Maintaining Perfect Matchings at Low Cost

Jannik Matuschke
Research Center for Operations Management, KU Leuven, Leuven, Belgium
jannik.matuschke@kuleuven.be

Ulrike Schmidt-Kraepelin
Institute of Software Engineering and Theoretical Computer Science, TU Berlin, Berlin, Germany
u.schmidt-kraepelin@tu-berlin.de

José Verschae
Institute of Engineering Sciences, Universidad de O'Higgins, Rancagua, Chile
jose.verschae@uoh.cl

Abstract

The min-cost matching problem suffers from being very sensitive to small changes of the input. Even in a simple setting, e.g., when the costs come from the metric on the line, adding two nodes to the input might change the optimal solution completely. On the other hand, one expects that small changes in the input should incur only small changes on the constructed solutions, measured as the number of modified edges. We introduce a two-stage model where we study the trade-off between quality and robustness of solutions. In the first stage we are given a set of nodes in a metric space and we must compute a perfect matching. In the second stage $2k$ new nodes appear and we must adapt the solution to a perfect matching for the new instance.

We say that an algorithm is (α, β)-robust if the solutions constructed in both stages are α-approximate with respect to min-cost perfect matchings, and if the number of edges deleted from the first stage matching is at most βk. Hence, α measures the quality of the algorithm and β its robustness. In this setting we aim to balance both measures by deriving algorithms for constant α and β. We show that there exists an algorithm that is $(3, 1)$-robust for any metric if one knows the number $2k$ of arriving nodes in advance. For the case that k is unknown the situation is significantly more involved. We study this setting under the metric on the line and devise a $(10, 2)$-robust algorithm that constructs a solution with a recursive structure that carefully balances cost and redundancy.

2012 ACM Subject Classification Mathematics of computing → Combinatorial algorithms

Keywords and phrases matchings, robust optimization, approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.82

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1811.10580`.

Funding The authors were supported by Project Fondecyt Nr. 1181527, the Alexander von Humboldt Foundation with funds of the German Federal Ministry of Education and Research (BMBF), BAYLAT, and the Deutsche Forschungsgemeinschaft under grant BR 4744/2-1.

Acknowledgements We thank anonymous reviewers for their helpful feedback.

1 Introduction

Weighted matching is one of the founding problems in combinatorial optimization, playing an important role in the settling of the area. The work by Edmonds [8] on this problem greatly influenced the role of polyhedral theory on algorithm design [23]. On the other hand, the problem found applications in several domains [1, 2, 6, 22, 26]. In particular

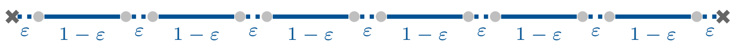

Figure 1 Example instance on the line. Vertices in Stage 1 are depicted by light grey dots and second stage arrivals are indicated as dark grey crosses. First and second state optimum are depicted by solid and dotted edges, respectively. The arrival of two new vertices leads to a significant drop in the cost of the optimum.

routing problems are an important area of application, and its procedures often appeared as subroutines of other important algorithms, the most notable being Christofides' algorithm [5] for the *traveling salesperson problem (TSP).*

An important aspect of devising solution methods for optimization problems is studying the sensitivity of the solution towards small changes in the input. This sensitivity analysis has a long history and plays an important role in practice [10]. Min-cost matching is a problem that has particularly sensitive optimal solutions. Assume for example that nodes lie on the real line at points ℓ and $\ell + 1 - \varepsilon$ for some $0 < \varepsilon < 1$ and all $\ell \in \{1, \dots, n\}$, see Figure 1. The min-cost matching, for costs equal the distance on the line, is simply the edges $\{\ell, \ell + 1 - \varepsilon\}$. However, even under a minor modification of the input, e.g., if two new nodes appear at points $1 - \varepsilon$ and $n + 1$, the optimal solution changes all of its edges, and furthermore the cost decreases by a $\Theta(1/\varepsilon)$ factor. Rearranging many edges in an existing solution is often undesirable and may incur large costs, for example in an application context where the matching edges imply physical connections or binding commitments between nodes. A natural question in this context is whether we can avoid such a large number of rearrangements by constructing a *robust* solution that is only slightly more expensive. In other words, we are interested in studying the trade-off between robustness and the cost of solutions.

We consider a two-stage robust model with recourse. Assume we are given an underlying metric space $(\mathcal{X}, c)$. The input for the *first stage* is a complete graph G_1 whose node set $V(G_1)$ is a finite, even subset of $\mathcal{X}$. The cost of an edge $\{v, w\}$ is given by the corresponding cost $c(v, w)$ in the metric space[1]. In a second stage we get an extended complete graph G_2 containing all nodes in $V(G_1)$ plus $2k$ additional nodes. As before, costs of edges in G_2 are given by the underlying metric. In the first stage we must create a perfect matching M_1 for G_1. In the second stage, after G_2 is revealed, we must adapt our solution by constructing a new perfect matching M_2 for G_2, called the *second stage reply.* We say that a solution M_1 is two-stage (α, β)-robust if for any instantiation of the second stage there exists a solution M_2 such that two conditions hold. First, the total cost of edges in M_i must satisfy $c(M_i) \leq \alpha \cdot c(O_i)$ for $i \in \{1, 2\}$, where O_i denotes a min-cost perfect matching in G_i. Second, it must hold that $|M_1 \setminus M_2| \leq \beta k$. An algorithm is two-stage (α, β)-robust if, given G_1 and c, it returns a two-stage (α, β)-robust matching and, given the set of new arrivals, a corresponding second stage reply. We refer to α as the *competitive factor* and β as the *recourse factor* of the algorithm. Our main goal is to balance cost and recourse, and thus we aim to obtain algorithms where α and β are constants.

Our model is closely related to an online model with recourse. Consider a graph whose nodes are revealed online two by two. Our objective is to maintain a perfect matching at all times. As above, irrevocable decisions do not allow for constant competitive factors. This suggests a model where in each iteration we are allowed to modify a constant number of edges.

[1] Graphs with arbitrary cost functions do not allow for $(O(1), O(1))$-robust matchings in general, e.g., consider a variant of the example in Figure 1 in which all omitted edges have infinite cost.

An online algorithm that maintains an α-approximation at all time while deleting at most β edges per iteration can be easily transformed into a two-stage (α, β)-robust algorithm. Given an instance of the two-stage model, we choose an arbitrary order for the nodes available in the first stage and create M_1 by following the updates proposed by an online algorithm. Then, we repeat the procedure for the arrivals in Stage 2. Thus, our two-stage model is also the first necessary step for understanding this more involved online model. Megow et al. [20] study a similar online model for minimum spanning trees and the TSP in metric graphs. After giving a $(1+\varepsilon)$-competitive algorithm with recourse factor $\frac{1}{\varepsilon}\log(\frac{1}{\varepsilon})$ for the former problem, they are able to derive a $(2+\varepsilon)$-competitive algorithm with constant recourse factor for the latter problem by combining their results with a modified version of the well-known double-tree algorithm. An algorithm for the online variant of our proposed model together with the aforementioned results, would give rise to an online variant of Christofide's algorithm which can yield an improved competitiveness factor for the considered online TSP.

Our Results and Techniques. We distinguish two variants of the model. In the *k-known* case we assume that in Stage 1 we already know the number of new nodes $2k$ that will arrive in Stage 2. For this case we present a simple two-stage $(3, 1)$-robust algorithm.

▶ **Theorem 1.** *Let $(\mathcal{X}, c)$ be a metric space, $V_1 \subseteq \mathcal{X}$ with $|V_1|$ even, and G_1 be the complete graph on V_1. For $k \in \mathbb{N}$ known in advance, there is a perfect matching M_1 in G_1 that is two-stage $(3, 1)$-robust for $2k$ arrivals. Such a matching and corresponding second stage reply can be computed in time poly$(|V_1|, k)$.*

The example in Figure 1 illustrates a worst case scenario for the strategy of choosing O_1 as the first stage matching for $k = 1$. The reason for this is that the nodes arriving in Stage 2 induce a path in $O_1 \Delta O_2$ that incurs a significant drop in the optimal cost. Our algorithm is designed towards preparing for such bad scenarios. To this end, we define the notion of *gain* for a path P with respect to a matching X as follows:

$$\text{gain}_X(P) := c(P \cap X) - c(P \setminus X).$$

In Stage 1, our algorithm chooses k edge-disjoint O_1-alternating paths of maximum total gain with respect to O_1. For each such path P we modify O_1 by removing $P \cap O_1$ and adding $(P \setminus O_1) \cup \{e(P)\}$, where $e(P)$ is the edge that connects the endpoints of P. Our choice of paths of maximum gain implies that $P \cap O_1$ is larger than $P \setminus O_1$. Therefore we can bound the cost of the solution in the first stage against that of O_1 and also infer that most of its costs is concentrated on the edges $e(P)$. For the second stage we construct a solution for the new instance by removing the k edges of the form $e(P)$ and adding new edges on top of the remaining solution. The algorithm is described in detail in Section 2.

For the case where k is unknown the situation is considerably more involved as a first stage solution must work for any number of arriving nodes simultaneously. In this setting we restrict our study to the real line and give an algorithm that is two-stage $(10, 2)$-robust.

▶ **Theorem 2.** *Let $\mathcal{X} = \mathbb{R}$ and $c = |\cdot|$, $V_1 \subseteq \mathcal{X}$ with $|V_1|$ even, and let G_1 be the complete graph on V_1. Then there is a perfect matching M_1 in G_1 that is two-stage $(10, 2)$-robust. Such a matching, as well as the second stage reply, can be computed in time poly$(|V_1|, k)$.*

The first stage solution M is constructed iteratively, starting from the optimal solution. We will choose a path P greedily such that it maximizes $\text{gain}_M(P)$ among all alternating paths that are *heavy*, i.e., the cost of $P \cap M$ is a factor 2 more expensive than the cost of $P \setminus M$. Then M is modified by augmenting along P and adding edge $e(P)$, which we fix to be in the final solution. We iterate until M only consists of fixed edges. As we are on the line, each path P corresponds to an interval and we can show that the constructed solution form a laminar family. Furthermore, our choice of heavy paths implies that their lengths satisfy an exponential decay property. This allows us to bound cost of the first stage solution. For the second stage, we observe that the symmetric difference $O_1 \Delta O_2$ induces a set of intervals on the line. For each such an interval, we remove on average at most two edges from the first stage matching and repair the solution with an optimal matching for the exposed vertices. A careful choice of the removed edges, together with the greedy construction of the first stage solution, enables us to bound the cost of the resulting second stage solution within a constant factor of the optimum. See Sections 3 and 4 for a detailed description of this case.

Related Work. Intense research has been done on several variants of the *online bipartite matching* problem [17, 16, 18, 4, 21]. In this setting we are given a known set of *servers* while a set of *clients* arrive online. In the *online bipartite metric matching problem* servers and clients correspond to points from a metric space. Upon arrival, each client must be matched to a server irrevocably, at cost equal to their distance. For general metric spaces, there is a tight bound of $(2n-1)$ on the competitiveness factor of deterministic online algorithms, where n is the number of servers [18, 16]. Recently, Raghvendra presented a deterministic algorithm [24] with the same competitiveness factor, that in addition is $O(\log(n))$-competitive in the random arrival model. Also, its analysis can be parameterized for any metric space depending on the length of a TSP tour and its diameter [21]. For the special case of the metric on the line, Raghvendra [25] recently refined the analysis of the competitive ratio to $O(\log(n))$. This gives a deterministic algorithm that matches the previously best known bound by Gupta and Lewi [15], which was attained by a randomized algorithm. As the lower bound of 9.001 [9] could not be improved for 20 years, the question whether there exists a constant competitive algorithm for the line remains open.

The *online matching with recourse problem* considers an unweighted bipartite graph. Upon arrival, a client has to be matched to a server and can be reallocated later. The task is to minimize the number of reallocations under the condition that a maximum matching is always maintained. The problem was introduced by Grove, Kao and Krishnan [11]. Chaudhuri et al. [4] showed that for the random arrival model a simple greedy algorithm uses $O(n\log(n))$ reallocations with high probability and proved that this analysis is tight. Recently, Bernstein, Holm and Rotenberg [3] showed that the greedy algorithm needs $O(n\log^2 n)$ allocations in the adversarial model, leaving a small gap to the lower bound of $O(n\log n)$. Gupta, Kumar and Stein [14] consider a related problem where servers can be matched to more than one client, aiming to minimize the maximum number of clients that are assigned to a server. They achieve a constant competitive factor server while doing in total $O(n)$ reassignments.

Online min-cost problems with reassignments have been studied in other contexts. For example in the *online Steiner tree problem with recourse* a set of points on a metric space arrive online. We must maintain Steiner trees of low cost by performing at most a constant (amortized) number of edge changes per iteration. While the pure online setting with no reassignment only allows for $\Omega(\log(n))$ competitive factors, just one edge deletion per iteration is enough to obtain a constant competitive algorithm [12]; see also [13, 20].

The concept of *recoverable robustness* is also related to our setting [19]. In this context the perfect matching problem on unweighted graphs was considered by Dourado et. al. [7]. They seek to find perfect matchings which, after the failure of some edges, can be recovered to a perfect matching by making only a small number of modifications. They establish computational hardness results for the question whether a given graph admits a robust recoverable perfect matching.

2 Known Number of Arrivals

In this section, we consider the setting where k (number of arrival pairs in Stage 2) is already known in Stage 1. Let G_1 be the complete graph given in Stage 1 (with edge costs c induced by an arbitrary metric) and let O_1 be a min-cost perfect matching in G_1. Without loss of generality assume that $|O_1| > 2k$, as otherwise, we can remove all edges of M_1 in Stage 2.

Algorithm 1.1 works as follows:

(i) Let $P_1, \dots, P_k$ be edge-disjoint, O_1-alternating paths maximizing $\sum_{i=1}^{k} \mathrm{gain}_{O_1}(P_i)$.
(ii) Set $\overline{M} := O_1 \Delta P_1 \Delta \dots \Delta P_k$.
(iii) Return $M_1 := \overline{M} \cup \{e(P_i) : i \in [k]\}$.

It is easy to see that each path P_i starts and ends with an edge from O_1 and $\mathrm{gain}_{O_1}(P_i) \geq 0$. As a consequence, M_1 is a perfect matching and

$$c(\overline{M}) = c(O_1) - \textstyle\sum_{i=1}^{k} \mathrm{gain}_{O_1}(P_i) \leq c(O_1).$$

Using $c(e(P_i)) \leq c(P_i)$ and $\bigcup_{i=1}^{k} P_i = O_1 \Delta \overline{M} \subseteq O_1 \cup \overline{M}$ we obtain

$$c(M_1) \leq c(\overline{M}) + \textstyle\sum_{i=1}^{k} c(P_i) \leq c(\overline{M}) + c(\overline{M}) + c(O_1) \leq 3 \cdot c(O_1).$$

Now consider the arrival of $2k$ new vertices, resulting in the graph G_2 with min-cost matching O_2. Note that $O_2 \Delta \overline{M}$ is a U-join, where U is the union of the endpoints of the paths $P_1, \dots, P_k$ and the $2k$ newly arrived vertices.

Algorithm 1.2 works as follows:

(i) Let $P'_1, \dots, P'_{2k}$ be the $2k$ maximal paths from $O_2 \Delta \overline{M}$.
(ii) Return $M_2 := \overline{M} \cup \{e(P'_i) : i \in [2k]\}$.

Note that $O_1 \Delta O_2$ consists of k alternating paths $R_1, \dots, R_k$, from which we remove the starting and ending O_2-edge. Then these paths would have been a feasible choice for $P_1, \dots, P_k$, implying that the total gain of the R_i's is at most that of the P_i's. We conclude that

$$c(\overline{M}) = c(O_1) - \textstyle\sum_{i=1}^{k} \mathrm{gain}_{O_1}(P_i) \leq c(O_1) - \textstyle\sum_{i=1}^{k} \mathrm{gain}_{O_1}(R_i) \leq c(O_2).$$

Applying $\bigcup_{i=1}^{2k} P'_i \subseteq O_2 \Delta \overline{M} \subseteq O_2 \cup \overline{M}$, we obtain

$$c(M_2) \leq c(\overline{M}) + \textstyle\sum_{i=1}^{2k} c(P'_i) \leq c(\overline{M}) + c(\overline{M}) + c(O_2) \leq 3 \cdot c(O_2).$$

As $|M_1 \setminus M_2| \leq |M_1 \setminus \overline{M}| = k$, we conclude that M_1 is indeed two-stage $(3, 1)$-robust.

We remark that $\overline{M}$ is always a min-cost matching of cardinality $|V(G_1)| - 2k$ in G_1. Thus, alternatively to Algorithm 1.1, we can compute a min-cost matching of cardinality $|V(G_1)| - 2k$ and match uncovered nodes at minimum cost. Finding $\overline{M}$ directly as well as finding gain-maximizing paths as described in Algorithm 1.1 can be done efficiently by solving a min-cost T-join problem in an extension of G_1. This concludes our proof of Theorem 1.

3 Unknown Number of Arrivals – Stage 1

In this section, we consider the case that the underlying metric corresponds to the real line. This implies that there is a Hamiltonian path L in G_1 such that $c(v, w) = c(L[v, w])$ for all $v, w \in V(G_1)$, where $L[v, w]$ is the subpath of L between nodes v and w. We will refer to L as *the line* and call the subpaths of L *intervals*. The restriction to the metric on the line results in a uniquely defined min-cost perfect matching O_1 with a special structure. All proofs omitted due to space constraints can be found in the appendix of the full version.

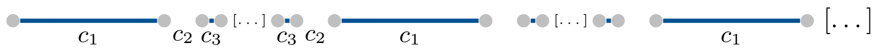

Figure 2 For fixed $\alpha, \beta \in O(1)$, we construct an instance for which Algorithm 1.1 is not (α, β)-robust for any $k \in O(1)$. G_1 is constructed such that O_1 contains $\beta + 1$ edges of size c_1 and $\beta^3 + \beta^2$ of size c_3 that are equally distributed between c_1-edges. The distance between any two consecutive edges in O_1 is c_2. Values c_1, c_2, c_3 are chosen depending on α, guaranteeing $c_1 \gg \beta^3 c_2 \gg \beta^6 c_3$. Algorithm 1.1 with $k \geq \beta + 1$ chooses $M_1 = O_1$. Now, assume there are two arrivals at the extremes of the line: while the optimal costs in the second stage decrease heavily to $c(O_2) \in \Theta(\beta^3 c_2)$, only β deletions are allowed within M_1. As a result, there does not exist a feasible second stage reply. Now, consider Algorithm 1.1 with $k \leq \beta$. Then, M_1 contains more than $\beta^2 + \beta$ edges of size c_2. If in the second stage $2(\beta + 1)$ nodes arrive right next to the endpoints of c_1-edges, the optimal costs drop to $\Theta(\beta^3 c_3)$ while only $\beta^2 + \beta$ deletions are allowed. Again, no feasible second stage reply exists.

▶ **Lemma 3.** *O_1 is the unique perfect matching contained in L.*

When the number of arrivals is not known in the first stage, the approach for constructing the first stage matching introduced in Section 2 does not suffice anymore. Figure 2 illustrates a class of instances for which Algorithm 1.1 cannot achieve $(O(1), O(1))$-robustness, no matter how we choose k. For a matching M, define $g(M) := \max_{e \in L} |\{\{v, w\} \in M : e \in L[v, w]\}|$. Informally speaking, $g(M)$ captures the maximal number of times a part of the line is traversed by edges in M. The example in Figure 2 can be generalized to show that we cannot restrict ourselves to constructing matchings M_1 such that $g(M_1)$ is bounded by a constant.

In view of the example in Figure 2, we adapt the approach from Section 2 as follows. Instead of creating a fixed number of paths, our algorithm now iteratively and greedily selects a path P of maximum gain with respect to a dynamically changing matching X (initially $X = O_1$). In order to bound the total cost incurred by adding edges of the form $e(P)$, we only consider paths P for which $X \cap P$ contributes a significant part to the total cost of P.

▶ **Definition 4.** *Let $X, P \subseteq E(G_1)$.*
1. *We say that P is X-heavy if $c(P \cap X) \geq 2 \cdot c(P \setminus X)$.*
2. *We say that P is X-light if $c(P \cap X) \leq \frac{1}{2} \cdot c(P \setminus X)$.*

Algorithm 2.1 works as follows:

```
(i)   Set $M_1 := \emptyset$ and $X := O_1$.
(ii)  While $X \neq \emptyset$: Let $P$ be an $X$-heavy $X$-alternating path maximizing $\text{gain}_X(P)$ and
      update $M_1 \leftarrow M_1 \cup \{e(P)\}$ and $X \leftarrow X \Delta P$.
(iii) Return $M_1$.
```

Note that in each iteration, the path P starts and ends with an edge from X as it is gain-maximizing (if P ended with an edge that is not in X, we could simply remove that edge and obtain a path of higher gain). Therefore it is easy to see that $X \cup M_1$ is always a perfect matching, and in each iteration the cardinality of X decreases by 1.

Now number the iterations of the while loop in Algorithm 2.1 from 1 to n. Let $X^{(i)}$ be the state of X at the beginning of iteration i. Let $P^{(i)}$ be the path chosen in iteration i and let $e^{(i)} = e(P^{(i)})$ be the corresponding edge added to M. The central result in this section is Lemma 7, in which we show that the paths $P^{(i)}$ form a laminar family of intervals on the line.

Within the proof we will make use of observations stated in Lemmas 5 and 6. For convenience, we define the projection $\psi(e) := L[v, w]$ that maps an edge $e = \{v, w\} \in E(G_1)$ to the corresponding subpath $L[v, w]$.

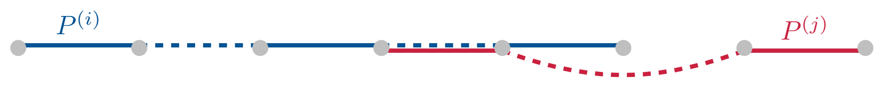

Figure 3 A minimal example of a situation in which Lemma 7 would be violated. There are two iterations $i < j$ with paths $P^{(i)}$ (depicted in blue) and $P^{(j)}$ (depicted in red) such that $X \cap P^{(j)}$ was not modified between iteration i and iteration j. Then, extending the blue path $P^{(i)}$ with the rightmost edge yields an $X^{(i)}$-heavy path with higher gain than $P^{(i)}$, a contradiction.

▶ **Lemma 5.** *Let $X \subseteq E(G_1)$.*

1. *Let $A, B \subseteq E(G_1)$ be two X-heavy (X-light, respectively) sets with $A \cap B = \emptyset$. Then $A \cup B$ is X-heavy (X-light, respectively).*
2. *Let $A, B \subseteq E(G_1)$ with $B \subseteq A$. If A is X-heavy and $\text{gain}_X(B) < 0$, then $A \setminus B$ is X-heavy. If A is X-light and $\text{gain}_X(B) > 0$, then $A \setminus B$ is X-light.*

▶ **Lemma 6.** *Let $X \subseteq L$ be a matching.*

1. *Let P be an X-heavy X-alternating path maximizing $\text{gain}_X(P)$. If X covers all vertices in $V(\psi(P))$, then $P = \psi(P)$.*
2. *Let $I \subseteq L$ be an interval. Then there is an X-alternating path P such that $c(P \cap X) = c(X \cap I)$ and $c(P \setminus X) = c(I \setminus X)$.*

▶ **Lemma 7.**

1. *$X^{(i)}, P^{(i)} \subseteq L$ for all $i \in [n]$.*
2. *For all $i, j \in [n]$ with $i < j$, either $P^{(i)} \cap P^{(j)} = \emptyset$ or $P^{(j)} \subset P^{(i)}$.*

Proof. We say a pair (i, j) with $i < j$ is *violating* if $\psi(P^{(i)}) \cap \psi(P^{(j)}) \neq \emptyset$ and $\psi(P^{(j)}) \setminus \psi(P^{(i)}) \neq \emptyset$. We will show that no violating pair exists. This proves the lemma as the following claim asserts.

▷ Claim. If $P^{(j)} \neq \psi(P^{(j)})$, then there is a violating pair (i', j') with $i' < j' \leq j$.

Proof. Let j' be minimal with $P^{(j')} \neq \psi(P^{(j')})$. Note that minimality of j' implies that $X^{(j')} = O_1 \Delta P^{(1)} \Delta \dots \Delta P^{(j'-1)} \subseteq L$. Then Lemma 6 implies that there must be a vertex $v \in V(\psi(P^{(j')}))$ not covered by $X^{(j')}$. Because $X^{(j')}$ covers exactly those vertices not covered by $\{e^{(i')} : i' < j'\}$, there must be an $i' < j'$ such that v is an endpoint of $P^{(i')}$. The vertex v cannot be an endpoint of $P^{(j')}$, because v is exposed in $X^{(j')}$ and $P^{(j')}$ starts and ends with edges from $X^{(j')}$. This implies that (i', j') is a violating pair. ◁

Now let us assume there are no violating pairs. Then $P^{(i)} = \psi(P^{(i)}) \subseteq L$ for all $i \in [n]$ by the claim, which also implies $X^{(i)} \subseteq L$. This implies the lemma as, in this situation, the condition for violating pairs coincides with the condition in point 2 of the lemma.

By contradiction assume there is a violating pair. Choose j such that j is minimal among all possible choices of violating pairs. Then choose i such that it is maximal for that j among all violating pairs.

Note that the claim implies that $P^{(i)}, X^{(i)}, X^{(j)} \subseteq L$. Furthermore, our choice of i and j implies that $\psi(P^{(j')}) \cap \psi(P^{(j)}) = \emptyset$ for all j' with $i < j' < j$, as otherwise (i, j') or (j', j) would be a violating pair. In particular, $P^{(j')} \cap P^{(j)} = \emptyset$ for all j' with $i < j' < j$ and thus

$$X^{(j)} \cap P^{(j)} = X^{(i+1)} \cap P^{(j)} = (X^{(i)} \Delta P^{(i)}) \cap P^{(j)}. \tag{1}$$

Now consider $I_1 := \psi(P^{(j)}) \cap P^{(i)}$ and $I_2 := \psi(P^{(j)}) \setminus P^{(i)}$, both of which are non-empty since (i, j) is a violating pair. Then (1) implies $X^{(j)} \cap I_1 = I_1 \setminus X^{(i)}$ and $I_1 \setminus X^{(j)} = X^{(i)} \cap I_1$. We conclude that $\text{gain}_{X^{(j)}}(I_1) = -\text{gain}_{X^{(i)}}(I_1) \leq 0$, as I_1 is a prefix of the gain-maximizing path $P^{(i)}$ (see the appendix of the full version for a formal proof).

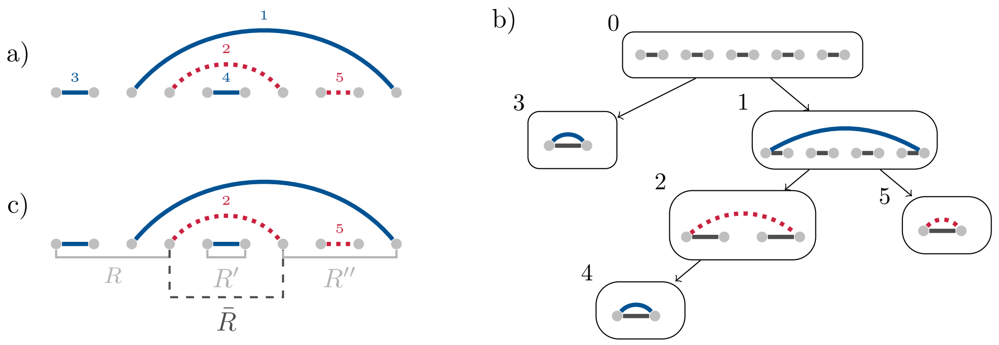

Figure 4 a) Illustration of the matching created by an example execution of Algorithm 2.1. Edges added to M_1 in an iteration from W_{H} are depicted by blue solid lines and edges created in an iteration from W_{L} are illustrated by red dotted lines. b) Illustration of the corresponding tree. For every tree-node $i \in W$, grey edges indicate $X^{(i)}$ and an arc illustrates the edge connecting the end nodes of $P^{(i)}$. c) Illustration of example assignment of requests to iterations (defined in Section 4). Requests $R, R', R'' \in \mathcal{R}$ are assigned such that $R, R'' \in \mathcal{R}(0)$ and $R' \in \mathcal{R}(2)$. $\bar{R} \in \bar{\mathcal{R}}(0)$ is a gap between two requests associated with tree-node 0.

Therefore $I_2 = \psi(P^{(j)}) \setminus I_1$ is $X^{(j)}$-heavy by Lemma 5. But then I_2 is also $X^{(i)}$-heavy because (1) implies $X^{(j)} \cap I_2 = X^{(i)} \cap I_2$. Hence $I' := P^{(i)} \cup I_2$ is $X^{(i)}$-heavy by Lemma 5 and further

$$\mathrm{gain}_{X^{(i)}}(I') = \mathrm{gain}_{X^{(i)}}(P^{(i)}) + \mathrm{gain}_{X^{(j)}}(I_2) > \mathrm{gain}_{X^{(i)}}(P^{(i)}),$$

because $\mathrm{gain}_{X^{(j)}}(I_2) \geq \frac{1}{3}c(I_2)$. By Lemma 6 there is an $X^{(i)}$-heavy, $X^{(i)}$-alternating path with higher gain than $P^{(i)}$, a contradiction. ◀

Tree structure. Lemma 7 induces a tree structure on the paths selected by Algorithm 2.1. We define the directed tree $T = (W, A)$ as follows. We let $W := \{0, \ldots n\}$ and define $P^{(0)} := L$. For $i, j \in W$ we add the arc (i, j) to A if $P^{(j)} \subset P^{(i)}$ and there is no $i' \in W$ with $P^{(j)} \subset P^{(i')} \subset P^{(i)}$. It is easy to see that T is an out-tree with root 0. We let $T[i]$ be the unique 0-i-path in T. We define the set of children of $i \in W$ by $\mathrm{ch}(i) := \{j \in W : (i,j) \in A\}$. Furthermore, let $W_{\mathrm{H}} := \{i \in W : |T[i]| \text{ is odd}\}$ and $W_{\mathrm{L}} := \{i \in W : |T[i]| \text{ is even}\}$ be the set of *heavy* and *light* nodes in the tree, respectively. These names are justified by the following lemma. See Figure 4 a)-b) for an illustration.

▶ **Lemma 8.** *If $i \in W_H$, then $P^{(i)} \cap X^{(i)} = P^{(i)} \cap O_1$ and, in particular, $P^{(i)}$ is O_1-heavy. If $i \in W_L \setminus \{0\}$, then $P^{(i)} \cap X^{(i)} = P^{(i)} \setminus O_1$ and, in particular, $P^{(i)}$ is O_1-light.*

Proof. Let $i \in W \setminus \{0\}$. From Lemma 7 we know that for every iteration $i' \in W \setminus \{0\}$ with $i' < i$ it holds that either $P^{(i)} \cap P^{(i')} = \emptyset$ or $P^{(i)} \subset P^{(i')}$. In the first case it holds that $P^{(i)} \cap X^{(i'+1)} = P^{(i)} \cap X^{(i')}$, in the latter case it holds that $P^{(i)} \cap X^{(i'+1)} = P^{(i)} \cap (P^{(i)} \Delta X^{(i')})$. Moreover, it is easy to see that $i' < i$ and $P^{(i)} \subset P^{(i')}$ holds if and only if $i' \in V(T[i]) \setminus \{0, i\}$. If $i \in W_{\mathrm{H}}$, this implies that there exist an even number of iterations $i' < i$ for which $P^{(i)} \subset P^{(i')}$ holds. Hence, we obtain

$$P^{(i)} \cap X^{(i)} = P^{(i)} \cap (\underbrace{P^{(i)} \Delta \ldots \Delta P^{(i)}}_{\text{evenly often}} \Delta\, O_1) = P^{(i)} \cap O_1.$$

If $i \in W_{\mathrm{L}}$, this implies that there exist an odd number of iterations $i' < i$ for which $P^{(i)} \subset P^{(i')}$ holds. Hence, we can deduce that

$$P^{(i)} \cap X^{(i)} = P^{(i)} \cap (\underbrace{P^{(i)} \Delta \ldots \Delta P^{(i)}}_{\text{oddly often}} \Delta\, O_1) = P^{(i)} \setminus O_1.$$ ◀

The fact that nested paths are alternatingly O_1-heavy and O_1-light implies an exponential decay property. As a consequence we can bound the cost of M_1.

▶ **Lemma 9.** *Let $i \in W \setminus \{0\}$. Then $\sum_{j \in \mathrm{ch}(i)} c(P^{(j)}) \leq \frac{1}{2} \cdot c(P^{(i)})$.*

Proof. Let $i \in W \setminus \{0\}$. Then

$$\textstyle\sum_{j \in \mathrm{ch}(i)} c(P^{(j)}) \leq \frac{3}{2} \sum_{j \in \mathrm{ch}(i)} c(P^{(j)} \cap X^{(j)}) \leq \frac{3}{2} c(P^{(i)} \setminus X^{(i)}) \leq \frac{1}{2} c(P^{(i)}),$$

where the first inequality follows from the fact that $P^{(j)}$ is $X^{(j)}$-heavy; the second inequality follows from the fact that $P^{(j)} \cap X^{(j)} \subsetneq P^{(i)} \setminus X^{(i)}$ for $j \in \mathrm{ch}(i)$ and the fact that the intervals $P^{(j)}$ for all children are disjoint; the last inequality follows from the fact that $P^{(i)}$ is $X^{(i)}$-heavy. ◀

▶ **Lemma 10.** *$c(M_1) \leq 3c(O_1)$.*

Proof. Note that $c(M_1) = \sum_{i=1}^{n} c(e^{(i)}) = \sum_{i \in W \setminus \{0\}} c(P^{(i)})$. For $\ell \in \mathbb{N}$, let

$$W_\ell := \{i \in W : |T[i]| = \ell\}.$$

Observe that Lemma 9 implies that $\sum_{i \in W_\ell} c(P^{(i)}) \leq \left(\frac{1}{2}\right)^{\ell-1} \sum_{i \in W_1} c(P^{(i)})$ for all $\ell \in \mathbb{N}$. Furthermore $\sum_{i \in W_1} c(P^{(i)}) \leq \frac{3}{2} c(O_1)$, because $W_1 \subseteq W_{\mathrm{H}}$. Hence

$$c(M_1) = \sum_{\ell=1}^{\infty} \sum_{i \in W_\ell} c(P^{(i)}) \leq \sum_{\ell=1}^{\infty} \left(\frac{1}{2}\right)^{\ell-1} \sum_{i \in W_1} c(P^{(i)}) = 2 \cdot \frac{3}{2} c(O_1).$$ ◀

4 Unknown Number of Arrivals – Stage 2

We now discuss how to react to the arrival of $2k$ additional vertices. We let O_2 be the min-cost perfect matching in the resulting graph G_2 and define

$$\mathcal{R} := \{P : P \text{ is a maximal path in } (O_1 \Delta O_2) \cap L\}.$$

We call the elements of $\mathcal{R}$ *requests*. An important consequence of our restriction to the metric space on the line is that $|\mathcal{R}| \leq k$ (in fact, each of the k maximal paths of $O_1 \Delta O_2$ is contained in L after removing its first and last edge).

▶ **Lemma 11.** *$|\mathcal{R}| \leq k$ and each $R \in \mathcal{R}$ starts and ends with an edge of O_1.*

For simplification of the analysis we make the following assumptions on the structure of the request set.

▶ **Assumption A.** *For all $i \in W_L$ and all $R \in \mathcal{R}$, either $P^{(i)} \cap R = \emptyset$ or $P^{(i)} \subseteq R$, or $R \subseteq P^{(i)}$.*

▶ **Assumption B.** *For all $j \in W_H$, if $\bigcup_{R \in \mathcal{R}} R \cap P^{(j)} \neq \emptyset$, then the first and last edge of $P^{(j)}$ are in $\bigcup_{R \in \mathcal{R}} R$.*

In the appendix of the full version we prove formally that these are without loss of generality. For intuition, we give a short sketch of the proof. Assume we are confronted with a set of requests $\mathcal{R}$ that violates at least one of the assumptions. Based on $\mathcal{R}$, we can construct a modified set of requests $\mathcal{R}'$ complying with the assumptions and fulfilling two properties: First, $\mathrm{gain}_{O_1}(\bigcup_{R \in \mathcal{R}'} R) > \mathrm{gain}_{O_1}(\bigcup_{R \in \mathcal{R}} R)$, i.e., R' induces smaller second stage optimal costs than $\mathcal{R}$ and secondly, $|\mathcal{R}'| \leq |\mathcal{R}|$, i.e., $\mathcal{R}'$ allows for at most as many modifications as $\mathcal{R}$ does. As a consequence, if we run our proposed second stage algorithm for $\mathcal{R}'$ and construct M_2 accordingly, the analysis of the approximation and recourse factor carry over from the analysis of $\mathcal{R}'$ to the actual set of requests $\mathcal{R}$. Hence, we can assume w.l.o.g. that $\mathcal{R}$ fulfills the assumptions.

From the set of requests $\mathcal{R}$, we will determine a subset of at most $2k$ edges that we delete from M_1. To this end, we assign each request to a light node in W_{L} as follows. For $R \in \mathcal{R}$ we define $i_R := \max\{i \in W_{\mathrm{L}} : R \subseteq P^{(i)}\}$, i.e., $P^{(i_R)}$ is the inclusionwise minimal interval of a light node containing R. For $i \in W_{\mathrm{L}}$, let

$$\mathcal{R}(i) := \big\{R \in \mathcal{R} : i_R = i\big\}.$$

Furthermore, we also keep track of the gaps between the requests in $\mathcal{R}(i)$ as follows. For $i \in W_{\mathrm{L}}$, let

$$\begin{aligned}\bar{\mathcal{R}}(i) := \big\{\bar{R} \subseteq P^{(i)} : \;& \bar{R} \text{ is a maximal path in } P^{(i)} \setminus \textstyle\bigcup_{R \in \mathcal{R}(i)} R \text{ and} \\ & \bar{R} \subseteq P^{(j)} \text{ for some } j \in \mathrm{ch}(i)\big\}.\end{aligned}$$

Note that $R' \cap R'' = \emptyset$ for all $R', R'' \in \mathcal{R}(i) \cup \bar{\mathcal{R}}(i), i \in W_{\mathrm{L}}$. However, $\bar{R} \in \bar{\mathcal{R}}(i)$ may contain a request $R \in \mathcal{R}(j)$ from descendants j of i. See Figure 4 c) for an illustration of the assignment.

For $i \in W_{\mathrm{L}}$, let $W_{\mathrm{H}}(i) := \mathrm{ch}(i)$ and $W_{\mathrm{L}}(i) := \{i' \in W : i' \in \mathrm{ch}(j) \text{ for some } j \in W_{\mathrm{H}}(i)\}$. Note that $W_{\mathrm{H}}(i) \subseteq W_{\mathrm{H}}$ and $W_{\mathrm{L}}(i) \subseteq W_{\mathrm{L}}$. Before we can state the algorithm for computing the second stage reply, we need one final lemma.

▶ **Lemma 12.** *Let $i \in W_L$. For every $R \in \mathcal{R}(i)$, there is a $j \in W_H(i)$ with $P^{(j)} \cap R \neq \emptyset$. For every $\bar{R} \in \bar{\mathcal{R}}(i)$, there is an $i' \in W_L(i)$ with $P^{(i')} \cap R \neq \emptyset$.*

We are now ready to state the algorithm. We first describe and discuss a simplified version, which yields an approximation guarantee of 19. At the end of the paper, we discuss how to slightly adapt the algorithm so as to obtain the factor of 10 given in Theorem 2.

Algorithm 2.2 works as follows:

- **(i)** Create the matching M' by removing the following edges from M_1 for each $i \in W_{\mathrm{L}}$:
  1. The edge $e^{(i)}$ if $i \neq 0$ and $\mathcal{R}(i) \neq \emptyset$.
  2. For each $R \in \mathcal{R}(i)$ the edge $e^{(j^*_R)}$ where $j^*_R := \min\{j \in W_{\mathrm{H}}(i) : P^{(j)} \cap R \neq \emptyset\}$.
  3. For each $\bar{R} \in \bar{\mathcal{R}}(i)$ the edge $e^{(i^*_{\bar{R}})}$ where $i^*_{\bar{R}} := \min\{i' \in W_{\mathrm{L}}(i) : P^{(i')} \cap R \neq \emptyset\}$.
- **(ii)** Let M'' be a min-cost matching on all vertices not covered by M' in G_2.
- **(iii)** Return $M_2 := M' \cup M''$.

Let Z be indices of the edges removed in step (i). It is not hard to see that $|\bar{\mathcal{R}}(i)| \leq |\mathcal{R}(i)| - 1$ for each $i \in W_{\mathrm{L}}$ and therefore $|Z| \leq 2k$, bounding the recourse of Algorithm 2.2 as intended.

▶ **Lemma 13.** $|Z| \leq 2k$.

Now let $Y := W \setminus (Z \cup \{0\})$ be the nodes corresponding to edges that have not been removed and

$$\bar{Y} := \{i \in Y : T[i] \setminus \{0, i\} \subseteq Z\}$$

the nodes that correspond to maximal intervals that have not been removed.

The following lemma is a consequence of the exponential decay property. It shows that in order to establish a bound on the cost of M_2, it is enough to bound the cost of all paths $P^{(i)}$ for $i \in \bar{Y}$.

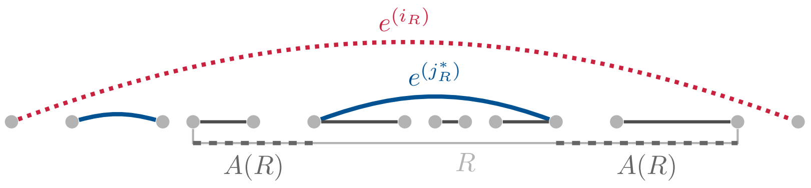

Figure 5 Illustration of the proof of Lemma 15. The solid dark grey lines depict edges in $O_1 \cap R$. At the time when Algorithm 2.1 constructed $P^{(j^*_R)}$, no other child interval of i_R intersecting with R was present. Thus $X^{(j^*_R)} \cap R = X^{(i_R+1)} \cap R = O_1 \cap R$. If $A(R)$ was O_1-heavy, then $P^{(j^*_R)} \cup A(R)$ would be an O_1-heavy path of higher O_1-gain than $P^{(j^*_R)}$, contradicting the greedy construction.

▶ **Lemma 14.** $c(M_2) \leq c(O_2) + 3\sum_{i \in \bar{Y}} c(P^{(i)})$.

It remains to bound the cost of the paths associated with the tree nodes in $\bar{Y}$. We establish a charging scheme by partitioning the line into three areas A, B, C:

1. For $R \in \mathcal{R}$, let $A(R) := R \setminus P^{(j^*_R)}$. We define $A := \bigcup_{R \in \mathcal{R}} A(R)$.
2. For $i \in W_{\mathrm{L}}$ and $\bar{R} \in \bar{\mathcal{R}}(i)$, let $B(\bar{R}) := \bar{R} \setminus \bigcup_{i' \in W_{\mathrm{L}}(i) \cap Z} P^{(i')}$.
   We define $B := \bigcup_{\bar{R} \in \bar{\mathcal{R}}} B(\bar{R})$.
3. We define $C := L \setminus (A \cup B)$.

Consider a set $A(R)$ for some $R \in \mathcal{R}$. Recall that i_R is the index of the smallest light interval constructed by Algorithm 2.1 containing R and that j^*_R is the first child interval of i_R created by Algorithm 2.1 that intersects R. From the choice of j^*_R and the greedy construction of $P^{(j^*_R)}$ as a path of maximum $X^{(j^*_R)}$-gain we can conclude that $A(R)$ is not O_1-heavy; see Figure 5 for an illustration. Therefore $c(A(R) \setminus O_1) > \frac{1}{3}c(A(R))$. Note that $O_2 \cap A(R) = A(R) \setminus O_1$, because $A(R) \subseteq R$. Hence we obtain the following lemma.

▶ **Lemma 15.** *Let $R \in \mathcal{R}$. Then $\frac{1}{3}c(A(R)) \leq c(O_2 \cap A(R))$.*

A similar argument implies the same bound for all sets of the type $B(\bar{R})$ for some $\bar{R} \in \bar{\mathcal{R}}(i)$ and $i \in W_{\mathrm{L}}$.

▶ **Lemma 16.** *Let $\bar{R} \in \bar{\mathcal{R}}$. Then $\frac{1}{3}c(B(\bar{R})) \leq c(O_2 \cap B(\bar{R}))$.*

Furthermore, one can show that the sets of the form $A(R)$ and $B(\bar{R})$ and the set C form a partition of L (see the appendix of the full version). We define $x : L \to \mathbb{R}_+$ by

$$x(e) := \begin{cases} \frac{1}{3} & \text{if } e \in A \cup B \\ 1 & \text{if } e \in C \cap O_2 \\ 0 & \text{if } e \in C \setminus O_2 \end{cases}$$

and obtain the following lemma as a consequence of Lemmas 15 and 16.

▶ **Lemma 17.** $\sum_{e \in L} c(e)x(e) \leq c(O_2 \cap L)$.

We are now able to bound the cost of each path $P^{(i)}$ for $i \in \bar{Y}$ against its local budget $\sum_{e \in P^{(i)}} c(e)x(e)$. We consider the cases where $i \in \bar{Y}$ corresponds to a heavy and light edge in Lemma 18 and Lemma 19, respectively.

▶ **Lemma 18.** *Let $j \in \bar{Y} \cap W_H$. Then $c(P^{(j)}) \leq 6\sum_{e \in P^{(j)}} c(e)x(e)$.*

Proof. We first show that each request intersecting with $P^{(j)}$ is either contained in a child interval of j or the edges in $P^{(j)}$ intersecting with the request are covered by A.

▷ Claim. Let $R \in \mathcal{R}$ with $R \cap P^{(j)} \neq \emptyset$. Then $R \cap P^{(j)} \subseteq A(R)$ or $R \subseteq P^{(i')}$ for some $i' \in \mathrm{ch}(j)$.

Proof. Assume $R \not\subseteq P^{(i')}$ for any $i' \in \mathrm{ch}(j)$. In particular, this implies that $i_R \notin \mathrm{desc}(j)$. Let i be the parent node of j in T. We first exclude the possibility that i_R is an ancestor of i. Indeed, if this was the case then $R \not\subseteq P^{(i)}$ and thus by Assumption A, $P^{(i)} \subseteq R$. But then $P^{(i)}$ can neither contain other request intervals, nor can it intersect any non-request intervals. Therefore $\mathcal{R}(i) = \emptyset$ and $i \neq i^*_{\bar{R}}$ for all $\bar{R} \in \bigcup_{i' \in W_{\mathrm{L}}} \bar{\mathcal{R}}(i')$. Therefore, $i \notin Z$, a contradiction to $j \in \bar{Y}$. This implies that $i_R = i$. Further note that $j \in \bar{Y}$ implies $j \neq j^*_R$ for all $R \in \mathcal{R}$, as otherwise, j would have been tagged for removal. We conclude that $j^*_R \in \mathrm{ch}(i) \setminus \{j\}$ and thus $R \cap P^{(j)} \subseteq R \setminus P^{(j^*_R)} = A(R)$, as $P^{(j)}$ and $P^{(j^*_R)}$ are disjoint. ◁

Let $Q := \bigcup_{i' \in \mathrm{ch}(j)} P^{(i')}$. Consider $e \in P^{(j)} \cap O_1$. Note that the claim implies that if $e \notin A \cup Q$, then $e \in O_2$. We conclude that $(P^{(j)} \cap O_1) \setminus Q \subseteq A \cup B \cup (C \cap O_2)$. Further, a variant of Lemma 9 implies that $c(P^{(j)} \cap O_1) \leq \frac{4}{3} c((P^{(j)} \cap O_1) \setminus Q)$ (see appendix of the full version for details). We obtain

$$c(P^{(j)}) \leq \tfrac{3}{2} c(P^{(j)} \cap O_1) \leq 2c((P^{(j)} \cap O_1) \setminus Q) \leq 6 \textstyle\sum_{e \in P^{(j)}} c(e)x(e),$$

where the first inequality follow from the fact that $P^{(j)}$ is O_1-heavy and the last inequality follows from the fact that $x(e) \geq 1/3$ for every $e \in A \cup B \cup (C \cap O_2)$. This proves the lemma. ◀

▶ **Lemma 19.** *Let $i \in \bar{Y} \cap W_L$. Then $c(P^{(i)}) \leq 3 \sum_{e \in P^{(i)}} c(e)x(e)$.*

The proof of Lemma 19 is given in the appendix of the full version. We state a short proof sketch for intuition. Let $i \in \bar{Y} \cap W_{\mathrm{L}}$. Due to the removal of light edges with associated requests and Assumption A, we know that $P^{(i)}$ either does not intersect with any request or $P^{(i)}$ is completely covered by a request. In the former case one can show that $P^{(i)}$ is included in B and hence the lemma holds. In the latter case, $P^{(i)}$ is either included in A and hence the lemma holds or it is included in C in which case the lemma holds since $P^{(i)}$ is O_1-light.

As a consequence, we can now show a first constant factor bound. Because the paths $P^{(i)}$ for $i \in \bar{Y}$ are pairwise disjoint, Lemmas 17 to 19 imply $\sum_{i \in \bar{Y}} c(P^{(i)}) \leq 6c(O_2)$. Plugging this into Lemma 14 we obtain

$$c(M_2) \leq c(O_2) + 3 \textstyle\sum_{i \in \bar{Y}} c(P^{(i)}) \leq 19c(O_2).$$

Improvement of approximation factor. We can improve the approximation factor from 19 to 10 by a slight modification of the set of edges removed in Stage 2. To this end, note that the factor for the bound given in Lemma 18 is greater than the one given in Lemma 19. Indeed, the only reason for the weaker bound is that edges in $\bar{Y} \cap W_{\mathrm{H}}$ can have descendants with associated requests. Excluding this case improves the factor within the bound of the lemma from 6 to 3. We formalize this in the following lemma.

▶ **Lemma 20.** *Let $j \in \bar{Y} \cap W_H$ such that $\mathrm{ch}(j) \cap Z = \emptyset$. Then $c(P^{(j)}) \leq 3 \sum_{e \in P^{(j)}} c(e)x(e)$.*

We now modify Algorithm 2.2 as follows: Compute the set Z of edges tagged for removal by Algorithm 2.2. Now construct the set Z' by defining

$$\bar{H} := \{j \in W_{\mathrm{H}} \setminus Z \,:\, \mathrm{ch}(j) \cap Z \neq \emptyset\} \qquad \text{and} \qquad Z' := (Z \cup \bar{H}) \setminus \textstyle\bigcup_{j \in \bar{H}} \mathrm{ch}(j).$$

Now execute step 2 of Algorithm 2.2 with Z' instead of Z, i.e., remove the edges with indices in Z' and connect the unmatched vertices by a min-cost matching. It is easy to see that $|Z'| \leq |Z|$ and therefore the recourse factor is still bounded by 2. For analyzing the approximation factor, we define $Y' := W \setminus (Z' \cup \{0\})$ the nodes corresponding to edges that have not been removed and $\bar{Y}' := \{i \in Y : T[i] \setminus \{0, i\} \subseteq Z'\}$ in analogy to the original analysis. It is easy to see that $\mathrm{ch}(j) \cap Z = \emptyset$ for every $j \in \bar{Y}' \cap W_{\mathrm{H}} = \bar{Y} \setminus \bar{H}$ and hence $c(P^{(j)}) \leq 3 \sum_{e \in P^{(j)}} c(e)x(e)$ by Lemma 20. Furthermore, if $i \in \bar{Y}' \cap W_{\mathrm{L}}$ then either $i \in \bar{Y}$ and $c(P^{(i)}) \leq 3 \sum_{e \in P^{(i)}} c(e)x(e)$ by Lemma 19, or $i \in \mathrm{ch}(j)$ for some $j \in \bar{Y} \setminus \bar{Y}' = \bar{H}$. Note that Lemmas 9 and 18 imply

$$\sum_{j \in \bar{H}} \sum_{i \in \mathrm{ch}(j)} c(P^{(i)}) \leq \tfrac{1}{2} \sum_{j \in \bar{H}} c(P^{(j)}) \leq 3 \sum_{j \in \bar{H}} \sum_{e \in P^{(j)}} c(e)x(e).$$

We thus obtain $\sum_{i \in \bar{Y}'} c(P^{(i)}) \leq 3c(O_2)$ and hence for the modified algorithm it holds that

$$c(M_2) \leq c(O_2) + 3 \sum_{i \in \bar{Y}'} c(P^{(i)}) \leq 10c(O_2).$$

References

1 Michael Ball, Lawrence Bodin, and Robert Dial. A Matching Based Heuristic for Scheduling Mass Transit Crews and Vehicles. *Transportation Science*, 17:4–31, 1983.

2 Colin E. Bell. Weighted matching with vertex weights: An application to scheduling training sessions in NASA space shuttle cockpit simulators. *European Journal of Operational Research*, 73:443–449, 1994.

3 Aaron Bernstein, Jacob Holm, and Eva Rotenberg. Online Bipartite Matching with Amortized $O(log^2 n)$ Replacements. In *Proceedings of the Twenty-ninth Annual ACM SIAM Symposium on Discrete Algorithms (SODA '18)*, pages 947–959, 2018.

4 Kamalika Chaudhuri, Constantinos Daskalakis, Robert D. Kleinberg, and Henry Lin. Online Bipartite Perfect Matching With Augmentations. In *Proceedings of the Twenty-eight IEEE Conference on Computer Communications (INFOCOM '09)*, pages 1044–1052, 2009.

5 Nicos Christofides. Worst-Case Analysis of a New Heuristic for the Traveling Salesman Problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie Mellon University, 1976.

6 U. Derigs and A. Metz. A matching-based approach for solving a delivery/pick-up vehicle routing problem with time constraints. *Operations-Research-Spektrum*, 14:91–106, 1992.

7 Mitre Costa Dourado, Dirk Meierling, Lucia D. Penso, Dieter Rautenbach, Fabio Protti, and Aline Ribeiro de Almeida. Robust recoverable perfect matchings. *Networks*, 66:210–213, 2015.

8 Jack Edmonds. Maximum matching and a polyhedron with 0,1-vertices. *Journal of Research of the National Bureau of Standards, Section B*, 69:125–130, 1965.

9 Bernhard Fuchs, Winfried Hochstättler, and Walter Kern. Online Matching on a Line. *Theoretical Computer Science*, 332:251–264, 2005.

10 A. M. Geoffrion and R. Nauss. Parametric and Postoptimality Analysis in Integer Linear Programming. *Management Science*, 23:453–466, 1977.

11 Edward F. Grove, Ming-Yang Kao, P. Krishnan, and Jeffrey Scott Vitter. Online perfect matching and mobile computing. In *Algorithms and Data Structures (WADS '95)*, volume 955 of *Lecture Notes in Computer Science*, pages 194–205. Springer, 1995.

12 A. Gu, A. Gupta, and A. Kumar. The power of deferral: maintaining a constant-competitive Steiner tree online. In *Proceedings of the Forty-fifth Annual ACM Symposium on Symposium on Theory of Computing (STOC '13)*, pages 525–534, 2013.

13 A. Gupta and A. Kumar. Online Steiner Tree with Deletions. In *Proceedings of the Twenty-fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '14)*, pages 455–467, 2014.

14 Anupam Gupta, Amit Kumar, and Cliff Stein. Maintaining Assignments Online: Matching, Scheduling, and Flows. In *Proceedings of the Twenty-fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '14)*, pages 468–479, 2014.

15 Anupam Gupta and Kevin Lewi. The Online Metric Matching Problem for Doubling Metrics. In *Proceedings of the Thirty-ninth International Colloquium on Automata, Languages and Programming (ICALP '12)*, pages 424–435, 2012.

16 Bala Kalyanasundaram and Kirk Pruhs. Online Weighted Matching. *Journal of Algorithms*, 14:478–488, 1993.

17 Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. An Optimal Algorithm for On-line Bipartite Matching. In *Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing (STOC '90)*, pages 352–358, 1990.

18 Samir Khuller, Stephen G. Mitchell, and Vijay V. Vazirani. On-line Algorithms for Weighted Bipartite Matching and Stable Marriages. *Theoretical Computer Science*, 127:255–267, 1994.

19 Christian Liebchen, Marco Lübbecke, Rolf Möhring, and Sebastian Stiller. *The Concept of Recoverable Robustness, Linear Programming Recovery, and Railway Applications*, pages 1–27. Springer Berlin Heidelberg, 2009.

20 Nicole Megow, Martin Skutella, José Verschae, and Andreas Wiese. The power of recourse for online MST and TSP. *SIAM Journal on Computing*, 45:859–880, 2016.

21 K. Nayyar and S. Raghvendra. An Input Sensitive Online Algorithm for the Metric Bipartite Matching Problem. In *Proceedings of the Fifty-eight Annual IEEE Symposium on Foundations of Computer Science (FOCS '17)*, pages 505–515, 2017.

22 Snjólfur Ólafsson. Weighted Matching in Chess Tournaments. *The Journal of the Operational Research Society*, 41:17–24, 1990.

23 William R. Pulleyblank. *Edmonds, Matching and the Birth of Polyhedral Combinatorics*, pages 181–197. Springer Berlin Heidelberg, 2012.

24 Sharath Raghvendra. A Robust and Optimal Online Algorithm for Minimum Metric Bipartite Matching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM '16)*, volume 60 of *Leibniz International Proceedings in Informatics*, pages 18:1–18:16. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016.

25 Sharath Raghvendra. Optimal Analysis of an Online Algorithm for the Bipartite Matching Problem on a Line, 2018. `arXiv:1803.07206`.

26 E. Reingold and R. Tarjan. On a Greedy Heuristic for Complete Matching. *SIAM Journal on Computing*, 10:676–681, 1981.