PRICING ON PATHS: A PTAS FOR THE HIGHWAY PROBLEM*

FABRIZIO GRANDONI† AND THOMAS ROTHVOß‡

Abstract. In the *highway* problem, we are given an n-edge path graph (the highway), and a set of paths (the drivers), each one with its own budget. For a given assignment of edge weights (the tolls), the highway owner collects from each driver the weight of the associated path, when it does not exceed the budget of the driver, and zero otherwise. The goal is choosing weights so as to maximize the profit. A lot of research has been devoted to this apparently simple problem. The highway problem was shown to be strongly **NP**-hard only recently [Elbassioni, Raman, Ray - '09]. The best-known approximation is $O(\log n/\log\log n)$ [Gamzu, Segev - '10], which improves on the previous-best $O(\log n)$ approximation [Balcan,Blum-'06]. Better approximations are known for a number of special cases. Finding a constant (or better!) approximation algorithm for the general case is a challenging open problem.

In this paper we present a PTAS for the highway problem, hence greatly improving our understanding of the complexity status of this problem. Our result is based on a novel randomized dissection approach, which has some points in common with Arora's quadtree dissection for Euclidean network design [Arora - '98]. The basic idea is enclosing the highway in a bounding path, such that both the size of the bounding path and the position of the highway in it are random variables. Then we consider a recursive $O(1)$-ary dissection of the bounding path, in subpaths of uniform optimal weight. Since the optimal weights are unknown, we construct the dissection in a bottom-up fashion via dynamic programming, while computing the approximate solution at the same time. Our algorithm can be easily derandomized.

The same basic approach provides PTASs also for two generalizations of the problem: the *tollbooth* problem with a constant number of leaves and the *maximum-feasibility subsystem* problem on interval matrices. In both cases the previous best approximation factors are polylogarithmic [Gamzu, Segev - '10, Elbassioni, Raman, Ray, Sitters - '09].

Key words. approximation algorithms, pricing problems, highway problem, tollbooth problem, maximum feasibility subsystem problem

AMS subject classifications. 68W05, 68W20, 68W25, 68W40

1. Introduction. Consider the following setting. We are given a single-road highway, which is partitioned into segments by tollbooths. The highway owner fixes a toll (or *price*) for each segment. A driver traveling between two tollbooths pays the total toll of the corresponding segments. However, if the total toll exceeds the budget of the driver, she will not use the highway. Our goal is to maximize the profit of the highway owner. To that aim, we need to compromise between very low tolls (in which case all the drivers take the highway, but providing a small profit) and very high tolls (in which case no driver takes the highway, and the profit is zero). It is not hard to imagine other applications with a similar nature. For example, the highway segments might be replaced by the links of a (high-bandwidth) telecommunication network. Alternatively, one might interpret the highway as a period of time (e.g., one day), and the segments as time slots (e.g., one slot per minute): now drivers are clients who need a service (e.g., an Internet connection) for a given interval of time.

The *highway* problem formalizes the scenarios above. We are given an n-edge path graph $G = (V, E)$ (the *highway*), and a set $\mathcal{D} = \{D_1, \ldots, D_m\}$ of m paths in G (the *drivers*), each one characterized by a value $b_j \in \mathbb{Q}_{\geq 0}$ (the *budgets*). For a given weight function $w : E \to \mathbb{Q}_{\geq 0}$ (the *tolls*) and a driver D, let $w(D) := \sum_{e \in D} w(e)$

*A preliminary version of this paper appeared in SODA'11 [23].

†IDSIA, University of Lugano, Switzerland, fabrizio@idsia.ch. Partially supported by the ERC Starting Grant NEWNET 279352.

‡Department of Mathematics, University of Washington, USA, rothvoss@uw.edu

be the *weight* of D[1]. Our goal is choosing w so as to maximize the following *profit* function:

$$\sum_{j:w(D_j)\leq b_j} w(D_j).$$

Despite the simplicity of its formulation and its clear relation to applications, there is a huge gap between known approximation and inapproximability results for the highway problem. The problem was shown to be strongly **NP**-hard very recently [18]. The best-known approximation factor is $O(\log n/\log\log n)$ [22] (see also [4]). A quasi-polynomial-time approximation scheme (QPTAS) is given in [20]. This is strong evidence of the existence of a PTAS for the problem. However, even finding a constant approximation is a non-trivial open problem, as witnessed by the number of results on special cases [4, 10, 24, 27].

1.1. Our Results and Techniques. In this paper we present a deterministic polynomial-time approximation scheme (PTAS) for the highway problem, hence greatly improving our understanding of the complexity status of this problem. To achieve our goal, we exploit a novel randomized dissection approach in combination with dynamic programming.

The basic idea is as follows. Let $\varepsilon > 0$ be a small constant. Via simple reductions (see Section 1.3), we can restrict ourselves to the case that optimal weights $w^*(e)$ are in $\{0, 1\}$, and that the sum W^* of the optimal weights along the highway is polynomially bounded in the number n of edges. This introduces a $1 + \Theta(\varepsilon)$ factor in the approximation.

The dynamic program is based on the following strategy. We consider all the subpaths P of the highway, and guess the value $W \in \{0, 1, \ldots, W^*\}$ of the sum of the optimal weights along P. Note that the number of pairs (P, W) is polynomially bounded in n, due to the reductions above.

We next restrict our attention to the drivers $\mathcal{D}(P) := \{D \in \mathcal{D} : \ D \subseteq P\}$ which are entirely contained in P, with the goal of approximating the corresponding optimal profit: The table entry for $(P, W) = (G, W^*)$ will eventually give the desired approximate solution.

If $W \leq \tilde{W}$, for a fixed constant $\tilde{W}$, we simply guess the W edges where the optimum solution puts a weight of one. This provides the optimal profit for drivers in $\mathcal{D}(P)$. Assume now that $W > \tilde{W}$. In this case, by considering all the possible partitions $\overline{P} = \{P_1, \ldots, P_\gamma\}$ of P in γ subpaths, we can guess the partition where each P_i takes a $1/\gamma$ fraction of the weight of P. Here $\gamma \geq 2$ is a sufficiently large constant, depending on ε. Observe that the set $\mathcal{P}_\gamma(P)$ of such partitions has polynomial cardinality. Given $\overline{P}$, for the drivers included into some P_i (i.e., in $\mathcal{D}(P_i)$), we account for the (previously computed) profit of table entry $(P_i, W/\gamma)$.

It remains to consider the profit of drivers $\overline{\mathcal{D}}(P) = \mathcal{D}(P) - \cup_{i=1}^{\gamma}\mathcal{D}(P_i)$ which are contained in P, but not in any P_i. This is also the crux of our method. Each driver $D \in \overline{\mathcal{D}}(P)$ consists of a (possibly empty) subset of consecutive subpaths $P_\ell, P_{\ell+1}, \ldots, P_r$, plus two (possibly empty) subpaths P_{left} and P_{right}, with $P_{left} \subset P_{\ell-1}$ and $P_{right} \subset P_{r+1}$. Observe that, if the budget of D is not exceeded, then each *middle* subpath P_i, $i \in \{\ell, \ldots, r\}$, contributes with an additive term W/γ to the profit

[1]Throughout this paper we denote in the same way a graph and its set of edges: the meaning will be clear from the context.

of D. In particular, this is independent from the way the weight W/γ is distributed along P_i.

The situation is radically different for the boundary subpaths P_{left} and P_{right}: for them the profit can range from 0 to W/γ, depending on the distribution of the weights along $P_{\ell-1}$ and P_{r+1}. In order to implement efficiently the dynamic programming step, we simply neglect the boundary subpaths. In other terms, we replace D with the *shortened* driver $D^s = D - (P_{left} \cup P_{right})$. At this point, we simply account $(r - \ell + 1)W/\gamma$ for the profit of D, if this quantity does not exceed its budget, and zero otherwise. This way we obtain the overall profit for drivers in $\overline{\mathcal{D}}(P)$, and hence in $\mathcal{D}(P)$.

This approach has two opposite drawbacks:

1. The profit computed might be too pessimistic. This is because we do not consider the profit coming from $P_{left} \cup P_{right}$ (in particular, it might be $D = P_{left} \cup P_{right}$, and hence $D^s = \emptyset$).
2. The profit computed might be too optimistic. In fact, it might happen that the weight along D^s is below the budget of D, while the weight along D exceeds it (due to the weight on $P_{left} \cup P_{right}$). In that case we account for a positive profit, while the actual profit is zero.

We solve the second problem by restricting our attention to *good* drivers $D \in \overline{\mathcal{D}}(P)$, i.e. drivers that contain $\Omega(1/\varepsilon)$ many subpaths P_i. It is then sufficient to scale down all the weights at the end of the process by a factor $1 - O(\varepsilon)$ to ensure that the budget of good paths is not exceeded.

Observe that this does not solve the first problem: indeed, it makes it even worse (since we consider less drivers, besides shortening them). At this point, randomization comes into play. We initially enclose the highway in a *bounding path*, i.e. we add a few dummy edges (not used by any driver) to the left and right of the input path graph. Both the length (i.e., the number of edges) of the bounding path and the position of the highway inside it are random variables. To this instance we apply the approach above. Consider a driver D that contributes to the optimal profit. For a proper choice of the random variables, with probability $1 - O(\varepsilon)$, D is considered in the dynamic program for a path P of weight W such that the profit of D is much larger than W/γ. Hence D is good with probability close to one. This introduces a factor $1 + O(\varepsilon)$ in the approximation ratio.

As we will see, the domain of the random variables has polynomial size. Hence, the algorithm above can be easily derandomized by considering all the possible realizations.

We believe that our technique will find other applications, and hence it might be of independent interest. In order to motivate that, we show how to apply it to two related problems (see Section 4):

- The *tollbooth* problem is the generalization of the highway problem where the input graph G is a tree (rather than a path). This problem is **APX**-hard, and the best-known approximation for it $O(\log n / \log \log n)$ [22]. Here we present a PTAS for the special case that G has a constant number of leaves. This might be useful in some applications.
- In the *maximum-feasibility subsystem* problem we are given a set of vectors $a_1, \ldots, a_m \in \mathbb{Q}^n$ and a set of m pairs (ℓ_j, u_j), with $0 \leq \ell_j \leq u_j$ and $j = 1, \ldots, m$. The goal is to compute a vector $w \in \mathbb{Q}^n_{\geq 0}$ such that the *constraint* $\ell_j \leq a_j^T w \leq u_j$ is satisfied by the largest possible number m^* of indexes j. Intuitively, the vectors a_j^T can be interpreted as the rows of a matrix A: the

product $A\,w \in \mathbb{Q}^m$ is what we wish to upper and lower bound. In this paper we restrict to the case that the vectors a_j have entries in $\{0, 1\}$, and the 1's appear consecutively (i.e., A is an *interval matrix*).
Elbassioni, Raman, Ray, and Sitters [19] show that this problem is **APX**-hard. Moreover, they consider the case that one is allowed to violate the lower and upper bounds by a factor $(1 + \varepsilon)$, and present a polylogarithmic approximation algorithm running in polynomial time. They also present a quasi-polynomial time algorithm that computes a solution satisfying (in the mentioned approximate sense) at least m^* constraints[2]. Here we show how to obtain a $(1+\varepsilon)$ approximation in polynomial time in the same framework.

1.2. Related Work. The highway problem was even not known to be **NP**-hard until recently. For example, this is posed as an open problem by Guruswami et al. [24]. Weakly **NP**-hardness was shown via a reduction from *partition* by Briest and Krysta [10]. Very recently, Elbassioni, Raman, and Ray [18] proved strong **NP**-hardness via a reduction from *max-2-SAT*. Balcan and Blum [4] give a $O(\log n)$ approximation for the problem. Their algorithm partitions the paths in groups of different length. Then it applies a constant factor approximation algorithm in [24] for the *rooted* version of the problem, where all drivers contain a given node, to each group separately. The approximation was very recently improved to $O(\log n/\log\log n)$ by Gamzu and Segev [22]. Their algorithm, which also works for the more general tollbooth problem, combines the notion of tree separators with a generalization of the algorithm for the rooted case mentioned before. The QPTAS by Elbassioni, Sitters, and Zhang [20] exploits the profiling technique introduced by Bansal et al. [5]. The basic idea is guessing the approximate shape of the cumulative weights to the left and right of a given edge. This allows one to partition the problem into two subproblems, which can be solved recursively.

There are better approximation results, all based on dynamic programming, for a number of special cases. In [4] a constant approximation is given for the case when all the paths have roughly the same length. An FPTAS is described by Hartline and Koltun [27] for the case when the highway has constant length (i.e., $n = O(1)$). This was generalized to the case of constant-length paths in [24]. In [24] the authors also present an FPTAS for the case when budgets are upper bounded by a constant. An FPTAS is also known [4, 10] for the case when paths induce a laminar family[3].

The *tollbooth* problem is the generalization of the highway problem where G is a tree. A $O(\log n)$ approximation was developed in [18]. As already mentioned, this was very recently improved to $O(\log n/\log\log n)$ [22]. Cygan et al. [17] present a $O(\log\log n)$ approximation for the case of uniform budgets. The tollbooth problem is **APX**-hard [24], and for general graphs it is **APX**-hard even when the graph has bounded degree, the paths have constant length and each edge belongs to a constant number of paths [10].

The highway and tollbooth problems belong to the family of pricing problems with single-minded customers and unlimited supply. Here we are given a set of customers: Each customer wants to buy a subset of items (*bundle*), if its total prize does not exceed her budget. In the highway terminology, each driver is a subset of edges (rather than a path). For this problem a $O(\log n + \log m)$ approximation is given in [24]. This bound was refined in [10] to $O(\log L + \log B)$, where L denotes

[2]The latter result is not a contradiction, since we compare to the optimum solution, which can not violate the inequalities even slightly.

[3]In a laminar family of paths, two paths which intersect are contained one in the other.

the maximum number of items in a bundle and B the maximum number of bundles containing a given item. Chalermsook et al. [13] showed that this problem is hard to approximate within $\log^{1-\varepsilon} n$ for any constant $\varepsilon > 0$, unless $\mathbf{NP} \subseteq \mathbf{ZPTIME}(n^{\log^{\delta} n})$, where $\delta > 0$ is a constant depending on ε. A $O(L)$ approximation is given in [4]. The latter approximation factor is asymptotically the best possible for constant values of L unless $\mathbf{P} = \mathbf{NP}$ as recently proved by Chalermsook et al. [14].

The highway problem has some aspects in common with the well-studied *unsplittable flow* problem on path graphs. In this problem we are given a path graph $G = (V, E)$, with edge capacities and a set of paths D_j, each one characterized by a demand and a profit. The goal is to select a maximum profit subset of paths such that the sum of the demands of selected paths on each edge does not exceed the corresponding capacity. For the special case of unit capacities and demands, a $(2 + \varepsilon)$ approximation is given by Calinescu et al. [11], improving on [7, 28]. Under the *no-bottleneck* assumption, the same approximation guarantee is achieved for the general case by Chekuri, Mydlarz, and Shepherd [16], improving on an earlier constant approximation under the same assumption [12]. The first constant approximation for the general case is given by Bonsma et al. [9], improving on an earlier logarithmic approximation by Bansal et al. [6]. The current best $2+\varepsilon$ approximation factor is due to Anagnostopoulos et al. [2]. A QPTAS for the problem is given in [5], assuming that demands are quasi-polynomially bounded in the number of paths D_j. Recently the latter assumption was removed by Batra et al. [8]. The QPTAS for the highway problem in [20] exploits the same basic technique as in [5]. Our hope is that, in turn, our PTAS for the highway problem will inspire a PTAS for the path-graph unsplittable flow problem. However, this seems to require some new ideas and we leave it as a challenging open problem. There is also a line of research on finding LP relaxations with small integrality gap [1, 12, 15].

For general $0/1$-matrices, the maximum-feasible subsystem problem (with no violation) is not approximable within $\Omega(n^{1/3-\varepsilon})$ for any $\varepsilon > 0$ even for $\ell_j = u_j$, unless $\mathbf{ZPP} = \mathbf{NP}$ [19]. If each row of A contains 3 non-zero arbitrary coefficients, then even $n^{1-\varepsilon}$ approximations are not possible in polynomial time [25] (see also the previous hardness result [21]). The best-known $O(n/\log n)$ approximation for this problem is due to Halldórsson [26].

The technique behind our PTAS resembles Arora's quadtree dissection for Euclidean network design [3]. The basic idea there is to enclose the set of input points into a bounding box, then recursively partition it in a constant number of boxes. This dissection is then randomly shifted. On the resulting random dissection, one applies dynamic programming. We similarly enclose the highway in a bounding path, and partition the latter. Like in Arora's approach, the dissection is randomly shifted. Differently from that case and crucially for our analysis, the size of the bounding path is a random variable as well. Another major difference is that the dissection is not uniform with respect to input properties, but with respect to the optimal weights: for this reason the dissection is constructed in a bottom-up, rather than top-down, fashion via dynamic programming (while computing the approximate solution in parallel).

1.3. Preliminaries. Let $OPT = (w^*, \mathcal{D}^*)$ be the optimum solution, where w^* is the optimal weight function and $\mathcal{D}^*$ is the set of drivers D_j such that $w^*(D_j) \leq b_j$. By *opt* we denote the optimal profit. Our PTAS starts with a sequence of rounding steps to transform the input (and the optimum solution) in a convenient form, while losing only a factor $1 - 2\varepsilon$ in the approximation. Since these steps are rather stan-

dard, we discuss them here, while in Section 2 we will focus on the novel techniques introduced in this paper.

W.l.o.g. we assume $1/(2\varepsilon) \in \mathbb{N}$ and $\varepsilon \leq 1/2$. Let $b_{\max}$ be the largest budget. Recall that m is the number of drivers. After scaling all budgets, one has $b_{\max} = m/\varepsilon^2$. Observe that trivially $opt \geq b_{\max}$. First, we discard all drivers with a budget smaller than $1/\varepsilon$. Next, we round down the budgets to the nearest integer. Any solution to the rounded instance gives a feasible solution of the same value for the original instance. Moreover, the optimal solution to the rounded instance is a good approximation of the original optimum. In fact, $(w, \mathcal{D})$ with $\mathcal{D} := \{D_j \in \mathcal{D}^* \mid b_j \geq 1/\varepsilon\}$ and $w(e) := \frac{w^*(e)}{1+\varepsilon}$ is a feasible solution to the new instance since

$$w(D_j) = \sum_{e \in D_j} \frac{w^*(e)}{1+\varepsilon} \leq \frac{b_j}{1+\varepsilon} \leq \lfloor b_j \rfloor$$

for any $D_j \in \mathcal{D}^*$ with $b_j \geq 1/\varepsilon$. The profit of this solution is

$$\sum_{D_j \in \mathcal{D}} w(D_j) \geq \sum_{D_j \in \mathcal{D}^* : b_j \geq 1/\varepsilon} \frac{w^*(D_j)}{1+\varepsilon} \geq \frac{opt}{1+\varepsilon} - \frac{m}{\varepsilon} \geq (1-2\varepsilon)opt.$$

As observed in [12], the optimal weights for this instance can be assumed to be integral. In fact, given the optimal drivers $\mathcal{D}^*$, the corresponding optimal weights w^* can be computed by solving an ILP whose 0-1 constraint matrix is totally unimodular. Since the largest weight in w^* is trivially not larger than the largest budget (i.e. m/ε^2 after rounding), we can conclude that $w^* : E \to \{0, 1, \ldots, m/\varepsilon^2\}$. By replacing each edge with a path of length m/ε^2, we can further assume $w^* : E \to \{0, 1\}$. Let $W^* = \sum_{e \in E} w^*(e)$ be the total weight of the solution, and $\gamma = (1/\varepsilon)^{1/\varepsilon}$. By adding $W^*\gamma$ dummy edges (not crossed by any driver), say, to the right of the highway, we can assume that $W^* = \gamma^\ell$ for some integer ℓ (in fact, the weight assigned to dummy edges is irrelevant). Observe that $W^* \leq n\, m\gamma/\varepsilon^2$: hence we can guess the value of W^* by trying a polynomial number of possible values.

We call an instance of the highway problem with the properties above *well-rounded*. The discussion above implies the following lemma.

LEMMA 1.1. *For any $\varepsilon > 0$, there is a polynomial reduction from the highway problem to the same problem on well-rounded instances that is approximation-preserving modulo a factor $(1+\varepsilon)$.*

2. A PTAS for the Highway Problem. From the discussion in Section 1.3, we assume that the input instance is well-rounded. Let $\varepsilon > 0$ be a constant parameter, $\delta = 1/(2\varepsilon) \in \mathbb{N}$ and $\gamma = (1/\varepsilon)^{1/\varepsilon}$. Our PTAS `hptas` for the highway problem is described in Figure 1.

In the *Bounding Phase* (B), we first guess the total optimal weight W^* (Step B1). By guessing, we mean that we run the rest of the algorithm for every feasible choice of W^* (there is a polynomial number of such choices). Then, we enclose the highway in a bounding path (Step B2). Both the length of the bounding path and the position of the highway inside the bounding path are carefully chosen functions of two random variables x and y. All the probabilities and expectations in this paper are with respect to the choice of those two variables.

In the *Dynamic Programming Phase* (D), we compute the almost optimal profit $\phi(P, W)$ which can be obtained from the drivers in P by placing W-many 1's along P. In the initialization step (Step D1), we compute profits $\phi(P, (1/\varepsilon)^y)$ by brute force,

Figure 1 PTAS for the highway problem. Here $\delta := 1/(2\varepsilon) \in \mathbb{N}$ and $\gamma = (1/\varepsilon)^{1/\varepsilon}$.

Input: Well-rounded highway instance $G = (V, E)$ and (P_j, b_j), $j = 1, 2, \ldots, m$.
Output: Edge weights $w : E \to \mathbb{Q}_{\geq 0}$
Algorithm:

(B) Bounding Phase:

(B1) Guess the value of the total weight $W^* = \gamma^\ell$, $\ell \in \mathbb{N}$.

(B2) Choose integers $x \in \{1, 2, \ldots, W^*\}$ and $y \in \{1, 2, \ldots, 1/\varepsilon\}$ uniformly and independently at random. Attach a path of length $W^* \cdot ((1/\varepsilon)^y - 1) - x$ to the right of G, and a path of length x to the left of G. Let G_0 be the resulting path graph, and $W' = W^* \cdot (1/\varepsilon)^y$.

(D) Dynamic Programming Phase:

(D1) For every path $P \subseteq G_0$,

$$\phi(P, (1/\varepsilon)^y) = \max_{\substack{w:P\to\{0,1\} \\ w(P)=(1/\varepsilon)^y}} \sum_{\substack{D_j \subseteq P, \\ w(D_j) \leq b_j}} w(D_j).$$

(D2) For every path $P \subseteq G_0$, and for $W = W'/\gamma^q$, $q = \ell - 1, \ell - 2, \ldots, 0$,

$$\phi(P, W) = \max_{\overline{P} \in \mathcal{P}_\gamma(P)} \left\{ \sum_{i=1}^{\gamma} \phi\left(P_i, W/\gamma\right) + \sum_{\substack{D_j \subseteq P, \\ n_j := |\{i : P_i \subseteq D_j\}| \geq \delta, \\ W/\gamma \cdot n_j \leq b_j}} W/\gamma \cdot n_j \right\}.$$

(S) Scaling Phase:

(S1) Derive $w' : G_0 \to \{0, 1\}$ determining the value of $\phi(G_0, W')$.

(S2) Output $w : E \to \mathbb{Q}_{\geq 0}$, where $w(e) = w'(e) \cdot \frac{\delta}{\delta+2}$.

considering all the $\binom{|P|}{(1/\varepsilon)^y}$-many possible ways to place $(1/\varepsilon)^y = O(1)$-many 1's on the edges of P. Here we implicitly assume that the profit is zero if P is too short (hence we maximize over an empty set). In the dynamic programming step (Step D2), we consider the *best* partition $\overline{P} = \{P_1, \ldots, P_\gamma\}$ of P into γ subpaths. The set of candidate partitions is denoted by $\mathcal{P}_\gamma(P)$. We first add to $\phi(P, W)$ the profits $\phi(P_i, W/\gamma)$ for each i. Then we consider the good drivers D_j, i.e. the drivers in P which contain $n_j \geq \delta$ subpaths P_i. For each such driver, we increase $\phi(P, W)$ by the profit associated to the shortened driver $D_j^s = \cup_{P_i \subseteq D_j} P_i$, i.e. $W/\gamma \cdot n_j$, unless this quantity exceeds the budget b_j.

In the final *Scaling Phase* (S), we derive from the dynamic programming table the weights w' determining the value of $\phi(G_0, W')$ (Step S1). Then we restrict our attention to the edges of the (original) highway, and scale the corresponding weights down by $\frac{\delta}{\delta+2}$ (Step S2).

Consider the possible ways in which it is possible to partition the path graph G_0 into γ subpaths, and then recursively partition each subpath, where we halt the recursion at level ℓ (corresponding to γ^ℓ subpaths). We call any such recursive partition of G_0 a *dissection*. Note that the solution returned by the algorithm induces one such dissection.

We remark that the above algorithm would also work by setting deterministically x to, say, 1. This is because the number of dummy edges in the leftmost/rightmost subpaths in each level of any feasible dissection cannot affect the profit of the corre-

sponding solution: indeed, those edges are not used by any driver. In other words, the modified algorithm can mimic the behavior of the original one with a different choice of the dissection (and vice versa). Still, we believe that the current version of the algorithm makes the analysis simpler and more intuitive.

3. Analysis. To avoid any confusion, let n and $\bar{n}$ denote the number of edges in the original and well-rounded instance, respectively. Recall that, for any constant ε, $\bar{n}$ is polynomially bounded in n and m.

LEMMA 3.1. *Algorithm* `hptas` *runs in polynomial time.*

Proof. Since W^* is an integer bounded by $nm\gamma/\varepsilon^2$, its value can be guessed by trying a polynomial number of values. For all the $O(\bar{n}^2)$ choices of P in Step D1, the number of candidate functions w to be considered is $O(\bar{n}^{(1/\varepsilon)^y})$ (i.e., the possible ways in which we can distribute $(1/\varepsilon)^y$ ones in $\bar{n}$ positions). In Step D2, for all the $O(\bar{n}^2)$ choices of P, there are $O(\bar{n}^{\gamma-1})$ possible choices for the P_i's (i.e., the possible ways in which we can partition a path into γ subpaths). The claim follows. □

In the rest of the analysis we consider only the run of the algorithm where W^* is guessed correctly. The next lemma shows that the profit apx of the solution returned by the algorithm essentially coincides with the value $apx_D = \phi(G_0, W')$ that is computed in the dynamic programming phase. Here we crucially exploit the fact that we only consider (good) drivers D_j with large n_j.

LEMMA 3.2. $apx \geq \frac{1}{1+4\varepsilon} apx_D$.

Proof. Let w' and $\mathcal{D}'$ be the weights and the set of drivers determining apx_D. Consider the dissection corresponding to apx_D, and let $n_j = |\{i : P_i \subseteq D_j\}|$ and $D_j^s = \bigcup_{P_i \subseteq D_j} P_i$ be defined with respect to that dissection for each D_j.

For any $D_j \in \mathcal{D}'$, $n_j \geq \delta = 1/(2\varepsilon)$ and $w'(D_j^s) = W/\gamma \cdot n_j \leq b_j$. The difference in weight between D_j and D_j^s only lies in the two sub-intervals owning the endings of D_j, and hence $w'(D_j^s) \leq w'(D_j) \leq \frac{W}{\gamma}(n_j + 2)$. It follows that $w(D_j) = \frac{\delta}{\delta+2} w'(D_j) \leq \frac{n_j}{n_j+2} w'(D_j) \leq n_j \frac{W}{\gamma} \leq b_j$. Hence, D_j contributes to apx with a profit $w(D_j) \geq \frac{\delta}{\delta+2} w'(D_j^s) = \frac{1}{1+4\varepsilon} w'(D_j^s)$. The claim follows since $apx \geq \sum_{D_j \in \mathcal{D}'} w(D_j) \geq \frac{1}{1+4\varepsilon} \sum_{D_j \in \mathcal{D}'} w'(D_j^s) = \frac{1}{1+4\varepsilon} apx_D$. □

It remains to lower bound apx_D in terms of opt. In order to simplify the analysis, suppose that we are given an oracle which, for a given $P \subseteq G_0$ with $w^*(P) = W = W'/\gamma^q$, $q < \ell$, produces a partition $\overline{P}^* = \{P_1^*, \ldots, P_\gamma^*\}$ such that $w^*(P_i^*) = W/\gamma$. Also assume that we remove all the drivers but the ones $\mathcal{D}^*$ in the optimal solution. Consider the variant of Step D where we apply recursively the following Bellman equation

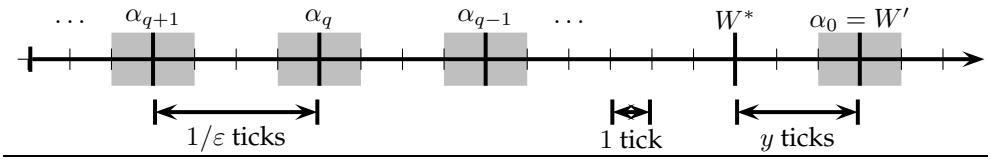
$$\phi'(P, W) = \sum_{i=1}^{\gamma} \phi'\left(P_i^*, W/\gamma\right) + \sum_{\substack{\mathcal{D}^* \ni D_j \subseteq P, \\ n_j := |\{i : P_i^* \subseteq D_j\}| \geq \delta, \\ W/\gamma \cdot n_j \leq b_j}} W/\gamma \cdot n_j,$$

until $W = (1/\varepsilon)^y$, in which case we use brute force to compute the optimal weights like in Step D1. It is not hard to see that $apx_O := \phi'(G_0, W')$ is a lower bound on apx_D.

LEMMA 3.3. $apx_D \geq apx_O$.

Hence it is sufficient to lower bound apx_O. The value apx_O is associated to a unique *optimal dissection*. With the same notation as in the proof of Lemma 3.2, we let, for a given driver D_j, n_j and D_j^s be defined with respect to the optimal dissection.

Figure 2 Log-scale axis. The regions of risky weights are grayshaded.



We next say that a subpath in the optimal dissection is *at level* $q \in \{0, 1, \ldots, \ell\}$ if its optimal weight is W'/γ^q. Similarly, we say that a driver D_j is at level q in the optimal dissection if it is contained in a subpath of level q, but not $q+1$.

Let $\alpha_q = W'/\gamma^q$. Consider any driver $D_j \in \mathcal{D}^*$, with $\alpha_{q+1} < w^*(D_j) \leq \alpha_q$. We call D_j *good* if it is at level ℓ in the dissection, or it is at level $q < \ell$ and it contains at least δ subpaths of level $q+1$ (i.e., $n_j \geq \delta$).

Observe that good drivers D_j contribute to the value of apx_O with a profit $w^*(D_j^s) \geq w^*(D_j) \cdot \frac{\delta}{\delta+2} = \frac{1}{1+4\varepsilon} \cdot w^*(D_j)$. Hence, it is sufficient to show that a given driver in $\mathcal{D}^*$ is good with probability close to one.

LEMMA 3.4. *Each driver $D_j \in \mathcal{D}^*$ is good with probability at least $1 - 3\varepsilon$.*

Proof. Let us upper bound the probability that a driver D_j is *bad* (i.e., not good). We say that driver D_j is *risky* if

$$\exists q : \varepsilon\, \alpha_q < w^*(D_j) < \frac{1}{\varepsilon}\alpha_q.$$

Consider a log-scale axis and term *tick* the distance that corresponds to a factor of $1/\varepsilon$. Then consecutive α_q's have a distance of $1/\varepsilon$ ticks to each other (see Figure 2). The region of risky weights w.r.t. a specific α_q is the interval $]\varepsilon \cdot \alpha_q, \alpha_q/\varepsilon[$, hence on the log-scaled axis it is an (open) interval of 2 ticks length. The random choice of y yields that all α_q's are simultaneously shifted by $y \in \{1, \ldots, 1/\varepsilon\}$ ticks to the right. Hence for each value of $w^*(D_j)$ at most 2 out of $1/\varepsilon$ choices of y cause that D_j is risky:

$$\Pr[D_j \text{ is risky}] \leq 2\varepsilon.$$

Next condition on the event that D_j is not risky. This implies that there exists a value q such that either $w^*(D_j) \leq \varepsilon\, \alpha_{q-1}$ with $q - 1 = \ell$ or

$$\frac{1}{\varepsilon}\alpha_q \leq w^*(D_j) \leq \varepsilon\, \alpha_{q-1}.$$

In the second case, D_j deterministically contains at least $1/\varepsilon - 1 \geq 1/(2\varepsilon) = \delta$ many level q subpaths. In both cases, since the random shift x is chosen uniformly at random from $\{1, \ldots, W^*\}$ and W^* is a multiple of α_{q-1}[4] we furthermore have

$$\Pr[D_j \text{ is at level } < q-1] \leq \frac{w^*(D_j)}{\alpha_{q-1}} \leq \varepsilon.$$

Applying the union bound, we obtain that driver D_j is bad with probability at most 3ε. □

[4]Except the case when $\alpha_{q-1} = W'$, but then deterministically the driver D_j cannot cross the boundary.

COROLLARY 3.5. $E[apx_O] \geq \frac{1-3\varepsilon}{1+4\varepsilon} opt$.

Proof. One has:

$$E[apx_O] \geq E\Big[\sum_{\substack{D_j \in \mathcal{D}^*,\\ D_j \text{ good}}} w^*(D_j^s)\Big] \geq \frac{1}{1+4\varepsilon} E\Big[\sum_{\substack{D_j \in \mathcal{D}^*,\\ D_j \text{ good}}} w^*(D_j)\Big]$$
$$\geq \frac{1-3\varepsilon}{1+4\varepsilon} \sum_{D_j \in \mathcal{D}^*} w^*(D_j) = \frac{1-3\varepsilon}{1+4\varepsilon} opt.$$

□

Now we have all the ingredients to prove the main result of this paper.

THEOREM 3.6. *There is a randomized PTAS for the highway problem.*

Proof. Consider the randomized algorithm that first transforms the input into a well-rounded instance as described in Section 1.3, and then applies algorithm `hptas`. From Lemmas 1.1 and 3.1, this algorithm takes polynomial time. By Lemma 1.1, Lemma 3.2, Lemma 3.3, and Corollary 3.5, the approximation ratio of the algorithm is $\frac{(1+4\varepsilon)^2(1+\varepsilon)}{1-3\varepsilon}$. □

The PTAS in Theorem 3.6 can be derandomized by considering all the (polynomially many) choices of x and y in Step B2.

COROLLARY 3.7. *There is a deterministic PTAS for the highway problem.*

4. Extensions. In this section we extend our approach to two variants of the highway problem.

4.1. Tollbooth with a Constant Number of Leaves. We next describe a PTAS for the tollbooth problem, when the input graph G is a tree with a constant number $\theta = O(1)$ of leaves. Recall that the problem is **APX**-hard when the number of leaves is arbitrary (hence a PTAS can not be expected in that case).

We start by sketching our algorithm and analysis. By the same arguments as in the highway case, we assume that optimal weights w^* are $0/1$-valued, and that their sum W^* is bounded by a polynomial in n. We choose an arbitrary leaf $s(G)$ of G as a *source*, and call the other leaves *sinks*. Analogously, given any subtree T of G, we call the leaf $s(T)$ of T that is closest to $s(G)$, the *source* of T. The other leaves of T are called *sinks* of T. By appending a path of length $W^*\gamma$ to each sink, we can assume that the total weight along each source-sink pair is $\tilde{W} := \gamma^\ell$ for some integer ℓ. The resulting instance is *well-rounded*.

Imagine to split G at any node whose w^*-distance from $s(G)$ is an integer multiple of $\tilde{W}/\gamma$. In such a way we obtain a forest $\overline{T} = \{T_1, \ldots, T_q\}$ of subtrees with the following property: any source-sink path in T_i has weight $\tilde{W}/\gamma$. We iterate this process until the total weight which has to be installed on the subtree reaches a constant value. We call this dissection *optimal*.

Consider a driver D_j and let T be the smallest subtree in the optimal dissection that fully contains D_j. Suppose $W = \tilde{W}/\gamma^q$ is the weight that w^* installs on any source-sink path of T. Let $\overline{T} = \{T_1, \ldots, T_q\}$ be the partition of T in the optimal dissection. We say that D_j *crosses* T_i if it contains exactly one source-sink path of T_i. We say that driver D_j is *good* if the number n_j of crossed subtrees is at least a large constant $\delta := \frac{1}{2\varepsilon}$. Also in this case, we can define a shortened driver $D_j^s = \bigcup_{T_i \text{ crossed by } D_j} (T_i \cap D_j)$. However note that in this case D_j^s might consist of two disjoint paths. (In particular, this might happen if D_j does not lie along a source-sink path of G).

Analogously to the highway case, it is sufficient to show that the profit coming from shortened drivers is large with respect to the optimal dissection. Then for subtrees T of the instance and weights W, we compute table entries $\phi(T, W)$ giving the optimum profit that can be obtained from the shortened paths of good drivers $D_j \subseteq T$, in such a way that on each path from $s(T)$ to any other leaf of T one installs a total weight of W.

A detailed description of the algorithm is given in Figure 3. By $\mathcal{P}_\gamma(T)$ we denote the set of potential dissections of subtree T into a forest $\overline{T} = \{T_1, \ldots, T_{q(\overline{T})}\}$. Observe that each source-sink path of T can contain at most $\gamma - 1$ break-points. Consequently, the number $q(\overline{T})$ of subtrees in each candidate forest is at most $\gamma \cdot \theta$. It follows that the cardinality of $\mathcal{P}_\gamma(T)$ is polynomially bounded when the number θ of leaves of G is constant.

Figure 3 PTAS for the tollbooth problem for a constant number of leaves. Here $\delta := 1/(2\varepsilon) \in \mathbb{N}$ and $\gamma = (1/\varepsilon)^{1/\varepsilon}$.

Input: Well-rounded tollbooth instance $G = (V, E)$ and (D_j, b_j), $j = 1, 2, \ldots, m$.
Output: Edge weights $w : E \to \mathbb{Q}_{\geq 0}$.
Algorithm:

(B) Bounding Phase:

(B1) Guess the value of the weight $\tilde{W} = \gamma^\ell$, $\ell \in \mathbb{N}$.

(B2) Choose integers $x \in \{1, 2, \ldots, \tilde{W}\}$ and $y \in \{1, 2, \ldots, 1/\varepsilon\}$ uniformly and independently at random. Attach a path of length $\tilde{W} \cdot ((1/\varepsilon)^y - 1) - x$ to each sink of G, and a path of length x to the source of G. Let G_0 be the resulting tree, and $W' = \tilde{W} \cdot (1/\varepsilon)^y$.

(D) Dynamic Programming Phase:

(D1) For every subtree $T \subseteq G_0$,

$$\phi(T, (1/\varepsilon)^y) = \max_{\substack{w:T\to\{0,1\} \\ w(T)=(1/\varepsilon)^y}} \sum_{\substack{D_j \subseteq T, \\ w(D_j) \leq b_j}} w(D_j).$$

(D2) For every subtree $T \subseteq G_0$, $W = W'/\gamma^q$, and $q = \ell - 1, \ell - 2, \ldots, 0$,

$$\phi(T, W) = \max_{\overline{T} \in \mathcal{P}_\gamma(T)} \left\{ \sum_{i=1}^{q(\overline{T})} \phi\left(T_i, W/\gamma\right) + \sum_{\substack{D_j \subseteq T \\ n_j := |\{i:\ D_j \text{ crosses } T_i\}| \geq \delta \\ W/\gamma \cdot n_j \leq b_j}} W/\gamma \cdot n_j \right\}$$

(S) Scaling Phase:

(S1) Derive $w' : G_0 \to \{0, 1\}$ determining the value of $\phi(G_0, W')$.

(S2) Output $w : E \to \mathbb{Q}_{\geq 0}$, where $w(e) = w'(e) \cdot \frac{\delta}{\delta+4}$.

THEOREM 4.1. *There is a deterministic PTAS for the tollbooth problem with a constant number of leaves.*

Proof. Consider the randomized algorithm described in Figure 3: this algorithm can be derandomized by considering all the possible values of random variables x and y. Assume $0 < \varepsilon \leq \frac{1}{8}$ without loss of generality.

Like in the highway case, let us restrict our attention to the dissection corresponding to the optimal weights, and let us discard drivers that do not provide any

profit in the optimal solution.

We start by showing that any residual driver D_j is good with probability at least $1 - 3\varepsilon$. Let us call a driver D_j *straight* if it lays along a source-sink path of G, and *bent* otherwise. By exactly the same argument as in the highway case, a straight path is good with probability at least $(1 - 3\varepsilon)$. Hence consider a bent driver D_j, and let D'_j and D''_j be the two straight subpaths that partition D_j. Paths D'_j and D''_j have a common endpoint, which is the node of D_j which is closest to the sink of G. Without loss of generality, $w^*(D'_j) \geq w^*(D''_j)$. With the same notation as in the highway case, and by a similar argument, with probability at least $1 - 2\varepsilon$, there is a q such that $\frac{1}{\varepsilon}\alpha_q \leq w^*(D'_j) \leq \varepsilon\alpha_{q-1}$. When this happens, D'_j is at level q in the dissection with probability at least $1 - \varepsilon$. Conditioning on the latter event, by the way the dissection is constructed and being $w^*(D''_j) \leq w^*(D'_j)$, D''_j is at level not smaller than q in the dissection. This implies that D_j is at level q as well. We can conclude that D_j crosses at least $\frac{1}{\varepsilon} - 4 \geq \frac{1}{2\varepsilon} = \delta$ many level q subtrees. The -4 here comes from the fact that the portion of D_j not crossing any subtree consists of at most 4 source-sink subpaths (2 for D'_j and 2 for D''_j, if D_j is bent). Altogether, D_j is good with probability at least $1 - 3\varepsilon$.

Given that D_j is good, the portion of D_j crossing subtrees at level $q + 1$ has weight at least $\frac{\delta}{\delta+4}w^*(D_j)$. This is by the same argument as above. Furthermore, the budget of D_j in the dynamic program is violated at most by a factor $\frac{\delta+4}{\delta}$: hence scaling the weights by $\frac{\delta}{\delta+4}$ in Step (S2) guarantees that good paths contribute to the actual profit. Considering that the initial rounding introduces a factor $1 + \varepsilon$ in the approximation, altogether the solution produced by the algorithm gives profit at least $(\frac{\delta}{\delta+4})^2 \cdot \frac{1-3\varepsilon}{1+\varepsilon} opt = \frac{1-3\varepsilon}{(1+8\varepsilon)^2(1+\varepsilon)} opt$ in expectation. □

4.2. Maximum-Feasible Subsystem for Interval Matrices. In this section we give a multi-criteria PTAS for the maximum-feasible subsystem problem on interval matrices. Indeed, we are able to achieve the same result for a weighted generalization of the problem, where we are also given profits $v_1, \ldots, v_m \in \mathbb{Q}_{\geq 0}$, and our goal is to compute weights w so as to maximize $\sum_{j:\ell_j \leq a_j^T w \leq u_j} v_j$. We next denote this weighted generalization as MAXFS, and let *opt* be the corresponding optimal value. We will prove the following theorem.

THEOREM 4.2. *Given an instance of* MAXFS *and any fixed $\varepsilon > 0$, one can compute in deterministic polynomial time a weight function $w \geq \mathbf{0}$ and a set $J \subseteq \{1, \ldots, m\}$ such that $\sum_{j \in J} v_j \geq (1 - \varepsilon)opt$ and $\ell_j \leq a_j^T w \leq (1 + \varepsilon)u_j$ for all $j \in J$.*

Observe that in the above claim only the upper bounds are violated[5].

Let us sketch our algorithms and analysis. By standard arguments, one can round the profits v_j such that they become integers between 0 and m/ε. Then each constraint j can be replaced by v_j many constraints with unit profit. It is sufficient to find a solution w that satisfies $opt/(1 + O(\varepsilon))$ many constraints j approximately, i.e. $\ell_j/(1 + O(\varepsilon)) \leq a_j^T w \leq u_j(1 + O(\varepsilon))$. The claim then follows by choosing ε accordingly smaller and scaling w by a factor $1 + O(\varepsilon)$.

It is maybe easier to think of MAXFS as a variant of the highway problem where:

1. the consecutive 1's in each row j define a driver D_j in a path graph $G = (V, E)$,
2. each driver D_j, besides having a budget $b_j = u_j$, also has a minimum amount of money ℓ_j that she wants to spend, and

[5]We might similarly violate only the lower bounds.

3. the goal now is to maximize the number of *satisfied* drivers who take the highway (rather than maximizing the profit). Here w can be interpreted as a vector of weights.

Let $OPT = (w^*, \mathcal{D}^*)$ be the optimal solution and define $W^* := \sum_{e \in E} w^*(e)$. Abbreviate $\ell_{\max} := \max\{\ell_j \mid j = 1, \ldots, m\}$. Observe that w.l.o.g. $w^*(e) \leq \ell_{\max}$ on all edges (otherwise one can lower $w^*(e)$ without violating any extra constraint). Hence, $W^* \leq n \cdot \ell_{\max}$ (as a consequence, we can also assume $u_j \leq n \cdot \ell_{\max}$ for all j). Since interval matrices are totally unimodular, we can also assume that $w^*(e) \in \mathbb{Z}_{\geq 0}$ for all $e \in E$. By adding a dummy edge to the left of the path graph (i.e., a zero column to the left of the matrix), we can assume that $W^* = (1/\varepsilon)^{\ell/\varepsilon}$ for some $\ell \in \mathbb{N}$. We also attach a dummy edge to the right of the graph.

Recall that in our highway PTAS we duplicate edges in order to obtain $0/1$ weights. The goal is guaranteeing that we can partition the total optimal weight in γ^i pieces, $i = 1, \ldots, \ell$, without splitting any edge. This was possible since, by preprocessing, we can assume that optimal edge weights are polynomially bounded in m. We are not able to enforce the same property here (in particular, optimal edge weights might depend exponentially on $\log \ell_{max}$). However, as we will detail later, it is sufficient to duplicate each edge $\gamma \cdot \ell \cdot m$ times to achieve the same goal[6]. Altogether, we obtain a *well-rounded* instance G_0 with the following properties:

1. between any two nodes that are starting point or end point of some driver, one has at least $\gamma \cdot \ell \cdot m$ edges;
2. the weight of the optimal solution is a power of $(1/\varepsilon)^{1/\varepsilon}$;
3. at both endings of the highway we have $\gamma \cdot \ell \cdot m$ many edges that are not used by any driver.

Our algorithm applies for such well-rounded instances and begins by guessing W^*. Since W^* is a power of $(1/\varepsilon)^{1/\varepsilon}$, there are at most a polynomial number of candidate values. Recall that the randomization in the algorithms before was used to create a new probabilistic optimal solution. The careful reader might have noticed that the random choice of x can also be moved to the analysis. To simplify a later derandomization, in the algorithm we only choose $y \in \{1, \ldots, 1/\varepsilon\}$ uniformly at random and approximate a solution that installs a total weight of $W' = (1/\varepsilon)^y \cdot W^*$ on the edges. For any subpath $P \subseteq G_0$, we compute table entries $\phi(P, W)$ over all weight assignments $w : P \to \mathbb{Z}_{\geq 0}$, with $w(P) = W$, and over all possible dissections of P, with the goal of maximizing the number of drivers D_j such that:

1. D_j is fully contained in P,
2. D_j is good in the same sense as in the highway case, and
3. $\ell_j/(1+4\varepsilon) \leq w(D_j^s) \leq u_j$ (the shortened driver is *approximately satisfied*).

The number of such table entries is bounded by a polynomial in n, m and $\log \ell_{\max}$, since we only consider values W of the form W'/γ^i. Eventually we output the solution $(w, \mathcal{D}')$ that attains the value $\phi(G_0, W')$. Using the arguments in Lemma 3.4 and Lemma 3.5, one can show that $E[\phi(G_0, W')] \geq (1 - 3\varepsilon)opt$. Similar to Lemma 3.2, one has $\ell_j/(1+4\varepsilon) \leq w(D_j) \leq u_j(1+4\varepsilon)$ for any $D_j \in \mathcal{D}'$. Theorem 4.2 then follows.

In more detail, consider the algorithm in Figure 4. Recall that a driver D_j *belongs* to a path P in a dissection, if P is the maximal path with $D_j \subseteq P$. Suppose the driver D_j indeed belongs to P and the dissection splits P into $\overline{P} = \{P_1, \ldots, P_\gamma\}$. Then D_j is termed *good* if the number of P_i's with $P_i \subseteq D_j$ is at least $\delta = \frac{1}{2\varepsilon}$. The algorithm computes table entries $\phi(P, W)$ representing the maximum number of good

[6]The same approach can be used in the highway problem as well, though it is not crucial to obtain a polynomial running time in that case.

Figure 4 PTAS with $1+O(\varepsilon)$-violation for the MAXFS problem. Here $\delta := 1/(2\varepsilon) \in \mathbb{N}$ and $\gamma = (1/\varepsilon)^{1/\varepsilon}$.

Input: Well-rounded MAXFS instance $G_0 = (V, E)$ and (P_j, ℓ_j, u_j), $j = 1, 2, \ldots, m$.
Output: Edge weights $w : E \to \mathbb{Z}_{\geq 0}$, drivers $\mathcal{D}' \subseteq \mathcal{D}$.
Algorithm:

(B) Bounding Phase:

(B1) Guess the value of the total weight $W^* = \gamma^\ell$, $\ell \in \mathbb{N}$.

(B2) Choose $y \in \{1, \ldots, 1/\varepsilon\}$ uniformly at random. Define $W' = W^* \cdot (1/\varepsilon)^y$.

(D) Dynamic Programming Phase:

(D1) For every path $P \subseteq G_0$,

$$\phi(P, (1/\varepsilon)^y) = \max_{\substack{w: P \to \mathbb{Z}_{\geq 0} \\ w(P) = (1/\varepsilon)^y}} \big|\{D_j \subseteq P \mid \ell_j/(1+4\varepsilon) \leq w(D_j) \leq u_j\}\big|.$$

For every path P with no $D_j \subseteq P$, define $\phi(P, W) = 0$ for any $W = W'/\gamma^q$, $q = 0, \ldots, \ell - 1$.

(D2) For every path $P \subseteq G_0$, and for $W = W'/\gamma^q$, $q = \ell - 1, \ell - 2, \ldots, 0$,

$$\phi(P, W) = \max_{\overline{P} \in \mathcal{P}_\gamma(P)} \left\{ \sum_{i=1}^{\gamma} \phi\left(P_i, W/\gamma\right) + \left| \left\{ D_j \subseteq P \mid \begin{smallmatrix} n_j := |\{i : P_i \subseteq D_j\}| \geq \delta, \\ \ell_j/(1+4\varepsilon) \leq \frac{W}{\gamma} \cdot n_j \leq u_j \end{smallmatrix} \right\} \right| \right\}.$$

(O) Output Phase:

(O1) Derive $w : E \to \mathbb{Z}_{\geq 0}$ and $\mathcal{D}' \subseteq \mathcal{D}$ determining the value of $\phi(G_0, W')$.

(O2) Output $(w, \mathcal{D}')$.

drivers $D_j \subseteq P$ that can be approximately satisfied under the constraint $w(P) = W$. The main difference with the highway algorithm is that, if we reach a path P not containing any driver D_j, then we define $\phi(P, W) = 0$.

First note that the number of table entries is bounded by a polynomial in n and $\log \ell_{\max}$. Hence, the table entries can be computed in time $poly(n, m, \log \ell_{\max})$.

Next, we argue why the value of the computed table entry is not much worse in expectation than the optimal number of satisfiable drivers.

LEMMA 4.3. *The final table entry satisfies $E[\phi(G_0, W')] \geq (1 - 3\varepsilon)opt$.*

Proof. Let $w^* : E \to \mathbb{Z}_{\geq 0}$ be the optimal weight function of total weight W^*. Recall that we have inserted dummy edges to the left and to the right, not contained in any driver D_j. We choose an integer $x \in \{1, 2, \ldots, W^*\}$ uniformly at random. Then increase the total weight on the dummy edges to the right by $W^* \cdot ((1/\varepsilon)^y - 1) - x$ and the weight on the dummy edges to the left by x. The total weight of w^* is now indeed $W' = W^* \cdot (1/\varepsilon)^y$. It now suffices to show the promised bound on $E[\phi(G_0, W')]$ over the random choices of y and x.

Recall that for MAXFS we could not assume that all edges carry just unit weight. Hence we need to argue, why there still is a proper dissection induced by w^*, when each edge is replaced by just $\gamma \cdot \ell \cdot m$ many edge segments. To see this, imagine the path graph G^* that indeed emerges by replacing any edge e with $w^*(e)$ many edges. As in previous sections, there is a proper dissection induced by w^* — potentially with an exponential number of leaves. We think of this dissection to be constructed in a top-down fashion, where the dynamic program truncates the dissection at *empty* paths, that do not contain any driver. How many paths (or nodes in the dissection

tree) can this truncated dissection have? Any of the m drivers is fully contained in no more than ℓ many paths (which is the depth of the dissection tree). And any remaining empty path must have a father that is non-empty. Hence the number of paths P in the truncated dissection tree is bounded by $\gamma \cdot \ell \cdot m$. Since we replaced any edge in the original graph by that many edge segments, this truncated dissection also exists in G_0.

Again, by Lemma 3.4, if we consider the (truncated) dissection of G_0 which is induced by the optimal solution, any driver D_j is good with probability at least $1-3\varepsilon$. Suppose that D_j is good and satisfied in the optimal solution, i.e. $\ell_j \leq w^*(D_j) \leq u_j$. Then

$$w^*(D_j^s) \geq \frac{\delta}{\delta+2} w^*(D_j) \geq \frac{\ell_j}{1+4\varepsilon}$$

and of course $w^*(D_j^s) \leq w^*(D_j) \leq u_j$. In other words, D_j would be included by the dynamic program. The claim follows again by linearity of expectation. □

Finally we argue that the returned drivers are approximately satisfied by the computed weight function.

LEMMA 4.4. *Let $(w, \mathcal{D}')$ be the returned solution. For every driver $D_j \in \mathcal{D}'$, one has $\ell_j/(1+4\varepsilon) \leq w(P_j) \leq u_j(1+4\varepsilon)$.*

Proof. Again let D_j^s be the shortened driver of D_j w.r.t. the dissection induced by the computed weight function w. First of all $w(D_j) \geq w(D_j^s) \geq \ell_j/(1+4\varepsilon)$. Next, the driver D_j is good, hence

$$w(D_j) \leq \frac{\delta+2}{\delta} w(D_j^s) = (1+4\varepsilon) w(D_j^s) \leq (1+4\varepsilon) \cdot u_j.$$

□

We observe that the above algorithm can be easily derandomized by trying out all $1/\varepsilon$ many choices of y. In total, Theorem 4.2 follows.

REFERENCES

[1] A. Anagnostopoulos, F. Grandoni, S. Leonardi, and A. Wiese. Constant integrality gap LP formulations of unsplittable flow on a path. In *International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 25–36, 2013.

[2] A. Anagnostopoulos, F. Grandoni, S. Leonardi, and A. Wiese. A mazing 2+ε approximation for unsplittable flow on a path. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 26–41, 2014.

[3] S. Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *Journal of the ACM*, 45(5):753–782, 1998.

[4] M.-F. Balcan and A. Blum. Approximation algorithms and online mechanisms for item pricing. In *ACM Conference on Electronic Commerce*, pages 29–35, 2006.

[5] N. Bansal, A. Chakrabarti, A. Epstein, and B. Schieber. A quasi-PTAS for unsplittable flow on line graphs. In *ACM Symposium on Theory of computing (STOC)*, pages 721–729, 2006.

[6] N. Bansal, Z. Friggstad, R. Khandekar, and R. Salavatipour. A logarithmic approximation for unsplittable flow on line graphs. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 702–709, 2009.

[7] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor, and B. Schieber. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM*, 48(5):1069–1090, 2001.

[8] J. Batra, N. Garg, A. Kumar, T. Mömke, and A. Wiese. New approximation schemes for unsplittable flow on a path. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 47–58, 2015.

[9] P. Bonsma, J. Schulz, and A. Wiese. A constant factor approximation algorithm for unsplittable flow on paths. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 47–56, 2011.

[10] P. Briest and P. Krysta. Single-minded unlimited supply pricing on sparse instances. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1093–1102, 2006.

[11] G. Calinescu, A. Chakrabarti, H. J. Karloff, and Y. Rabani. Improved approximation algorithms for resource allocation. In *International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 401–414, 2002.

[12] A. Chakrabarti, C. Chekuri, A. Gupta, and A. Kumar. Approximation algorithms for the unsplittable flow problem. *Algorithmica*, 47(1):53–78, 2007.

[13] P. Chalermsook, J. Chuzhoy, S. Kannan, and S. Khanna. Improved hardness results for profit maximization pricing problems with unlimited supply. In *International Workshop on Approximation, Randomization, and Combinatorial Optimization (APPROX-RANDOM)*, pages 73–84, 2012.

[14] P. Chalermsook, B. Laekhanukit, and D. Nanongkai. Independent set, induced matching, and pricing: Connections and tight (subexponential time) approximation hardnesses. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 370–379, 2013.

[15] C. Chekuri, A. Ene, and N. Korula. Unsplittable flow in paths and trees and column-restricted packing integer programs. In *International Workshop on Approximation, Randomization, and Combinatorial Optimization (APPROX-RANDOM)*, pages 42–55, 2009.

[16] C. Chekuri, M. Mydlarz, and F. B. Shepherd. Multicommodity demand flow in a tree and packing integer programs. *ACM Transactions on Algorithms*, 3(3), 2007.

[17] M. Cygan, F. Grandoni, S. Leonardi, M. Pilipczuk, and P. Sankowski. A path-decomposition theorem with applications to pricing and covering on trees. In *European Symposium on Algorithms (ESA)*, pages 349–360, 2012.

[18] K. M. Elbassioni, R. Raman, S. Ray, and R. Sitters. On profit-maximizing pricing for the highway and tollbooth problems. In *International Symposium on Algorithmic Game Theory (SAGT)*, pages 275–286, 2009.

[19] K. M. Elbassioni, R. Raman, S. Ray, and R. Sitters. On the approximability of the maximum feasible subsystem problem with 0/1-coefficients. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1210–1219, 2009.

[20] K. M. Elbassioni, R. Sitters, and Y. Zhang. A quasi-PTAS for profit-maximizing pricing on line graphs. In *European Symposium on Algorithms (ESA)*, pages 451–462, 2007.

[21] U. Feige and D. Reichman. On the hardness of approximating max-satisfy. *Information Processing Letters*, 97(1):31 – 35, 2006.

[22] I. Gamzu and D. Segev. A sublogarithmic approximation for highway and tollbooth pricing. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 582–593, 2010.

[23] F. Grandoni and T. Rothvoß. Pricing on paths: A PTAS for the highway problem. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 675–684, 2011.

[24] V. Guruswami, J. D. Hartline, A. R. Karlin, D. Kempe, C. Kenyon, and F. McSherry. On profit-maximizing envy-free pricing. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1164–1173, 2005.

[25] V. Guruswami and P. Raghavendra. A 3-query PCP over integers. In *ACM Symposium on Theory of computing (STOC)*, pages 198–206, 2007.

[26] M. M. Halldorsson. Approximations of weighted independent set and hereditary subset problems. *Journal of Graph Algorithms and Applications*, 4(1), 2000.

[27] J. D. Hartline and V. Koltun. Near-optimal pricing in near-linear time. In *Workshop on Algorithms and Data Structures (WADS)*, pages 422–431, 2005.

[28] C. A. Phillips, R. N. Uma, and J. Wein. Off-line admission control for general scheduling problems. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 879–888, 2000.