices to show that

$$\begin{aligned} \max_{\sigma} \sum_{i=1}^{k} \frac{\alpha_i}{2\psi} \left((1 + \lambda\varepsilon)^{y - \mathrm{cost}_k(\sigma,i,\mathtt{C})} + (1 + \lambda\varepsilon)^{y - \mathrm{cost}_k(\sigma,i,\mathtt{R})}\right) &\leq (1 + \lambda\varepsilon)^y \\ \max_{\sigma} \sum_{i=1}^{k} \frac{\alpha_i}{2} \left((1 + \lambda\varepsilon)^{-\mathrm{cost}_k(\sigma,i,\mathtt{C})} + (1 + \lambda\varepsilon)^{-\mathrm{cost}_k(\sigma,i,\mathtt{R})}\right) &\leq \psi \qquad (3.6) \end{aligned}$$

The value of λ depends only on g, $g^{(2)}$, and k; we will define it exactly later. For a fixed value of λ, we choose the smallest ψ such that (3.6) holds (i.e. ψ is equal to the left-hand side of the inequality). We denote the strategy which maximizes the expression on the left-hand side by σ'. This concludes the proof by induction.

Using the Taylor series

$$(1 + x)^{-m} = 1 - mx + \binom{m+1}{2} x^2 + O(x^3)$$

we obtain the following estimate of the value of ψ:

$$\psi = \sum_{i=1}^{k} \frac{\alpha_i}{2} \left((1+\lambda\varepsilon)^{-\operatorname{cost}_k(\sigma',i,\mathtt{C})} + (1+\lambda\varepsilon)^{-\operatorname{cost}_k(\sigma',i,\mathtt{R})} \right)$$

$$\psi = \sum_{i=1}^{k} \frac{\alpha_i}{2} \left(1 - \operatorname{cost}_k(\sigma',i,\mathtt{C})\,\lambda\varepsilon + \binom{\operatorname{cost}_k(\sigma',i,\mathtt{C})+1}{2}\lambda^2\varepsilon^2 + \right.$$

$$\left. 1 - \operatorname{cost}_k(\sigma',i,\mathtt{R})\,\lambda\varepsilon + \binom{\operatorname{cost}_k(\sigma',i,\mathtt{R})+1}{2}\lambda^2\varepsilon^2 + O(\varepsilon^3) \right)$$

$$\psi = 1 - \operatorname{avg}_k(\sigma',\vec{\alpha})\,\lambda\varepsilon + \operatorname{avg}_k^{(2)}(\sigma',\vec{\alpha})\,\lambda^2\varepsilon^2 + O(\varepsilon^3) \tag{3.7}$$

As ε is infinitesimally small, σ' must have the lowest average cost of all strategies, otherwise it would not maximize the left-hand side of (3.6). For the same reason, it must have the highest second-order average cost among strategies with the lowest average cost. Therefore,

$$\psi = 1 - g\lambda\varepsilon + g^{(2)}\lambda^2\varepsilon^2 + O(\varepsilon^3) \tag{3.8}$$

We will need an estimate of $\lg\psi$; from the Taylor series

$$\lg(1+x) = \frac{1}{\ln 2}\left(x - \frac{x^2}{2} + O(x^3) \right)$$

we obtain

$$\lg\psi = \lg\left(1 - g\lambda\varepsilon + g^{(2)}\lambda^2\varepsilon^2 + O(\varepsilon^3) \right)$$

$$= \frac{1}{\ln 2}\left(-g\lambda\varepsilon + \left(g^{(2)} - \frac{1}{2}g^2 \right)\lambda^2\varepsilon^2 + O(\varepsilon^3) \right) \tag{3.9}$$

In the next step, we prove an upper bound on $\lg f(\ell\vec{\alpha}, \ell(g - \varepsilon\operatorname{avg}_k(\vec{\alpha})) + O(1))$ starting with (3.5):

$$f(\ell\vec{\alpha}, \ell(g-\varepsilon\operatorname{avg}_k(\vec{\alpha})) + O(1)) \le (1+\lambda\varepsilon)^{\ell(g-\varepsilon\operatorname{avg}_k(\vec{\alpha}))+O(1)} \prod_{i=1}^{k}\left(\frac{2\psi}{\alpha_i}\right)^{\alpha_i\ell}$$

$$\lg f(\ell\vec{\alpha}, \ell(g-\varepsilon\operatorname{avg}_k(\vec{\alpha})) + O(1)) \le (\ell(g-\varepsilon\operatorname{avg}_k(\vec{\alpha})) + O(1))\lg(1+\lambda\varepsilon) +$$

$$+ \sum_{i=1}^{k} \alpha_i\ell\,(1 + \lg\psi - \lg\alpha_i))$$

$$\le \ell(g-\varepsilon\operatorname{avg}_k(\vec{\alpha}))\lg(1+\lambda\varepsilon) + O(1) +$$

$$+ \ell\left(1 + \lg\psi - \sum_{i=1}^{k}\alpha_i\lg\alpha_i \right)$$

We substitute the estimate (3.9) for $\lg \psi$, and the Taylor series for $\lg(1 + \lambda\varepsilon)$:

$$\begin{aligned}
&\lg f(\ell\vec{\alpha}, \ell(g - \varepsilon \operatorname{avg}_k(\vec{\alpha})) + O(1)) \leq \\
&\quad \leq \frac{\ell}{\ln 2}(g - \varepsilon \operatorname{avg}_k(\vec{\alpha})) \left(\lambda\varepsilon - \frac{1}{2}\lambda^2\varepsilon^2 + O(\varepsilon^3)\right) + \ell\left(1 - \sum_{i=1}^{k} \alpha_i \lg \alpha_i\right) + \\
&\quad\quad + \frac{\ell}{\ln 2}\left(-g\lambda\varepsilon + \left(g^{(2)} - \frac{1}{2}g^2\right)\lambda^2\varepsilon^2 + O(\varepsilon^3)\right) + O(1) \\
&\quad \leq \frac{\ell}{\ln 2}\left(\left(\left(g^{(2)} - \frac{1}{2}g^2 - \frac{1}{2}g\right)\lambda^2 - \operatorname{avg}_k(\vec{\alpha})\,\lambda\right)\varepsilon^2 + O(\varepsilon^3)\right) + \\
&\quad\quad + \ell\left(1 - \sum_{i=1}^{k} \alpha_i \lg \alpha_i\right) + O(1) \qquad (3.10)
\end{aligned}$$

Let us return to the lower bound on advice; we apply (3.10) to (3.4):

$$b(\ell) \geq \frac{\ell}{\ln 2}\left(\left(\left(\frac{g(g+1)}{2} - g^{(2)}\right)\lambda^2 + \operatorname{avg}_k(\vec{\alpha})\,\lambda\right)\varepsilon^2 + O(\varepsilon^3)\right) + O(\lg \ell) \tag{3.11}$$

We aim to make this lower bound as tight as possible, hence we choose the value of λ which maximizes the coefficient of ε^2. The expression

$$\left(\frac{g(g+1)}{2} - g^{(2)}\right)\lambda^2 + \operatorname{avg}_k(\vec{\alpha})\,\lambda$$

is a quadratic function of variable λ with a negative leading coefficient (by lemma 16), therefore it attains its maximum for

$$\lambda = \frac{\operatorname{avg}_k(\vec{\alpha})}{2g^{(2)} - g^2 - g} \tag{3.12}$$

Substituting this value of λ into (3.11), we obtain

$$b(\ell) \geq \ell\left(\frac{(\operatorname{avg}_k(\vec{\alpha}))^2}{\ln 2(4g^{(2)} - 2g^2 - 2g)}\varepsilon^2 + O(\varepsilon^3)\right) + O(\lg \ell) \tag{3.13}$$

The coefficient of ε^2 is positive (by lemma 16), therefore $b(\ell) \in \Omega(\ell)$ (assuming that ε is sufficiently small). As the number of vertices $n \leq 2k\ell + 4k^2 + k$, we have also shown that agent A reads $\Omega(n)$ bits of advice on some sun graphs of type k with ray lengths distribution $\vec{\alpha}$. □

3.2.8 Uniform Distribution

Lemma 18 works only for demanding ray lengths distributions. In the next part, we prove that the uniform distribution satisfies this condition. Consequently, we show a lower bound on competitive ratio for which at least linear advice is necessary. However, we will need a different distribution for our main result.

Definition 14. A *uniform ray lengths distribution* $\vec{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_k)$ is a ray lengths distribution such that $\alpha_i = 1/k$ for all $i \in \{1, 2, \ldots, k\}$.

Lemma 19. *Let $\vec{\alpha}$ be a uniform ray lengths distribution. Then*

$$\mathrm{avg}_k(\vec{\alpha}) = 2k + 1$$

Proof. From the definition of the average optimal cost we have

$$\mathrm{avg}_k(\vec{\alpha}) = k + \sum_{i=1}^{k} \frac{2i}{k} = 2k + 1 \qquad \square$$

Lemma 20. *Any uniform ray lengths distribution $\vec{\alpha}$ is demanding.*

Proof. We are supposed to prove that for any strategy σ,

$$\mathrm{avg}_k(\sigma, \vec{\alpha}) = \sum_{i=1}^{k} \frac{1}{2k}(\mathrm{cost}_k(\sigma, i, C) + \mathrm{cost}_k(\sigma, i, R)) \geq 3k + 1$$

Let us consider challenges from a family and its transpose in the way how an agent applying strategy σ sees them before completion: There is the entrance and two paths incident to it. The paths are marked `1` and `2` correspondingly with the notation of the strategy. We denote the family for which the agent's first move is `R` by $\{H_1, H_2, \ldots, H_k\}$.

Now, let us examine the paths' nodes in the order specified by σ. The first node in σ is the last vertex of the ray in challenge H_1, from which the agent must return to the entrance and take additional k steps towards the exit. Therefore, $\mathrm{cost}_k(\sigma, 1, \mathtt{R}) = k + 2$. If the second node in σ also lies on path `1`, it is the last ray vertex in H_2, and we have $\mathrm{cost}_k(\sigma, 2, \mathtt{R}) = k + 4$. Otherwise, this node is the last ray vertex in H_1^{T}, hence $\mathrm{cost}_k(\sigma, 1, \mathtt{C}) = k + 4$.

In general, the j-th node in σ is the last ray vertex in a challenge of length a_j. The agent must visit $j - a_j$ vertices of the cycle before it enters the j-th node, therefore the cost of its response is at least $2(j - a_j) + 2a_j + k = k + 2j$.

As the ray lengths distribution is uniform, we may simply add up the lower bounds on individual response costs in order to obtain a lower bound on the average cost:

$$\mathrm{avg}_k(\sigma, \vec{\alpha}) \geq \frac{1}{2k} \sum_{j=1}^{2k} (k + 2j) = 3k + 1 \qquad \square$$

Theorem 21. *For any $\varepsilon > 0$, any agent exploring sun graphs with competitive ratio $3/2 - \varepsilon$ requires $\Omega(n)$ bits of advice.*

Proof. When we choose some constant $k \geq 1$ and a uniform ray lengths distribution $\vec{\alpha}$, by lemma 20 we are allowed to use lemma 18. After substituting $2k + 1$ for $\mathrm{avg}_k(\vec{\alpha})$ (from lemma 19) it states that any agent with competitive ratio strictly less than

$$\frac{k + \mathrm{avg}_k(\vec{\alpha})}{\mathrm{avg}_k(\vec{\alpha})} = \frac{3k + 1}{2k + 1} \tag{3.14}$$

requires $\Omega(n)$ bits of advice.

The limit of the fraction as k approaches infinity is

$$\lim_{k \to \infty} \frac{3k + 1}{2k + 1} = \frac{3}{2}$$

Therefore, for any $\varepsilon > 0$, there is a value of k such that fraction (3.14) is strictly greater than $3/2 - \varepsilon$. $\square$

3.2.9 Optimal Demanding Distribution

In order to obtain a better lower bound on competitive ratio, we need to find a demanding distribution with smaller average optimal cost.

Lemma 22. *Let $k \geq 1$ be a type of sun graphs and $\vec{\alpha} = (\alpha_1, \alpha_2, \dots, \alpha_k)$ a ray lengths distribution such that*

$$\alpha_i = \frac{2i}{k+i} - \frac{2(i-1)}{k+i-1}$$

Then $\vec{\alpha}$ is a demanding distribution. Moreover, it has the minimal average optimal cost $\mathrm{avg}_k(\vec{\alpha})$ among demanding distributions.

Proof. Given a strategy, we will call any its maximal substring containing a single kind of letter a *run*. Let us define a set of *basic strategies* $\mathcal{B} = \{\sigma_1, \sigma_2, \dots, \sigma_k\}$, where $\sigma_j = \mathtt{1}^j\mathtt{2}^k\mathtt{1}^{k-j}$. Clearly, set $\mathcal{B}$ consists of all strategies with at most 3 runs.

An agent employing strategy σ_j visits j vertices of the first path before trying the second path. Therefore, if the first path is actually the cycle segment in a challenge of length i, the cost of the agent's response equals

$$\mathrm{cost}_k(\sigma_j, i, \mathtt{C}) = k + 2i + 2j$$

If the first path is the ray in a challenge of length i, the cost depends on whether the agent encounters the last ray vertex during its j moves on the first path:

$$\mathrm{cost}_k(\sigma_j, i, \mathtt{R}) = \begin{cases} k + 2i & \text{if } i \leq j \\ 3k + 2i + 2j & \text{if } i > j \end{cases}$$

For any demanding distribution $\vec{\alpha} = (\alpha_1, \alpha_2, \dots, \alpha_k)$, the average cost of a basic strategy σ_j is at least $k + \mathrm{avg}_k(\vec{\alpha})$. Let us first compute $\mathrm{avg}_k(\sigma_j, \vec{\alpha})$:

$$\begin{aligned}
\mathrm{avg}_k(\sigma_j, \vec{\alpha}) &= \sum_{i=1}^{k} \frac{\alpha_i}{2}(k + 2i + 2j) + \sum_{i=1}^{j} \frac{\alpha_i}{2}(k + 2i) + \sum_{i=j+1}^{k} \frac{\alpha_i}{2}(3k + 2i + 2j) \\
&= \sum_{i=1}^{k} 2\alpha_i(k + i + j) - \sum_{i=1}^{j} \alpha_i(k + j) \\
&= k + 2j + \mathrm{avg}_k(\vec{\alpha}) - (k + j)\sum_{i=1}^{j} \alpha_i
\end{aligned}$$

If $\vec{\alpha}$ is demanding, we have

$$k + \mathrm{avg}_k(\vec{\alpha}) \leq k + 2j + \mathrm{avg}_k(\vec{\alpha}) - (k + j)\sum_{i=1}^{j} \alpha_i$$

$$\sum_{i=1}^{j} \alpha_i \leq \frac{2j}{k+j} \tag{3.15}$$

Let us set

$$\alpha_i = \frac{2i}{k+i} - \frac{2(i-1)}{k+i-1}$$

so that the equality holds in (3.15) for any $j \in \{1, 2, \ldots, k\}$ (which also implies $\sum_{i=1}^{k} \alpha_i = 1$). Firstly, we will prove that no distribution satisfying (3.15) achieves lower average optimal cost than $\vec{\alpha}$ does. Secondly, we will show that $\vec{\alpha}$ is demanding.

Let us consider a distribution $\vec{\beta} = (\beta_1, \beta_2, \ldots, \beta_k)$ satisfying (3.15) such that for some j, the inequality is strict, i.e.

$$\sum_{i=1}^{j} \beta_i = \frac{2j}{k+j} - \varepsilon$$

for some $\varepsilon > 0$. We may increase β_j by ε and correspondingly decrease one or more of $\beta_{j'}$s for $j' > j$, thus obtaining a distribution with lower average optimal cost that satisfies (3.15) too. Therefore, the only distribution that cannot be improved is $\vec{\alpha}$.

Now, we are going to prove that for any strategy σ, $\text{avg}_k(\sigma, \vec{\alpha}) \geq k + \text{avg}_k(\vec{\alpha})$. We already know this condition holds for strategies with at most 3 runs. Let us consider a strategy σ with at least 4 runs; we will denote the lengths of the last two runs by x and y. Regardless of the letters in these runs, there are some other runs before them with the same letters, hence $1 \leq x, y \leq k-1$. We create another strategy σ' by exchanging the last two runs of σ (the number of runs decreases by one).

Similarly as in the proof of lemma 20, we examine the vertices of two paths of a challenge in the order specified by σ and σ'. In particular, we are interested in the last x vertices of one path and the last y vertices of the other path, because that is where our two strategies differ. Let us denote by t the number of steps an agent employing σ takes to visit the $(2k-x-y)$-th vertex. To reach a node among the following x vertices, the agent makes additional $k-y$ steps to return to the entrance and then i steps, where i is the distance between the node and the entrance. For each of the following y vertices, the agent makes $t + (k-y) + 2k + i$ steps. Therefore, the contribution of the last $x+y$ vertices to the average cost of σ is

$$\sum_{i=k-x+1}^{k} \frac{\alpha_i}{2}(t + 2k - y + 2i) + \sum_{i=k-y+1}^{k} \frac{\alpha_i}{2}(t + 4k - y + 2i) \tag{3.16}$$

On the other hand, an agent employing σ' does not change paths after visiting the $(2k-x-y)$-th vertex. Hence, to reach a node among the following y vertices, it takes just $i - (k-y)$ steps (again, i is the distance between the node and the entrance). For the following x vertices, the agent makes $y + k + i$ steps. Altogether, the contribution to the average cost of σ' is

$$\sum_{i=k-y+1}^{k} \frac{\alpha_i}{2}(t + y + 2i) + \sum_{i=k-x+1}^{k} \frac{\alpha_i}{2}(t + 2k + y + 2i) \tag{3.17}$$

We show that (3.17) is less than (3.16):

$$\begin{aligned}
\sum_{i=k-x+1}^{k} \frac{\alpha_i}{2} y + \sum_{i=k-y+1}^{k} \frac{\alpha_i}{2} y &< \sum_{i=k-x+1}^{k} \frac{\alpha_i}{2}(-y) + \sum_{i=k-y+1}^{k} \frac{\alpha_i}{2}(4k-y) \\
\sum_{i=k-x+1}^{k} \alpha_i y &< \sum_{i=k-y+1}^{k} \alpha_i (2k-y) \\
y\left(1 - \sum_{i=1}^{k-x} \alpha_i\right) &< (2k-y)\left(1 - \sum_{i=1}^{k-y} \alpha_i\right) \\
y\left(1 - \frac{2(k-x)}{2k-x}\right) &< (2k-y)\left(1 - \frac{2(k-y)}{2k-y}\right) \\
y \cdot \frac{x}{2k-x} &< (2k-y) \cdot \frac{y}{2k-y} \\
x &< k
\end{aligned}$$

The average cost of any strategy with at least 4 runs can be improved while decreasing the number of runs by one. Therefore, basic strategies achieve lower average cost than others, and we proved that they satisfy the condition $\mathrm{avg}_k(\sigma_j, \vec{\alpha}) \geq k + \mathrm{avg}_k(\vec{\alpha})$. Hence, $\vec{\alpha}$ is a demanding distribution. □

Lemma 23. *Let $\vec{\alpha}$ be the ray lengths distribution defined in the statement of lemma 22. Then*

$$\mathrm{avg}_k(\vec{\alpha}) = 4k\left(H_{2k-1} - H_{k-1}\right) - k$$

where H_n is the n-th harmonic number.

Proof. From the definition of the average optimal cost we have

$$\begin{aligned}
\mathrm{avg}_k(\vec{\alpha}) &= k + \sum_{i=1}^{k} 2i \left(\frac{2i}{k+i} - \frac{2(i-1)}{k+i-1}\right) \\
&= k + 2k \cdot \frac{2k}{2k} - 4\sum_{i=1}^{k} \frac{i-1}{k+i-1} \\
&= 3k - 4\sum_{i=1}^{k}\left(1 - \frac{k}{k+i-1}\right) \\
&= 4k(H_{2k-1} - H_{k-1}) - k
\end{aligned}$$

□

Theorem 24. *For any $\varepsilon > 0$, any agent exploring sun graphs with competitive ratio*

$$1 + \frac{1}{\ln 16 - 1} - \varepsilon \approx 1.564 - \varepsilon$$

requires $\Omega(n)$ bits of advice.

Proof. This time we use the ray lengths distribution defined in lemma 22 for a chosen type of sun graphs k. Lemma 18 states that any agent with competitive ratio strictly less than

$$\frac{k + \mathrm{avg}_k(\vec{\alpha})}{\mathrm{avg}_k(\vec{\alpha})} = 1 + \frac{1}{4(H_{2k-1} - H_{k-1}) - 1} \tag{3.18}$$

requires $\Omega(n)$ bits of advice.

The limit of $H_{2k-1} - H_{k-1}$ as k approaches infinity is

$$\lim_{k\to\infty} \sum_{i=0}^{k-1} \frac{1}{k+i} = \lim_{k\to\infty} \frac{1}{k} \sum_{i=0}^{k-1} \frac{1}{1+\frac{i}{k}} = \int_0^1 \frac{1}{1+x}\,\mathrm{d}x = \ln 2$$

from which we have the limit of (3.18):

$$\lim_{k\to\infty} 1 + \frac{1}{4(H_{2k-1} - H_{k-1}) - 1} = 1 + \frac{1}{\ln 16 - 1}$$

Therefore, for any $\varepsilon > 0$, there is a value of k such that fraction (3.18) is strictly greater than

$$1 + \frac{1}{\ln 16 - 1} - \varepsilon \qquad \square$$

Conclusion

In this thesis, we presented an overview of the field of online algorithms and competitive analysis. Especially, we were interested in advice complexity of online algorithms, i.e. how one can improve the competitive ratio of an online algorithm by providing it with a limited amount of information about the input instance the algorithm is being run on. We chose the online graph exploration problem as the subject of our research.

We summarized known results and previous work on the online graph exploration problem not only in the general case, but also when the input instances are restricted to cycles or unweighted graphs (i.e. graphs containing only edges of a unit length). In general, it remains an open question whether there is a deterministic algorithm with a constant competitive ratio. The best known lower bound on the competitive ratio of an agent without advice is $5/2-\varepsilon$ ([4]). Using a linear amount of advice, one can achieve competitive ratio $6+\varepsilon$ (also [4]). On cycles, an optimal algorithm with the competitive ratio of $(1+\sqrt{3})/2 \approx 1.366$ was discovered by [9]. On unweighted graphs, the depth-first search algorithm is optimal with the competitive ratio of 2, as was shown by [9].

Our results contribute to the research of the two aforementioned special cases of the online graph exploration problem. We proved a lower bound of $\lg n + O(1)$ advice bits necessary to find an optimal solution if the set of input instances is restricted to weighted cycles. Conversely, we showed that $\lg n + O(\lg\lg n)$ bits of advice are sufficient for an optimal agent. Furthermore, we devised an algorithm that is able to efficiently utilize a constant number of advice bits: Given $2k+1$ bits, it explores cycles with the competitive ratio of $1 + 3/2^{2^k+1}$. In the case of unweighted graphs, we proved a lower bound of $\Omega(n \lg n)$ advice bits necessary to find an optimal solution. On the other hand, we showed that $O(n \lg n)$ bits are also sufficient. For sparse graphs, our optimal algorithms require $O(n \lg \Delta)$ or $O(m)$ bits of advice, where Δ is the maximum degree of a graph and m is the number of its edges. Finally, we proved that any agent exploring unweighted graphs with a competitive ratio less than $1 + 1/(\ln 16 - 1) \approx 1.564$ must obtain $\Omega(n)$ bits of advice.

There are several questions left open, mainly in the case of exploring unweighted graphs. We showed that $\Omega(n \lg n)$ bits of advice are necessary for an agent to find the optimal exploration walk, but what is the lower bound on advice for an agent performing only slightly worse than the optimum (i.e. using at most a constant number of additional steps)? Can the lower bound of 1.564 on the competitive ratio of an agent with sublinear advice be pushed towards 2? It would also be interesting to find an algorithm that uses $o(n \lg n)$ advice and achieves a competitive ratio less than 2.

Bibliography

[1] Y. Asahiro, E. Miyano, S. Miyazaki, and T. Yoshimuta. Weighted nearest neighbor algorithms for the graph exploration problem on cycles. *Information Processing Letters*, 110(3):93 – 98, 2010.

[2] H.-J. Böckenhauer, D. Komm, R. Královič, R. Královič, and T. Mömke. On the advice complexity of online problems. In *Algorithms and Computation*, pages 331–340. Springer, 2009.

[3] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis.* Cambridge University Press, 1998.

[4] S. Dobrev, R. Královič, and E. Markou. Online graph exploration with advice. In G. Even and M. Halldórsson, editors, *Structural Information and Communication Complexity*, volume 7355 of *Lecture Notes in Computer Science*, pages 267–278. Springer Berlin Heidelberg, 2012.

[5] A. Fiat, R. M. Karp, M. Luby, L. A. McGeoch, D. D. Sleator, and N. E. Young. Competitive paging algorithms. *Journal of Algorithms*, 12(4):685–699, 1991.

[6] C. A. Hurkens and G. J. Woeginger. On the nearest neighbor rule for the traveling salesman problem. *Operations Research Letters*, 32(1):1–4, 2004.

[7] B. Kalyanasundaram and K. R. Pruhs. Constructing competitive tours from local information. In *Automata, Languages and Programming*, pages 102–113. Springer Berlin Heidelberg, 1993.

[8] N. Megow, K. Mehlhorn, and P. Schweitzer. Online graph exploration: New results on old and new algorithms. *Theoretical Computer Science*, 463:62–72, 2012. Special Issue on Theory and Applications of Graph Searching Problems.

[9] S. Miyazaki, N. Morimoto, and Y. Okabe. The online graph exploration problem on restricted graphs. *IEICE transactions on information and systems*, 92(9):1620–1627, 2009.

[10] D. Rosenkrantz, R. Stearns, and P. Lewis, II. An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing*, 6(3):563–581, 1977.

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