The Power of Recourse for Online MST and TSP

Nicole Megow[1], Martin Skutella[1⋆], José Verschae[2⋆⋆], and Andreas Wiese[3⋆⋆⋆]

[1] Department of Mathematics, Technische Universität Berlin, Germany.
{nmegow,skutella}@math.tu-berlin.de
[2] Departamento de Ingeniería Industrial, Universidad de Chile, Chile.
jverscha@ing.uchile.cl
[3] Department of Computer and System Sciences, Sapienza University of Rome, Italy.
wiese@dis.uniroma1.it

Abstract. We consider the online MST and TSP problems with recourse. The nodes of an unknown graph with metric edge cost appear one by one and must be connected in such a way that the resulting tree or tour has low cost. In the standard online setting, with irrevocable decisions, no algorithm can guarantee a constant competitive ratio. In our model we allow recourse actions by giving a limited budget of edge rearrangements per iteration. It has been an open question for more than 20 years if an online algorithm equipped with a constant (amortized) budget can guarantee constant-approximate solutions [7].

As our main result, we answer this question affirmatively in an amortized setting. We introduce an algorithm that maintains a nearly optimal tree when given constant amortized budget. In the non-amortized setting, we specify a promising proof technique and conjecture a structural property of optimal solutions that would prove a constant competitive ratio with a single recourse action. It might seem rather optimistic that such a small budget should be sufficient for a significant cost improvement. However, we can prove such power of recourse in the offline setting in which the sequence of node arrivals is known. Even this problem prohibits constant approximations if there is no recourse allowed. Surprisingly, already a smallest recourse budget significantly improves the performance guarantee from non-constant to constant.

Unlike in classical TSP variants, the standard double-tree and shortcutting approach does not give constant guarantees in the online setting. However, a non-trivial robust shortcutting technique allows to translate online MST results into TSP results at the loss of small factors.

1 Introduction

In the *online Minimum Spanning Tree* (MST) problem and *online Traveling Salesman Problem* (TSP) we aim at constructing low-cost spanning trees, resp.

⋆ Supported by the DFG Research center Matheon in Berlin.
⋆⋆ Supported by the Berlin Mathematical School (BMS) and Nucleo Milenio Información y Coordinación en Redes ICM/FIC P10-024F.
⋆⋆⋆ Supported by the German Academic Exchange Service (DAAD).

tours, for an unknown graph that is revealed online. In each iteration a new node becomes known, together with all connections to previously revealed nodes, and an algorithm must make an irrevocable decision on how to connect it. When a tree must be maintained, then the decision concerns inserting one edge, whereas for maintaining a tour, an edge must be removed and two new ones inserted. Such problems appear naturally in applications related to multicast routing in multimedia distribution systems, video-conferencing, software delivery, and groupware [12, 13]. They have been studied extensively, in particular online MST and Steiner tree variants. Here, constant competitive ratios are not achievable; the best possible performance ratio is $\Theta(\log t)$, where t is the number of iterations [7].

However, in many of the above-mentioned applications it is possible to adapt solutions in some limited way when the node set changes [14, 16]. Clearly, such *recourse actions* allow for improved solutions. However, an increasing number of adaptations—in particular, a complete reconstruction of solutions—might not be feasible or may cause unacceptable additional cost, and is therefore not desirable. Our goal is to understand the tradeoff between the amount of adaptivity and the quality of solutions. As a main problem, we want to determine the amount of recourse that is necessary to allow for provably near-optimal solutions. Looking from a different perspective, we construct solutions that satisfy some adequate concept of *robustness*, where we measure robustness by the (amortized) *recourse budget* that is necessary to guarantee solutions of a particular quality.

More precisely, we consider the *online MST* and *online TSP with recourse*: An undirected complete metric graph is revealed online node by node. In each iteration a new node becomes known and all edges (with corresponding costs) to previously arrived nodes are revealed. The objective is to construct in each iteration a low-cost spanning tree, resp. tour, of the revealed vertices, without any assumption on the vertices that might arrive in future. We measure the quality of the solution sequence with standard competitive analysis by comparing online solutions to the offline optimum, i. e., the MST, resp. TSP tour, on the currently known subgraph. We control the amount of recourse, i. e., how much the solution changes along iterations, by limiting the number *rearrangements*. More precisely, we give a budget for the number of edges that can be inserted in each iteration. We say that an algorithm needs budget k if the number of inserted edges in each iteration is bounded by k. Similarly, the algorithm uses an *amortized budget* k if up to iteration t the total number of inserted edges is at most $t \cdot k$. Notice that by this definition, the standard online MST problem equips algorithms with a budget of 1 whereas the online TSP without recourse requires a minimum budget of 2.

It has been a longstanding open question if constant budget suffices to maintain constant-approximate solutions [7, 2]. In this paper we not only answer this question affirmatively, but we also show the surprising fact that already a smallest amount of recourse suffices to significantly improve the solution.

Related Work. The online MST and Steiner Tree problems have been studied intensively. The best possible competitive ratio for online algorithms is known to be $\Theta(\log t)$, where t is the number of iterations [7]. A simple greedy algorithm

that connects a new node to the current tree through a shortest edge achieves this bound. Even in the special case of Euclidean distances, there is a lower bound of $\Omega(\frac{\log t}{\log \log t})$ on the competitive guarantee of any online algorithm [1].

Unlike in stochastic programming [3], where recourse actions are an important concept when optimizing under limited information, the literature on recourse models for online optimization seems rather sparse. Regarding our model, we are aware only of the work by Imase and Waxman [7] that deals with the online Minimum Steiner Tree problem with recourse (or *Dynamic Steiner Tree*). The model they introduce is slightly more general as nodes do not only arrive but may also *depart* from the terminal set. For this setting, they give an algorithm that is 8-competitive and performs in t iterations at most $O(t^{3/2})$ rearrangements. This translates by our definitions to an algorithm that requires an amortized recourse budget of $O(t^{1/2})$. In the more restricted setting considered in this paper, with no node leaving the terminal set, their algorithm achieves a competitive guarantee of 4. Furthermore, for the online MST problem, their algorithm is even 2-competitive when given the mentioned non-constant amortized budget. The question if there is a constant-competitive algorithm that uses only constant (amortized) budget has been left open, but was conjectured to be true.

Interestingly, the maximization variant of our problem, i. e., finding a sequence of spanning trees of *maximum* cost, does not admit constant-competitive algorithms with a low budget [15]. However, this changes when comparing the solution to an optimal solution under the same limited recourse budget, instead of optimal MSTs. For this case, there is a 2-competitive algorithm, even in the general context of matroids and a constant competitive ratio for the intersection of a constant number of matroids [15].

A related online MST variant has been studied in [6]. Here, in each iteration the cost of some edge increases or decreases by one. The task is to maintain a sequence of *optimal* MSTs with the goal to minimize the number of rearrangements. They give a best possible deterministic algorithm that is $O(t^2)$-competitive and a randomized algorithm with expected competitive ratio $O(t \cdot \log t)$.

The TSP is one of the most prominent problems in combinatorial optimization.Despite a remarkable recent progress, the best known approximation algorithm for the offline metric TSP is still the classic 3/2-approximation by Christofides [4]. In the online setting, when the nodes appear over time as the salesman traverses its tour, constant-competitive algorithms are known [8]. In a different online TSP model, related to graph exploration, where nodes are revealed when the salesman moves to one of its neighbors, constant-competitive algorithms are known even without any assumption on the cost function [9, 11]. Both online models clearly differ from our setting.

Our Contribution.[4] Our main contribution, presented in Sect. 3, is an online algorithm for the online MST problem with recourse that is $(1+\varepsilon)$-competitive, for any $\varepsilon > 0$, when given an amortized budget of $O(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$. This is the first and significant improvement on a 2-competitive algorithm with non-constant budget

[4] Due to space limitations for many proofs and details we refer to the full version of this paper (see also [15]).

by Imase and Waxman [7]. We complement our result by showing that any $(1+\varepsilon)$-competitive algorithm for the online MST problem needs an amortized budget of $\Omega(\frac{1}{\varepsilon})$. Thus, our algorithm is best possible up to logarithmic factors. Using a standard argument, we immediately obtain a $(2+\varepsilon)$-competitive algorithm for the online Steiner Tree problem with the same amortized budget.

Our algorithm is simple and easy to implement, but it captures subtleties in the structure of the problem that allow the improved analysis. Similarly to the algorithm proposed in [7], we implement the following natural idea: when a new node appears, we (i) connect it to its closest neighbor, and (ii) iteratively perform edge swaps if they yield a sufficient improvement of the solution. The key difficulty when implementing this idea is to balance the number of swaps and the cost of the solution. As our crucial refinement of this approach, we introduce two *freezing rules* that effectively avoid performing unnecessary swaps. The first rule prevents from removing edges whose cost is very small. The second rule is more subtle and prohibits an edge swap if the removed edge can be traced back to a subgraph whose MST has negligible cost compared to the current MST.

Our results imply that amortized budget is much more powerful than its non-amortized counterpart. Indeed, we contrast our findings for constant amortized budget with a simple example showing that no online algorithm can be $(2-\varepsilon)$-competitive, for any $\varepsilon > 0$, if it uses non-amortized constant budget. It is, however, an important and longstanding open question [2, 7] whether there exists a constant-competitive algorithm with constant budget. In Sect. 4, we contribute towards an affirmative answer of this question as follows: We consider a simple online greedy algorithm with budget 2 that is similar to one given in [7]. We state a structural condition on the behavior of optimal solutions that guarantees this algorithm to be constant competitive. We conjecture that this condition is satisfied for every input sequence. We believe that this conjecture is an important step towards understanding the problem structure and will foster further research.

It might seem rather optimistic that a single extra rearrangement should be sufficient to yield a factor-$(\log n)$ improvement in the competitive ratio. To support our conjecture, we study the problem in the case of full information, i. e., the offline variant in which the input sequence and the cost function are known in advance. Even in this case, any algorithm with unit budget has a competitive factor in $\Omega(\log t)$. However, allowing just one additional edge insertion leads to a significant improvement. We give a polynomial time 14-competitive algorithm with budget 2. Furthermore, we show how to obtain a constant competitive ratio with "almost" unit budget, i. e., we allow one additional swap every k iterations, for some constant k. These two offline results prove that only a smallest relaxation of the unit-budget restriction may lead to significantly better solutions.

A very natural approach to solve the online TSP with recourse is to combine the algorithms proposed for the online MST problem with the folkloric classical double-tree and shortcutting technique [10]. Indeed, with this technique most offline variants of TSP are equivalent to MST from an approximation point of view and performance guarantees differ only in a factor of 2. Hence, one might be tempted to assume that the same conversion technique applies directly to the

online model with recourse. However, we observe that this is not true. We give examples in which two trees differ in just a single edge, but the standard shortcutting technique leads to completely different tours, no matter which Eulerian walk on the doubled tree edges is chosen. In Sect. 5 we overcome this difficulty by introducing a *robust* variant of the shortcutting technique: we choose the Eulerian tour in a specific way and keep track of which copy of a node in the Eulerian tour is visited by the TSP tour. With this robust shortcutting technique we show that any algorithm for the online MST problem with recourse can be converted to an algorithm for the online TSP by increasing the competitive ratio by a factor 2 and the budget by a factor 4.

2 Problem Definitions

An instance of the online MST problem with recourse is defined as follows. A sequence of nodes $v_0, v_1, \ldots$ arrives online one by one. In iteration $t \geq 0$, node v_t appears together with all edges $v_t v_s$ for $s \in \{0, \ldots, t-1\}$. The cost $c(e) \geq 0$ of an edge e is revealed when the edge appears. We assume that the edges are undirected and that the costs satisfy the triangular inequality, that is, $c(vw) \leq c(vz) + c(zw)$ for all nodes v, w, z. For each iteration t, the current graph is denoted by $G_t = (V_t, E_t)$ where $V_t = \{v_0, \ldots, v_t\}$ and $E_t = V_t \times V_t$, that is, G_t is a complete graph. We are interested in constructing an online sequence $T_0, T_1, T_2, \ldots$ where $T_0 = \emptyset$ and for each $t \geq 1$ tree T_t is a spanning tree of G_t. We say that the sequence needs budget k if $|T_t \setminus T_{t-1}| \leq k$ for all $t \geq 1$. A relaxed version of this concept is obtained by considering the average or *amortized* budget k, $\sum_{s=1}^{t} |T_s \setminus T_{s-1}| \leq k \cdot t$.

In online TSP with recourse, the nodes of a complete metric graph arrive in the same online fashion as described above and yield a sequence of graphs $G_0, \ldots, G_t, \ldots$. The objective, now, is to construct a sequence of TSP tours $Q_2, Q_3, \ldots$ for graphs $G_2, G_3, \ldots$ with minimum cost for each tour. We apply the same budget constraints as for trees.

We measure the performance of our online algorithms using classic competitive analysis. Let OPT_t be the cost of an MST, resp. TSP tour, of G_t, and for a given set of edges E denote $c(E) := \sum_{e \in E} c(e)$. We say that an algorithm is α-competitive for some $\alpha \geq 1$, if for any input sequence the algorithm computes a solution sequence $X_0, X_1, \ldots$ such that $c(X_t) \leq \alpha \cdot \mathrm{OPT}_t$ for each t.

3 An Online PTAS with Amortized Constant Budget

In this section we give a $(1+\varepsilon)$-competitive algorithm for the online MST problem with constant amortized budget for any $\varepsilon > 0$. This improves on a previous 2-competitive algorithm that requires non-constant amortized budget [7]. We also show that our budget bound is best possible up to logarithmic factors.

A natural approach to solve our problem is as follows. Let T_{t-1} be the tree solution in iteration $t-1$. To construct T_t, we first find the closest connection between the new node v_t and V_{t-1}, edge g_t, and initialize T_t as $T_{t-1} \cup \{g_t\}$. We

can diminish the cost of T_t by subsequently inserting a low cost edge f to T_t and removing the largest edge h in the formed cycle. Indeed, performing this swapping operation often enough will eventually turn T_t into the optimal solution, i. e., the MST. The difficulty lies in balancing the number of swaps that increases the budget, on the one hand, and the closeness of the tree to the MST, and thus, the total cost, on the other hand. We cope with this challenge by introducing two *freezing rules* that effectively avoid performing unnecessary swaps.

The intuition behind these freezing rules is as follows. Note that if at iteration t the optimal value OPT_t is much higher than OPT_s for some $s < t$, e. g., $\mathrm{OPT}_s \in O(\varepsilon \mathrm{OPT}_t)$ then the edges in T_s—whose total cost is approximately OPT_s—are already very cheap. Thus, replacing these edges by cheaper ones would only waste rearrangements. To avoid technical difficulties we use $\mathrm{OPT}_t^{\max} := \max\{\mathrm{OPT}_s : 1 \leq s \leq t\}$ instead of OPT_t to determine whether $\mathrm{OPT}_s \in O(\varepsilon \mathrm{OPT}_t)$; note that since we assume the triangle inequality for the costs of the edges, $\mathrm{OPT}_t \leq \mathrm{OPT}_t^{\max} \leq 2\mathrm{OPT}_t$ holds.

With this in mind, we define $\ell(t)$ as the largest iteration with ignorable edges with respect to $\mathrm{OPT}_t^{\max}$, i. e. , $\ell(t) \leq t-1$ is the largest non-negative integer such that $\mathrm{OPT}_{\ell(t)}^{\max} \leq \varepsilon \mathrm{OPT}_t^{\max}$.

For our first freezing rule we consider sequences of edges $(g_s^0, \ldots, g_s^{i(s)})$, where g_s^0 corresponds to the greedy edge added at iteration s (that is, an edge connecting v_s to one of its closest neighbors in V_{s-1}). At the moment when edge g_s^0 is removed from our solution we define g_s^1 as the element that replaces g_s^0. In general g_s^i is the edge that was swapped in for edge g_s^{i-1}. In this way, the only edge in the sequence that belongs to the current solution is $g_s^{i(s)}$. Note that $i(s)$ changes through the iterations. Notationally, $i(s)$ will refer to the value at the iteration in consideration in the current context (unless it is stated otherwise).

With this construction, we *freeze* a sequence $(g_s^0, \ldots, g_s^{i(s)})$ in iteration t if $s \leq \ell(t)$. Note that since $\ell(\cdot)$ is non-decreasing, once the sequence is frozen $g_s^{i(s)}$ will stay indefinitely in the solution. Our second freezing rule is somewhat simpler. We skip swaps that remove edges that are too small, namely, smaller than $\varepsilon \mathrm{OPT}_t^{\max}/(t-\ell(t))$. Combining these ideas we propose the following algorithm.

Algorithm Sequence-Freeze

Define $T_0 = \emptyset$. For each iteration $t \geq 1$ do as follows.

1. Let g_t^0 be any minimum cost edge in $\{v_t v_s : 0 \leq s \leq t-1\}$.
2. Initialize $T_t := T_{t-1} \cup \{g_t^0\}$ and $i(t) := 0$.
3. While there exists a pair of edges $(f, h) \in (E_t \setminus T_t) \times T_t$ such that $(T_t \cup \{f\}) \setminus h$ is a tree, and the following three conditions are satisfied
   (C1) $c(h) > (1+\varepsilon) \cdot c(f)$,
   (C2) $h = g_s^{i(s)}$ for some $s \geq \ell(t)+1$, and
   (C3) $c(h) > \varepsilon \frac{\mathrm{OPT}_t^{\max}}{t-\ell(t)}$,
   then set $T_t := (T_t \cup \{f\}) \setminus \{h\}$, $i(s) := i(s)+1$ and $g_s^{i(s)} := f$.
4. Return T_t.

Conditions (C2) and (C3) correspond to the two freezing rules described above. In the following we show that this algorithm is $(1+\varepsilon)$-competitive and uses amortized budget $O\left(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon}\right)$.

Competitive Analysis. To prove that our algorithm is $(1+O(\varepsilon))$-competitive, we first show that Conditions (C1) and (C3) imply a cost increase of at most a factor $(1+3\epsilon)$. Then we show that skipping swaps because of Condition (C2) can increase the cost of the solution by at most $O(\varepsilon \mathrm{OPT}_t)$.

Consider an iteration t and let $\ell := \ell(t)$. We partition the tree T_t into two disjoint subsets, $T_t = T_t^{\text{old}} \cup T_t^{\text{new}}$ where $T_t^{\text{old}} := \{g_1^{i(1)}, \ldots, g_\ell^{i(\ell)}\}$ and $T_t^{\text{new}} := \{g_{\ell+1}^{i(\ell+1)}, \ldots, g_t^{i(t)}\}$.

Lemma 1. *For each iteration t it holds that $c(T_t^{\text{new}}) \leq (1+3\,\varepsilon)\mathrm{OPT}_t$.*

For bounding the cost of T_t^{old}, we use induction over the iterations. The inductive step is given in the following lemma.

Lemma 2. *Let $\varepsilon < \frac{1}{7}$. Consider an iteration t and suppose that $c(T_{\ell(t)}) \leq (1+7\varepsilon)\mathrm{OPT}_{\ell(t)}$. Then it holds that $c(T_t^{\text{old}}) \leq 4\,\varepsilon\,\mathrm{OPT}_t$.*

The above reasoning implies the following lemma.

Lemma 3. *Algorithm* Sequence-Freeze *is $(1+7\varepsilon)$-competitive for any $\varepsilon < \frac{1}{7}$.*

Amortized Budget Bound. To show the constant amortized budget bound, we define $k_q := |T_q \setminus T_{q-1}|$ and prove that for every $t \geq 1$ it holds that $\sum_{q=1}^{t} k_q \leq D_\varepsilon \cdot t$, where $D_\varepsilon \in O\left(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon}\right)$.

Lemma 4. *Assume that $\sum_{q=\ell(t)+1}^{t} k_q \leq C_\varepsilon \cdot (t - \ell(\ell(t)+1))$ for every $t \geq 1$ with $C_\varepsilon \in O\left(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon}\right)$. Then for every $t \geq 1$ it holds that $\sum_{q=1}^{t} k_q \leq 2C_\varepsilon \cdot t$.*

It remains to prove that the assumption of Lemma 4 holds. The two freezing rules, Conditions (C2) and (C3), are crucial for this purpose. Indeed, we will bound the length of the sequences $(g_s^0, \ldots, g_s^{i(s)})$, which will give a direct bound on $\sum_{q=\ell(t)+1}^{t} k_q \leq C_\varepsilon \cdot (t - \ell(\ell(t)+1))$. This can be done since by Condition (C1), we only swap edges when the cost is decreased by a $(1+\varepsilon)$ factor, that is, $c(g_s^j) \leq c(g_s^{j-1})/(1+\varepsilon)$ for each j. Thus, the length of this sequence is bounded by $\log_{1+\varepsilon} c(g_s^0) - \log_{1+\varepsilon} c(g_s^{i(s)-1}) + 1$. We can bound this quantity further by lower bounding the cost $g_s^{i(s)-1}$ with our freezing rules and by exploiting a particular cost structure of greedy edges g_s^0.

More precisely, consider the values $i(s)$ at the end of iteration t, and let $i'(s)$ be the value of $i(s)$ at the beginning of iteration $\ell(t)+1$ (and $i'(s) := 0$ for $s \geq \ell(t)+1$). By Condition (C2) in the algorithm, in iterations $\ell(t)+1$ to t we only touch edges belonging to $\{g_s^{i'(s)}, g_s^{i'(s)+1}, \ldots, g_s^{i(s)}\}$ for some $s \in \{\ell(\ell(t)+1)+1, \ldots, t\}$. Let us denote $r := \ell(\ell(t)+1)$. Then,

$$\sum_{q=\ell(t)+1}^{t} k_q \leq \sum_{s=r+1}^{t} (i(s) - i'(s) + 1) = 2(t-r) + \sum_{s=r+1}^{t} (i(s) - 1 - i'(s)). \quad (1)$$

We now upper bound each term $i(s) - 1 - i'(s)$ for $s \in \{r+1, \ldots, t\}$, which corresponds to the length of the sequence $(g_s^{i'(s)+1}, g_s^{i'(s)+2}, \ldots, g_s^{i(s)-1})$.

Lemma 5. *For each $s \in \{r, r+1 \ldots, t\}$ it holds that*

$$i(s) - 1 - i'(s) \leq \frac{1}{\ln(1+\varepsilon)} \cdot \left(\ln c(g_s^0) - \ln c(g_s^{i(s)-1}) \right). \tag{2}$$

Proof. The lemma follows due to Condition (C1), since whenever we add an edge g_s^j and remove g_s^{j-1}, then $c(g_s^j) < c(g_s^{j-1})/(1+\varepsilon)$. □

In the next claims, we lower bound $c(g_s^{i(s)-1})$ and upper bound $\sum_{s=r+1}^{t} \ln c(g_s^0)$. These bounds applied to Equation (2) will lead with (1) to the desired bound on $\sum_{q=\ell(t)+1}^{t} k_q$.

Proposition 1 *Due to Condition (C3), it holds that either $c(g_s^{i(s)-1}) \geq \varepsilon^2 \frac{\mathrm{OPT}_t^{\max}}{(t-r)}$ or $i(s) - 1 - i'(s) \leq 0$.*

Recall that for any s, g_s^0 is a closest connection between v_s and any element in $\{v_0, \ldots, v_{s-1}\}$. Such greedy edges are known to have a special cost structure [1].

Lemma 6 (Alon and Azar [1]). *Let $e_1, \ldots, e_t$ be greedy edges reindexed such that $c(e_1) \geq c(e_2) \geq \ldots \geq c(e_t)$. Then, $c(e_j) \leq \frac{2\mathrm{OPT}_t}{j}$ for all $j \in \{1, \ldots, t\}$.*

Lemma 7. $\sum_{s=r+1}^{t} \ln c(g_s^0) \leq (t-r) \cdot (\ln(2 \cdot \mathrm{OPT}_t^{\max}) - \ln(t-r) + 1)$.

Proof. We rename edges $\{g_{r+1}^0, \ldots, g_t^0\} = \{e_1, \ldots, e_{t-r}\}$ such that $c(e_1) \geq \ldots \geq c(e_{t-r})$. Lemma 6 implies that $c(e_j) \leq 2\frac{\mathrm{OPT}_t}{j} \leq 2\frac{\mathrm{OPT}_t}{j}^{\max}$ for all j. We conclude

$$\sum_{s=r+1}^{t} \ln c(g_s^0) = \sum_{j=1}^{t-r} \ln c(e_j) \leq (t-r) \ln(2 \cdot \mathrm{OPT}_t^{\max}) - \sum_{j=1}^{t-r} \ln j.$$

The lemma follows since for any $n \in \mathbb{N}_{>0}$ it holds that $\sum_{j=1}^{n} \ln j \geq n \ln(n) - n$. □

The above statement and basic arithmetics imply the desired bound.

Lemma 8. *For each $t \geq 1$ it holds that $\sum_{q=1}^{t} k_q \leq D_\varepsilon \cdot t$, where $D_\varepsilon \in O\left(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon}\right)$.*

Our main result follows from Lemmas 3 and 8.

Theorem 2. *There exists a $(1+\varepsilon)$-competitive algorithm for the online MST problem with amortized recourse budget $O\left(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon}\right)$.*

Finally, we show that the amortized budget of our algorithm is best possible up to logarithmic factors. This result also implies that 1-competitive solutions need non-constant amortized budget.

Theorem 3. *Any $(1+\varepsilon)$-competitive algorithm for the online MST problem requires an amortized recourse budget of $\Omega(\frac{1}{\varepsilon})$.*

4 The Non-Amortized Scenario

For the amortized setting we have seen that with sufficient (but constant) budget we can obtain a competitive ratio of $1+\epsilon$. In the non-amortized setting however, there can be no $(2-\epsilon)$-competitive algorithm with constant budget. Nevertheless, it is a long standing open question [7] whether in the non-amortized setting one can obtain constant-competitive online algorithms with constant budget. We contribute two pieces of evidence for an affirmative answer to this question: we state a conjecture about a structural property of optimal solutions and show that a natural greedy algorithm with budget 2 is constant-competitive if the conjecture holds true. Furthermore, we show that in the full information scenario even one extra recourse action every $O(1)$ iterations is enough to obtain a constant-competitive algorithm.

4.1 A Greedy Algorithm with Budget 2

A natural online algorithm with budget 2 works as follows: In each iteration t find the shortest connection g_t from v_t to V_{t-1} and find a pair of edges f_t, h_t maximizing $c(h_t)-c(f_t)$ among all edges such that $T_t := (T_{t-1} \cup \{g_t, f_t\}) \setminus \{h_t\}$ is a spanning tree for V_t. Then T_t is the tree for iteration t. Imase and Waxman [7] suggested to require that $c(f_t) \leq c(h_t)/2$ and f_t is adjacent to v_t. If no such edges f_t, h_t exist they define $T_t := T_{t-1} \cup \{g_t\}$. With this modification, we prove a relation between the competitive factor of this algorithm and a structural property of optimal solutions that we conjecture to be true.

To state our conjecture we need the following definition: given a complete graph $G = (V, E)$ and a non-negative cost function c on the edges, we say that the graph is 2-*metric* if for every cycle $C \subseteq E$ it holds that $c(e) \leq 2 \cdot c(C \setminus \{e\})$ for all $e \in C$. Moreover, for a given real number x we define $x_+ := \max\{x, 0\}$ and $x_- := \max\{-x, 0\}$. Note that $x = x_+ - x_-$. Also, denote $\Delta\mathrm{OPT}_t := \mathrm{OPT}_t - \mathrm{OPT}_{t-1}$.

Conjecture 4 *There exists a constant $\alpha \geq 1$ satisfying the following. Consider any input sequence $G_0, G_1, \ldots, G_n$ of the online MST problem with recourse, with a cost function c' on the edges such that G_t is 2-metric for all $t \geq 0$. If OPT_t denotes the optimal cost of the tree in iteration t for cost function c', then for all $t \geq 1$ it holds that $\sum_{s=1}^{t} (\Delta\mathrm{OPT}_s)_- \leq \alpha \cdot \mathrm{OPT}_t$.*

We do not know how to show that the conjecture holds. However, it is easy to see that it holds if OPT_t, as a function of t, is *unimodal*, i. e., there exists a t^* such that OPT_t is non-decreasing for $t < t^*$, and non-increasing for $t \geq t^*$.

Theorem 5. *If Conjecture 4 holds then the greedy algorithm with budget 2 is $(2 \cdot (\alpha + 1))$-competitive.*

To show the theorem let $I \subseteq \mathbb{N}_0$ be the subset of iterations t such that $T_t = T_{t-1} \cup \{g_t\}$. In particular, for all iterations $t \notin I$ we have that $c(T_t) \leq c(T_{t-1})$. Setting $\Delta T_t := c(T_t) - c(T_{t-1})$ for each t, we can decompose the cost of each tree

T_t by $c(T_t) = \sum_{s=1}^{t} \Delta T_t$, and by our previous observation $c(T_t) \leq \sum_{s \in I, s \leq t} \Delta T_s$. It remains to bound the latter value. To this end, we use the fact that in the iterations in I the algorithm did not find any pair of edges f_t, h_t to swap such that $c(f_t) \leq c(h_t)/2$ with f_t being adjacent to v_t. As we will see this implies the same for the optimal solution. In particular, if we double the cost of all edges of the form $v_t v_s$ with $t \in I, s \leq t-1$ and $v_t v_s \neq g_t$, then the optimal solution would not see any gain in inserting any other edge but g_t in this iteration. More precisely, consider the following set $E_I := \{v_t v_s : t \in I, s \leq t-1\} \setminus \{g_t : t \in I\}$. We define a new cost function c' by setting $c'(e) := 2 \cdot c(e)$ for all $e \in E_I$ and $c'(e) := c(e)$ for all $e \notin E_I$.

Let us denote by OPT'_t the cost of an MST in iteration t with cost function c' and note that $\mathrm{OPT}'_t \leq 2 \cdot \mathrm{OPT}_t$. Also, since c is metric, graph G_t with cost function c' is 2-metric. The next lemma implies that with the new cost function $\mathrm{OPT}'_t - \mathrm{OPT}'_{t-1} = c'(g_t) = c(g_t) = c(T_t) - c(T_{t-1}) = \Delta T_t$.

Lemma 9. *For all t there exists a minimum spanning tree T_t^{**} for graph $G_t = (V_t, E_t)$ with cost function c' such that $T_t^{**} \cap E_I = \emptyset$. In particular $T_t^{**} = T_{t-1}^{**} \cup \{g_t\}$ if $t \in I$, where g_t is a shortest connection between v_t and any vertex in V_{t-1}.*

We can show Theorem 5 using the bound $c(T_t) \leq \sum_{s \in I, s \leq t} \Delta T_s$, the last lemma, our conjecture, and a short computation.

4.2 The Full Information Scenario

To support the conjecture that there is a constant-competitive online algorithm with budget 2, we show that there always exists a sequence of spanning trees with constant competitive ratio, even when allowing one unit of budget in each iteration and one additional edge swap exactly every k iterations. More formally, we say that a sequence of trees $T_0, T_1, ...$ needs budget $1+1/k$ when: $|T_t \setminus T_{t-1}| \leq 2$ if $t \equiv 0 \bmod k$, and $|T_t \setminus T_{t-1}| = 1$ if $t \not\equiv 0 \bmod k$.

First, we present a constant-competitive algorithm with budget 2. Given the cost function c and graphs $G_0, \ldots, G_n$ in advance, we compute a 2-approximate Hamiltonian path X_n (which is a tree) for graph G_n by computing an MST and using the folkloric shortcutting technique[5]. Taking shortcuts of this path to vertices in G_t we obtain a Hamiltonian path X_t for G_t with cost at most $2\mathrm{OPT}_n$. It is not hard to see that the budget used by the sequence $X_1, \ldots, X_n$ is at most 2. However, it is not necessarily constant-competitive. By carefully embedding this idea into the classic *doubling framework* [5] we obtain the following result.

Theorem 6. *Under full information there exists a 14-competitive algorithm with budget 2.*

Finally, skipping most of the edge swaps done by the above sequence still yields constant-competitive solutions. The intuitive idea is that we group the

[5] The *shortcutting* technique takes a tree, obtains a Eulerian graph by doubling the edges, finds a Eulerian walk, and visits the first copy of each node in the walk [10].

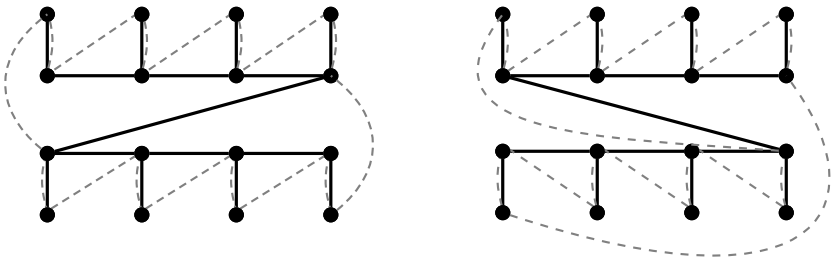

Fig. 1: Two trees (solid) differing in one edge such that standard shortcutting of a(ny) walk in the double-tree yields arbitrarily different tours (dotted).

edges into sets $\{g_s^0, ..., g_s^{i(s)}\}$ where g_s^0 is the greedy edge added in iteration s and each edge g_s^i is at some point swapped out for the edge g_s^{i+1} with $c(g_s^i) \leq 2 \cdot c(g_s^{i+1})$. Then, we can construct a new sequence with budget $1 + \frac{1}{k}$ which performs at least one out of $O(k)$ swaps of each sequence.

Theorem 7. *Let $k \in \mathbb{N}$. In the full information scenario, there exists a $2^{O(k)}$-competitive algorithm with budget $1 + \frac{1}{k}$.*

5 Applications to TSP

In this section we consider the online TSP with recourse. In a natural approach we aim for combining our algorithms for the online MST problem with the classic *shortcutting* technique [10], which yields a sequence of tours, each with cost at most twice the cost of the tree. This implies a $(2 + \varepsilon)$-competitive algorithm. However, bounding the budget is intricate. Obviously, MSTs that differ in few edges might have quite different Eulerian walks and thus TSP tours. However, even when adapting the Eulerian walks as much as possible, the standard shortcutting might lead to very different TSP tours. In fact, there are examples (see Figure 1) in which two trees T, T' differ in just one edge and shortcutting *any* Eulerian walk on (doubled) T' in the standard way yields a tour Q' which differs from Q for T in an unbounded number of edges.

Our key ingredients for solving this problem are as follows: In case of an edge swap, we decompose the Eulerian walk W corresponding to the tour Q before the swap into 4 sub-walks defined by the swap edges, which we concatenate then in an appropriate way. Furthermore, we find a *robust* variant of the shortcutting technique: instead of shortcutting the new Eulerian walk by visiting the *first* appearance of a node, we remember the copy of each node that we visit in W to construct Q, and then we visit the same copy when constructing Q'. In general we cannot expect to obtain the same tour, but we prove that Q and Q' differ in at most 4 edges which may be necessary when concatenating the sub-walks.

Lemma 10. *Given an online algorithm computing a sequence of trees $T_0, T_1, \ldots$, there exists an online algorithm that computes tours $Q_2, Q_3, \ldots$ such that $c(Q_t) \leq 2 \cdot c(T_t)$ and $|Q_t \setminus Q_{t-1}| \leq 4 \cdot |T_t \setminus T_{t-1}|$ for any $t \geq 1$.*

This lemma and Theorems 2 and 5 yield directly the following results.

Theorem 8. *For online TSP with recourse, (i) there exists a $(2+\varepsilon)$-competitive algorithm with amortized budget $O(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$ for any $\varepsilon > 0$, and (ii) if Conjecture 4 holds for some α, there exists a $(4(\alpha+1))$-competitive algorithm with budget 8.*

References

1. N. Alon and Y. Azar. On-line Steiner trees in the euclidean plane. *Discrete Comp. Geom.*, 10:113–121, 1993.
2. V. Bafna, B. Kalyanasundaram, and K. Pruhs. Not all insertion methods yield constant approximate tours in the euclidean plane. *Theor. Comput. Sci.*, 125:345–360, 1994.
3. J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming.* Springer Series in Operations Research. Springer, New York, 1997.
4. N. Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Report 388, Graduate School of Industrial Administration, CMU, 1976.
5. M. Chrobak and C. Kenyon-Mathieu. Competitiveness via doubling. *SIGACT NEWS*, 37:115–126, 2006.
6. M. Dynia, M. Korzeniowski, and J. Kutylowski. Competitive maintenance of minimum spanning trees in dynamic graphs. In *SOFSEM 2007*, pages 260–271, 2007.
7. M. Imase and B. M. Waxman. Dynamic Steiner tree problem. *SIAM J. Discrete Math.*, 4:369–384, 1991.
8. P. Jaillet and M. R. Wagner. Online vehicle routing problems: A survey. In *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces*, pages 221–237. Springer, 2008.
9. B. Kalyanasundaram and K. Pruhs. Constructing competitive tours from local information. *Theor. Comput. Sci.*, 130:125–138, 1994.
10. E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. *The Traveling Salesman Problem.* John Wiley and Sons, Chichester, 1985.
11. N. Megow, K. Mehlhorn, and P. Schweitzer. Online graph exploration: New results on old and new algorithms. In *ICALP 2011*, pages 478–489, 2011.
12. C. A. S. Oliveira and P. M. Pardalos. A survey of combinatorial optimization problems in multicast routing. *Comput. & Oper. Res.*, 32:1953–1981, 2005.
13. J.-J. Pansiot and D. Grad. On routes and multicast trees in the internet. *ACM SIGCOMM Comp. Comm. Review*, 28:41–50, 1998.
14. N. Subramanian and S. Liu. Centralized multi-point routing in wide area networks. In *SAC 1991*, pages 46–52, 1991.
15. J. Verschae. *The Power of Recourse in Online Optimization.* PhD thesis, Technische Universität Berlin, Germany, 2012.
16. B. M. Waxman. Routing of multipoint connections. *IEEE J. Sel. Area Comm.*, 6:1617–1622, 1988.