Online Minimum Spanning Tree with Advice
(Extended Abstract)

Maria Paola Bianchi[1(✉)], Hans-Joachim Böckenhauer[1], Tatjana Brülisauer[1], Dennis Komm[1], and Beatrice Palano[2]

[1] Department of Computer Science, ETH Zurich, Zürich, Switzerland
{maria.bianchi,hjb,tatjana.bruelisauer,dennis.komm}@inf.ethz.ch
[2] Dipartimento di Informatica, Università Degli Studi di Milano, Milan, Italy
palano@di.unimi.it

Abstract. In the online minimum spanning tree problem, a graph is revealed vertex by vertex; together with every vertex, all edges to vertices that are already known are given, and an online algorithm must irrevocably choose a subset of them as a part of its solution. The advice complexity of an online problem is a means to quantify the information that needs to be extracted from the input to achieve good results. For a graph of size n, we show an asymptotically tight bound of $\Theta(n \log n)$ on the number of advice bits to produce an optimal solution for any given graph. For particular graph classes, e.g., with bounded degree or a restricted edge weight function, we prove that the upper bound can be drastically reduced; e.g., $5(n-1)$ advice bits allow to compute an optimal result if the weight function is the Euclidean distance; if the graph is complete, even a logarithmic number suffices. Some of these results make use of the optimality of Kruskal's algorithm for the offline setting. We also study the trade-off between the number of advice bits and the achievable competitive ratio. To this end, we perform a reduction from another online problem to obtain a linear lower bound on the advice complexity for any near-optimal solution. Using our results from the advice complexity finally allows us to give a lower bound on the expected competitive ratio of any randomized online algorithm for the problem.

1 Introduction

Computing problems are called *online* if the input arrives gradually in consecutive time steps. An *online algorithm* has to create parts of the definite output while only knowing a prefix of the input in the current time step [8]. A broad subclass are online graph problems, i.e., online problems where the input corresponds to some graph that is revealed in an online fashion. In this paper, the mainly studied model reveals the vertices of an underlying graph one after the other; together with every vertex, all edges are shown that connect this vertex to all other vertices that are already known to the online algorithm. Sleator

This work was partially supported by SNF grant 200021–146372 and by MIUR under the project "PRIN: Automi e Linguaggi Formali: Aspetti Matematici e Applicativi."

R.M. Freivalds et al. (Eds.): SOFSEM 2016, LNCS 9587, pp. 195–207, 2016.
DOI: 10.1007/978-3-662-49192-8_16

and Tarjan introduced the concept of *competitive analysis* to measure the performance of an online algorithm [23]. In this worst-case measurement, one compares the cost or gain of the solution produced by the online algorithm to the optimal one that could hypothetically be computed if the whole instance were known from the start. Here, one assumes that the input is produced by a malicious *adversary*. The ratio between the two is called the *competitive ratio* of the online algorithm; a detailed introduction is given by Borodin and El-Yaniv [8].

While competitive analysis is an extremely powerful and widely-used tool to assess the performance of online algorithms, it does not address the question of the essential parts of the input that the online algorithm is missing, i.e., the *information content* of the problem [16]. To be able to answer this question, we study the *advice complexity*. An *online algorithm with advice* has an additional resource available, the so-called *advice tape*, that may contain any kind of information about the instance at hand. The content of this tape is an infinite binary string called the *advice*, and it is written onto the tape before the computation starts by an *oracle* that sees the whole input in advance. The advice complexity then measures the number of *advice bits* that allows to achieve a certain performance.

A first model of online computation with advice was introduced by Dobrev et al. [12]. This model was then refined simultaneously by Fraigniaud et al. [14] and Hromkovič et al. [16]. The latter model, which is the one we use in this paper, was first applied to three different online problems by Böckenhauer et al. [6]. Here, in every time step, the online algorithm can query the advice tape for any number of advice bits (analogously to the model of the random tape of a randomized Turing machine). The advice complexity is the length of a maximum prefix of the advice tape that is read; this length usually depends on the input size n of the given instance.

The advice complexity has been widely applied to a large number of online problems so far including paging [6] and k-server [7,14]. In particular, this model has recently been studied for quite a number of graph problems such as different coloring problems [3,4,15,22], the independent set problem [10], the dominating set problem [9], the Steiner tree problem [2], or graph exploration [11]. Moreover, online computation with advice also has some interesting connections to randomized online computation [7,18]. This line of research tries to answer the question of how well any additional information on the input may be exploited. The crucial part is the generality of the answer that is given: the advice may encode any information, it is not restricted to a specific problem parameter or property of the input. This way, a lower bound on the advice complexity to achieve some given competitive ratio c means that it will never be possible to obtain c-competitiveness with less information, no matter what the information will actually be.

To the best of our knowledge, the advice complexity of the minimum spanning tree problem has not been studied so far. Megow et al. [20] investigated the online minimum spanning tree problem in a model that allows an online algorithm to do some recourse actions, meaning that it can perform a certain amount of edge

rearrangements. However, in this model, the algorithm has to compute a feasible spanning tree for the graph that has been presented so far in any time step. For randomly weighted graphs with edge weights that are uniformly distributed over the interval between 0 and 1, the problem was studied by Remy et al. [21]. In their model, both the algorithm and the adversary do not know the edge weights before they are presented. Tsai and Tsang investigated the competitiveness of a certain family of randomized algorithms [24]; they restricted the inputs to graphs in the Euclidean space. The minimum spanning tree problem was also considered in the setting of min-max regret [17], in which the goal is to minimize the maximal possible deviation of a given solution from optimum.

This paper is organized as follows. In Sect. 2, we formally introduce the model of online computation with advice and the online minimum spanning tree problem. In Sect. 3, we study the advice complexity of optimal online algorithms with advice for different graph classes. In Subsect. 3.1, we give an asymptotically tight bound of $\Theta(n \log n)$ to compute an optimal solution for general graphs. In Subsect. 3.2, we present a linear lower bound for graphs that have three different edge weights. Here, we also study a different model of online computation where the structure of the graph is known in advance, but the edge weights appear online. The interesting point of the proof is that the optimality of this online algorithm is a consequence of the optimality of Kruskal's offline algorithm. In Subsect. 3.4, we first study graphs with bounded degree. Furthermore, we prove that there is an optimal online algorithm that uses $5(n-1)$ advice bits to be optimal on graphs with a Euclidean weight function. Section 4 studies the trade-off between the number of advice bits and the competitive ratio that is achievable. We prove a linear lower bound to obtain a near-optimal competitive ratio by giving a reduction from the bit guessing problem. In Sect. 5, we extend this result to randomized algorithms. Due to space limitations, some proofs are omitted in this extended abstract.

2 Preliminaries

In this paper, we only consider the objective to minimize a given cost function.

Definition 1 (Online Minimization Problem). *An* online minimization problem *consists of a set $\mathcal{I}$ of inputs and a cost function. Every input $I \in \mathcal{I}$ is a sequence $I = (x_1, x_2, \ldots, x_n)$ of* requests. *Furthermore, a set of feasible outputs (or solutions) is associated with every I; every output is a sequence $O = (y_1, y_2, \ldots, y_n)$ of* answers. *The* cost function *assigns a positive real value* $\mathrm{cost}(I, O)$ *to every input I and any feasible output O. For every input I, we call any feasible output O for I that has smallest possible cost (i.e., that minimizes the cost function) an* optimal solution *for I.*

In what follows, we will simply write $\mathrm{cost}(I)$ instead of $\mathrm{cost}(I, O)$ as O is always clear from context, and we let $[I]_k = (x_1, \ldots, x_k)$, for $k \leq n$, be the sequence of the first k requests in I. In the settings we study, the input always corresponds to a weighted undirected graph G with a weight function ω. Throughout

this paper, the set of vertices of G is denoted by $V(G)$, and $E(G)$ denotes its set of edges; if G is clear from context, we simply write V and E. G is usually revealed to the online algorithm as follows. Let $V = \{v_1, v_2, \ldots, v_n\}$; then v_i is presented in time step i together with all edges $\{v_i, v_j\} \in E$ for which $j < i$, i.e., edges that are connected to vertices that have already been revealed in previous time steps. After every newly revealed vertex, an online algorithm for the *online minimum spanning tree problem* (OMST for short) must choose some of the newly revealed edges that are part of the solution; this decision is final. Note that we do not require the set of chosen edges to be a spanning tree of the already revealed vertices in the intermediate steps. To correctly capture the online environment, the number of vertices is not known to the online algorithm in advance.

Next, we formally define online algorithms with advice.

Definition 2 (Online Algorithm with Advice). *Consider an input I of an online minimization problem. An* online algorithm ALG with advice *computes the output sequence $\text{ALG}^{\phi}(I) = (y_1, y_2, \ldots, y_n)$ such that y_i is computed from $\phi, x_1, x_2, \ldots, x_i$, where ϕ is the content of the advice tape, i.e., an infinite binary sequence.* ALG *is* c-competitive with advice complexity $b(n)$ *if there exists a non-negative constant α such that, for every n and for any input sequence I of length at most n, there exists some advice string ϕ such that* $\text{cost}(\text{ALG}^{\phi}(I)) \leq c \cdot \text{cost}(\text{OPT}(I)) + \alpha$ *and at most the first $b(n)$ bits of ϕ have been accessed during the computation of the solution $\text{ALG}^{\phi}(I)$. If the above inequality holds with $\alpha = 0$, we call* ALG strictly c-competitive with advice complexity $b(n)$. ALG *is called* optimal *if it is strictly* 1*-competitive.*

For the sake of an easier notation, we omit ϕ as it is always clear from context. Moreover, we denote the binary logarithm of a natural number x simply by $\log x$.

3 Optimality

In this section, we show that any online algorithm with advice that solves the OMST problem optimally on general graphs needs to read $\Omega(n \log n)$ advice bits. We also provide an online algorithm that achieves optimality with $O(n \log n)$ advice bits. We will then discuss the problem for input graphs that are somehow limited (such as special graph classes, bounded edge weights, or bounded degree).

3.1 General Graphs

First, we provide an online algorithm that gets as advice the parent of every newly revealed vertex with respect to an optimal spanning tree.

Theorem 1. *There exists an online algorithm with advice for the OMST problem that uses $n\lceil \log n \rceil + 2\lceil \log(\lceil \log n \rceil + 1) \rceil$ advice bits and that is optimal on every instance of length n.*

Proof. The oracle computes an optimal offline solution T. At each request v, the algorithm asks for the index of the parent of v in T (if the requested vertex is the arbitrarily chosen root of T, then the oracle encodes v itself). This takes $n\lceil \log n \rceil$ bits of advice in total. In order to know how many advice bits it should read after each request, the algorithm needs to ask first for the number $\lceil \log n \rceil$, which is encoded using a prefix code (such as Elias' delta-code [13]) at the beginning of the advice string with $2\lceil \log(\lceil \log n \rceil + 1) \rceil$ bits. □

Although the above online algorithm uses a straightforward approach, this upper bound is asymptotically tight; in fact, we can prove the lower bound even on a very restricted class of graphs.

Theorem 2. *Any online algorithm with advice for the OMST problem on bipartite graphs needs to read at least* $\log(((n-1)/2)!) \in \Omega(n \log n)$ *advice bits to be optimal on every instance of length* n.

Proof. Let $n = 2k + 1$ be an odd number. We consider a bipartite graph G having vertices $\{v_1, v_2, \ldots, v_k, u_1, u_2, \ldots, u_k, w\}$ such that, for each $1 \leq i \leq k$, the vertex u_i is connected to w through an edge of weight 1 and, for $i \leq j \leq k$, u_i is connected to v_j through an edge of weight $k - i + 1$. Clearly, such a graph has a unique minimum spanning tree, as shown in Fig. 1. As set of instances $\mathcal{I}$, we consider all online presentation of G such that first, a permutation of the vertices $v_1, v_2, \ldots, v_k$ is presented, then the vertices $u_1, u_2, \ldots, u_k, w$ are presented in this order. These instances differ only in the order of the first k vertices.

Suppose towards contradiction that there is an algorithm ALG that optimally solves OMST on any instance of $\mathcal{I}$ using less than $\log(((n-1)/2)!)$ bits of advice. This implies that there are two different instances, with two different permutations σ_1 and σ_2 of the vertices $v_1, v_2, \ldots, v_k$, which receive the same advice string. Let v_i be the first vertex that is not at the same position in σ_1 and σ_2, say at position s in σ_1 and position t in σ_2. Up to and including the $(k+i)$-th time step, the input looks the same for ALG. However, in time step $k + i$, ALG

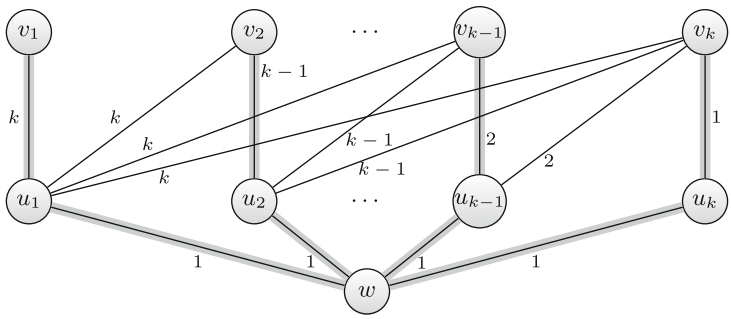


Fig. 1. Graph structure used in the proof of Theorem 2; gray edges denote the (unique) optimal solution.

has to choose the edge that connects u_i to the vertex that was presented in time step s (t, respectively) in the case of σ_1 (σ_2, respectively). But since the input looked exactly the same to the algorithm so far, it cannot distinguish between these two cases, so it will not output the optimal solution for at least one of these instances. □

3.2 Graphs with Bounded Edge Weights

We now consider graphs with a bounded number of different edge weights. The next theorem shows that, even for only those different weights, still a linear number of advice bits is needed.

Theorem 3. *Any online algorithm with advice that solves the OMST problem on graphs with* 3 *or more different edge weights needs to read at least* $n-2$ *advice bits to be optimal on every instance of length* n*.* □

Next, we prove a result for graphs with two different edge weights, which will come in handy when considering different graph classes. However, for the following result, the online setting differs in the following sense. The structure of the graph is known to the online algorithm in advance, and only the concrete edge weights are revealed in the respective time steps.

Theorem 4. *Let* $G = (V, E, \omega)$ *be a connected graph with two different edge weights* a *and* b*, where* $0 \leq a < b$*. For the online problem in which the structure of* G *is fully known to the algorithm and only the edge weights are presented online, there exists an optimal online algorithm* ALG *that uses no advice.*

Proof sketch. For each instance I which is an online presentation of the graph $G = (V, E, \omega)$, let $m = |E|$, let e_i be the edge presented to ALG at time step i, and let w_i be the weight of e_i. The algorithm works on I as follows: if $w_i = a$, then ALG includes e_i in the partial solution if and only if e_i does not create a cycle; if $w_i = b$, then it includes $e_i = \{u, v\}$ in the partial solution if and only if, for all the other paths connecting u and v, ALG has already rejected one of the edges composing that path. The resulting graph T_{ALG} is clearly a spanning tree whose optimality is easy to show. □

Theorem 4 implies logarithmic upper bounds on the advice complexity for optimal online algorithms for complete and complete bipartite graphs. Indeed, with $\max\{2, \lceil \log n \rceil + 2\lceil \log \lceil \log n \rceil \rceil\}$ bits, the input size can be encoded in a self-delimiting way, and an online algorithm can then treat a complete graph instance as if it were presented in the online model used in the proof of Theorem 4.

Corollary 1. *For all complete graphs* G *of size* n *with edge weights in* $\{a, b\}$*, with* $0 \leq a < b$*, there exists an optimal online algorithm with advice that solves the OMST problem for* G *and uses* $\max\{2, \lceil \log n \rceil + 2\lceil \log \lceil \log n \rceil \rceil\}$ *bits of advice.* □

Next, we show that the bound from Corollary 1 is essentially tight.

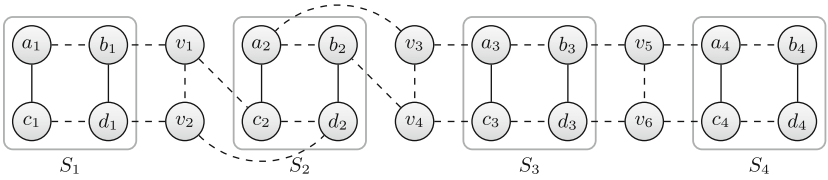


Fig. 2. An instance of the graph class $\mathcal{G}_{22}$ that is used in the proof of Theorem 6. Dashed edges have weight 1, solid edges have weight 2. First, the four isolated squares S_1, S_2, S_3 and S_4 are presented, then the remaining vertices $v_1, v_2, \ldots, v_6$ in this order. To find an optimal minimum spanning tree, ALG has to choose exactly one edge of weight 2 in square S_2, which is oriented horizontally in the ladder, but no edge of weight 2 in the vertically oriented squares S_1, S_3 and S_4.

Theorem 5. *Any online algorithm with advice that solves the OMST problem optimally for complete graphs with at least two different edge weights needs at least* $\log(\lfloor n/2 \rfloor)$ *bits of advice to be optimal on every input sequence of size at most* n.

Proof sketch. For every even $2 \leq m \leq n$, we consider the complete graph instance $\hat{G}_m$ with edge weights $\{1, 2\}$ defined as follows: $\hat{G}_m$ has m vertices and, by calling v_j the vertex presented at time step j, each v_j with j odd (even, respectively) is connected with all vertices v_i, with $i < j$, by an edge of weight 1 (2, respectively). By using the partition tree technique introduced in [1], we can show that any algorithm needs a different advice string for each $\hat{G}_m$ to be optimal. The intuitive idea is that any algorithm needs to know when the end of the input is reached, as only in the final time step, it has to choose an edge of weight b. □

With a similar technique, we can obtain an analogous upper and lower bounds for the case of complete bipartite graphs.

3.3 Ladders

We now restrict our attention to a special class of bipartite graphs, namely ladders, which can be defined as the Cartesian product of two path graphs, one of which has only one edge. Despite of this simple structure, we show that such graphs still require linear advice, even for only two different edge weights.

Theorem 6. *For ladders with two different edge weights, any online algorithm with advice for the OMST problem needs at least* $\lfloor \frac{n+2}{6} \rfloor$ *advice bits to be optimal on every input sequence of length* n.

Proof sketch. Let n be an arbitrary natural even number. We now provide a graph class $\mathcal{G}_n$ that contains $2^{\lfloor \frac{n+2}{6} \rfloor}$ graphs of size n and show that, for any two graphs in $\mathcal{G}_n$, ALG needs different advice strings. We define $\mathcal{G}_n$ as follows: For

any graph $G \in \mathcal{G}_n$, first $\lfloor \frac{n+2}{6} \rfloor$ isolated squares $S_1, \ldots, S_{\lfloor \frac{n+2}{6} \rfloor}$ are presented. Then, the squares are connected to a ladder with the remaining $n - 4 \cdot \lfloor \frac{n+2}{6} \rfloor$ vertices. All edges incident to these vertices have weight 1. Any square S_i can either be oriented horizontally or vertically in the ladder (see Fig. 2).

As there are $\lfloor \frac{n+2}{6} \rfloor$ squares in every graph $G \in \mathcal{G}_n$ and all squares can be oriented in two ways, there are $2^{\lfloor \frac{n+2}{6} \rfloor}$ graphs in $\mathcal{G}_n$, therefore it suffices to show that ALG needs different advice strings for any two graphs of $\mathcal{G}_n$. □

Theorem 7. *For ladders with two different edge weights, there exists an online algorithm with advice for the OMST problem that is optimal on every input sequence of even length $n \geq 2$ and reads at most $\lceil \frac{3}{4}n \rceil + 4\lceil \log n \rceil + 2\lceil \log \lceil \log n \rceil \rceil$ advice bits.*

Proof sketch. The algorithm reads the size of the input, the positions of the four corner vertices, and, for each vertex, if it lies on the top or the bottom line of the ladder, if needed. □

3.4 Further Special Graph Classes

We now analyze our problem on graphs of bounded degree.

Theorem 8. *For graphs with degree at most g, there exists an online algorithm with advice that solves the OMST problem and uses at most $(n-1)\lceil \log g \rceil + \max\{2, \lceil \log n \rceil + 2\lceil \log \lceil \log n \rceil \rceil\}$ advice bits to be optimal on every instance of length n.* □

For graphs with degree 3 and 4 we obtain asymptotically matching lower bounds.

We complement our results on special graph classes by showing that also in a geometric setting, where the vertices are points in the Euclidean plane and the edge weights are their distance, we can compute an optimal solution using linear advice.

Theorem 9. *For Euclidean graphs, there exists an online algorithm* ALG *with advice that solves the OMST problem and uses at most $5(n-1)$ advice bits to be optimal on every instance of length n.* □

4 Competitiveness

In this section, we analyze the trade-off between the advice complexity and the competitiveness of online algorithms for the OMST problem. We recall that, as proved in [20], for general graphs no deterministic online algorithm can be competitive. If we have edge weights bounded by a constant k, then the greedy algorithm is clearly k-competitive on every input, since any spanning tree on a graph with n vertices has weight at least $n-1$ and at most $k(n-1)$. If the degree of the graph is bounded by 3, we can even prove a better competitive ratio without advice for a bounded number of edge weights.

Theorem 10. *Let $G = (V, E)$ be a graph with maximum degree 3 where $\omega\colon E \to W$ is a weight function that maps edges into a bounded set W of weight values. Let a denote the minimum and b the maximum element in W. Then, the greedy algorithm* Greedy *has a competitive ratio of at most $((a+b)\frac{n}{2} - 2a + b)/(a(n-1))$ for the OMST problem on G.* □

In Subsect. 3.4, we have shown that any online algorithm with advice needs a linear amount of advice to be optimal for the OMST problem on complete graphs with three different edge weights. We now show what an algorithm with logarithmic advice can achieve in this setting. For simplicity, we will only discuss the case with the three edge weights 1, 2 and 3. Note that similar results can be obtained using the same proof idea, and any 3 different edge weights.

Theorem 11. *There exists an online algorithm* Alg *with advice for the OMST problem on complete graphs with edge weights $1, 2$ and 3 that reads $\max\{2, \lceil \log n \rceil + 2\lceil \log \lceil \log n \rceil \rceil\} + 1$ bits of advice on an input of length n and achieves a strict competitive ratio of $(5 - \sqrt{5})/2 \approx 1.382$.*

Proof sketch. For each instance of size n, the oracle encodes n on the advice tape with $\max\{2, \lceil \log n \rceil + 2\lceil \log \lceil \log n \rceil \rceil\}$, then uses an additional bit to suggest one of the following two strategies:

1. in the first $n-1$ steps, choose greedily all edges of weight 1 which do not close a cycle, then in the last step choose the cheapest edges needed to connect all the currently separated components in the partial solution,
2. in the first $n-1$ steps choose greedily all edges of weight 1 and 2 without closing cycles, then connect the components in the last step with the cheapest possible edges.

Given the solution T obtained with strategy 2, consider the tree T' constructed in the following way: whenever there exists an edge e_1 of weight 1 in the input graph that would create a cycle in T that contains an edge e_2 of weight 2, replace e_2 with e_1. We call *suboptimal* all the edges of weight 2 removed in this process. The oracle suggests the second strategy if and only if the number of suboptimal edges chosen with this strategy is less than $p(n-1)$, for a suitable $0 < p < 1$. □

We now provide a linear lower bound that even holds for the case that the maximum degree is 3. To this end, we use a general technique, namely reducing the *bit guessing problem with known history*, which was introduced by Böckenhauer et al. [5], to the OMST problem.

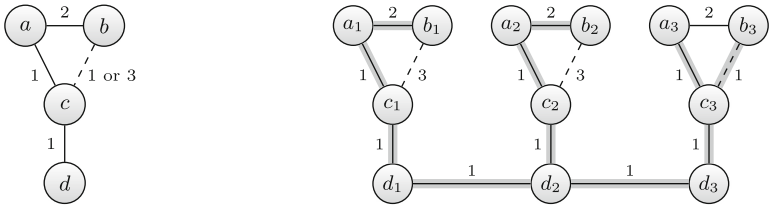
Definition 3 (Bit Guessing with Known History). *The* bit guessing problem with known history *(BGKH) is the following online minimization problem. The input $I = (n, d_1, d_2, \ldots, d_n)$ consists of a natural number n and the bits $d_1, d_2, \ldots, d_n$, that are revealed one by one. The online algorithm* Alg *computes the output sequence $\textsc{Alg}(I) = y_1 y_2 \ldots y_n$, where $y_i = f(n, d_1, \ldots, d_{i-1}) \in \{0, 1\}$, for some computable function f. The algorithm is not required to respond with any output in the last time step. The cost of a solution $\textsc{Alg}(I)$ is the number of wrongly guessed bits, i.e., the Hamming distance $\mathrm{Ham}(d, \textsc{Alg}(I))$ between $d = d_1 d_2 \ldots d_n$ and $\textsc{Alg}(I)$.*

We start by formally describing the reduction on a specific class of instances for the OMST problem.

Lemma 1. *Let s be any BGKH instance of length n'. Let $\delta \in \mathbb{R}$ with $1/2 \leq \delta \leq 1$ be such that any online algorithm with advice for BGKH reading b advice bits can guess at most $\delta n'$ bits of s correctly. Then, no online algorithm* Alg *with advice for the corresponding OMST instance G_s reads b advice bits and achieves*

$$\text{cost}(\text{Alg}(G_s)) < \text{cost}(\text{Opt}(G_s)) + (1-\delta)n',$$

where $\text{cost}(\text{Opt}(G_s))$ *is the cost of an optimal minimum spanning tree of G_s.*



(a) Basic component used to build the graphs

(b) The graph G_s corresponding to the bit string $s = 110$. The dashed edge of the i-th component in G_s has weight 1 if the i-th bit of s has value 0 and weight 3 otherwise.

Fig. 3. Graph construction used in the proof of Theorem 1. Edges that depend on s are dashed.

Proof sketch. For every bit string $s = s_1 s_2 \dots s_{n'}$ of length n', which is an instance of the BGKH problem, we construct a corresponding instance G_s for the OMST problem as follows: for every bit s_i of s, we build a component as illustrated in Fig. 3a, where the edge $\{b, c\}$ has weight 1 if and only if s_i has value 0 and weight 3 otherwise. Then, we complete these n' components to a connected graph by adding edges of weight 1 between the vertices d_i and d_{i+1}, for all $1 \leq i < n'$. As an example, the graph G_{110} is shown in Fig. 3b. The vertex presentation order is $a_1, b_1, c_1, d_1, a_2, \dots, c_{n'}, d_{n'}$. □

Using Lemma 1, we can now proceed to prove the following lower bound on the advice complexity for any c-competitive algorithm for OMST.

Theorem 12. *There is no online algorithm with advice that is strictly c-competitive, $1 \leq c \leq 11/10$, for the OMST problem on graphs with maximum degree 3, maximum edge weight 3 and n vertices, $n = 4n'$ for some $n' \in \mathbb{N}$, that reads less than $(1 + (5c-5)\log(5c-5) + (6-5c)\log(6-5c))\frac{n}{4}$ bits of advice.* □

In the case of unbounded edge weights, we can extend the linear lower bound from Theorem 12 to consider competitive ratios up to $\frac{5}{4}$ as follows: Instead of the edge weights 1, 2, and 3, we choose weights $1, k+1$, and $2k+1$, for arbitrarily large k. Then, Lemma 1 can be proven analogously and we get that ALG has to read at least $\text{cost}(\text{OPT}(G_s)) + k \cdot (1-\delta)n'$ bits of advice. With the same calculations as above, we can show that the competitive ratio of ALG is $c \geq 1+(1-\delta)\cdot\frac{k}{2+2(k+1)} \overset{k\to\infty}{\longrightarrow} 1+(1-\delta)\frac{1}{2}$. As a result, we get the following corollary.

Corollary 2. *In the case where edge weights are unbounded, there is no online algorithm with advice that is c-competitive,* $1 \leq c \leq \frac{5}{4}$*, for the OMST problem on graphs with maximum degree* 3 *and n vertices,* $n = 4n'$ *for some* $n' \in \mathbb{N}$*, that reads less than* $(1+(2c-2)\log(2c-2)+(3-2c)\log(3-2c))\frac{n}{4}$ *bits of advice.* □

5 Randomized Online Algorithms

In this section, we give a lower bound on the competitive ratio achievable by any randomized online algorithm. Our proof is based on the following result.

Lemma 2. (Böckenhauer et al. [7]). *Consider an online minimization problem U, and let* $\mathcal{I}(n)$ *be the set of all possible inputs of length n and* $I(n) := |\mathcal{I}(n)|$*. Furthermore, suppose that there is a randomized online algorithm for U with worst-case expected competitive ratio at most E. Then, for any fixed* $\varepsilon > 0$*, it is possible to construct an online algorithm with advice that uses at most*

$$\log n + 2\log\log n + \log\left(\log I(n)/\log(1+\varepsilon)\right) + c$$

advice bits, for a constant c, and achieves a competitive ratio of $(1+\varepsilon)E$*.* □

In Theorem 12, we constructed, for each integer n, a set $\mathcal{I}(n)$ of $2^{\frac{n}{4}}$ instances (depicted in Fig. 3) such there is no $\frac{11}{10}$-competitive online algorithm that uses $o(n)$ advice bits. This, together with Lemma 2, implies the following result.

Theorem 13. *For arbitrarily small* $\delta > 0$*, every randomized algorithm (using an arbitrary number of random bits) for the OMST problem on graphs with maximum degree* 3 *and maximum edge weight* 3 *has a worst-case expected competitive ratio of at least* $\frac{11}{10}(1-\delta)$ *on sufficiently large instances.* □

Acknowledgments. The authors would like to thank Juraj Hromkovič for enlightening discussions.

References

1. Barhum, K., Böckenhauer, H.-J., Forišek, M., Gebauer, H., Hromkovič, J., Krug, S., Smula, J., Steffen, B.: On the power of advice and randomization for the disjoint path allocation problem. In: Geffert, V., Preneel, B., Rovan, B., Štuller, J., Tjoa, A.M. (eds.) SOFSEM 2014. LNCS, vol. 8327, pp. 89–101. Springer, Heidelberg (2014)

2. Barhum, K.: Tight bounds for the advice complexity of the online minimum steiner tree problem. In: Geffert, V., Preneel, B., Rovan, B., Štuller, J., Tjoa, A.M. (eds.) SOFSEM 2014. LNCS, vol. 8327, pp. 77–88. Springer, Heidelberg (2014)
3. Bianchi, M.P., Böckenhauer, H.-J., Hromkovič, J., Keller, L.: Online coloring of bipartite graphs with and without advice. In: Gudmundsson, J., Mestre, J., Viglas, T. (eds.) COCOON 2012. LNCS, vol. 7434, pp. 519–530. Springer, Heidelberg (2012)
4. Bianchi, M.P., Böckenhauer, H.-J., Hromkovič, J., Krug, S., Steffen, B.: On the advice complexity of the online $L(2,1)$-coloring problem on paths and cycles. In: Du, D.-Z., Zhang, G. (eds.) COCOON 2013. LNCS, vol. 7936, pp. 53–64. Springer, Heidelberg (2013)
5. Böckenhauer, H.-J., Hromkovič, J., Komm, D., Krug, S., Smula, J., Sprock, A.: The string guessing problem as a method to prove lower bounds on the advice complexity. Theoret. Comput. Sci. **554**, 95–108 (2014)
6. Böckenhauer, H.-J., Komm, D., Královič, R., Královič, R., Mömke, T.: On the advice complexity of online problems. In: Dong, Y., Du, D.-Z., Ibarra, O. (eds.) ISAAC 2009. LNCS, vol. 5878, pp. 331–340. Springer, Heidelberg (2009)
7. Böckenhauer, H.-J., Komm, D., Královič, R., Královič, R.: On the advice complexity of the k-server problem. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part I. LNCS, vol. 6755, pp. 207–218. Springer, Heidelberg (2011)
8. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press, New York (1998)
9. Boyar, J., Favrholdt, L.M., Kudahl, C., Mikkelsen, J.W.: The advice complexity of a class of hard online problems. CoRR abs/1408.7033 (2014)
10. Dobrev, S., Královič, R., Královič, R.: Independent set with advice: the impact of graph knowledge. In: Erlebach, T., Persiano, G. (eds.) WAOA 2012. LNCS, vol. 7846, pp. 2–15. Springer, Heidelberg (2013)
11. Dobrev, S., Královič, R., Markou, E.: Online graph exploration with advice. In: Even, G., Halldórsson, M.M. (eds.) SIROCCO 2012. LNCS, vol. 7355, pp. 267–278. Springer, Heidelberg (2012)
12. Dobrev, S., Královic, R., Pardubská, D.: Measuring the problem-relevant information in input. RAIRO ITA **43**(3), 585–613 (2009)
13. Elias, P.: Universal codeword sets and representations of the integers. IEEE Trans. Inf. Theory **21**(2), 194–203 (1975)
14. Emek, Y., Fraigniaud, P., Korman, A., Rosén, A.: Online computation with advice. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009, Part I. LNCS, vol. 5555, pp. 427–438. Springer, Heidelberg (2009)
15. Forišek, M., Keller, L., Steinová, M.: Advice complexity of online coloring for paths. In: Dediu, A.-H., Martín-Vide, C. (eds.) LATA 2012. LNCS, vol. 7183, pp. 228–239. Springer, Heidelberg (2012)
16. Hromkovič, J., Královič, R., Královič, R.: Information complexity of online problems. In: Hliněný, P., Kučera, A. (eds.) MFCS 2010. LNCS, vol. 6281, pp. 24–36. Springer, Heidelberg (2010)
17. Kasperski, A.: Discrete Optimization with Interval Data: Minmax Regret and Fuzzy Approach. Springer, Heidelberg (2008)
18. Komm, D., Královič, R.: Advice complexity and barely random algorithms. Theor. Inf. Appl. (RAIRO) **45**(2), 249–267 (2011). IEEE Computer Society
19. Kruskal Jr., J.B.: On the shortest spanning subtree of a graph and the traveling salesman problem. Proc. Am. Math. Soc. **7**(1), 48–50 (1956)

20. Megow, N., Skutella, M., Verschae, J., Wiese, A.: The power of recourse for online MST and TSP. In: Czumaj, A., Mehlhorn, K., Pitts, A., Wattenhofer, R. (eds.) ICALP 2012, Part I. LNCS, vol. 7391, pp. 689–700. Springer, Heidelberg (2012)
21. Remy, J., Souza, A., Steger, A.: On an online spanning tree problem in randomly weighted graphs. Comb. Probab. Comput. **16**(1), 127–144 (2007). Cambridge University Press
22. Seibert, S., Sprock, A., Unger, W.: Advice complexity of the online coloring problem. In: Spirakis, P.G., Serna, M. (eds.) CIAC 2013. LNCS, vol. 7878, pp. 345–357. Springer, Heidelberg (2013)
23. Sleator, D.D., Tarjan, R.E.: Amortized efficiency of list update and paging rules. Commun. ACM **28**(2), 202–208 (1985)
24. Teh Tsai, Y., Yi Tang, C.: The competitiveness of randomized algorithms for online Steiner tree and on-line spanning tree problems. Inf. Process. Lett. **48**(4), 177–182 (1993). Elsevier