Packing Cars into Narrow Roads: PTASs for Limited Supply Highway

Fabrizio Grandoni
IDSIA, USI-SUPSI, Switzerland
fabrizio@idsia.ch

Andreas Wiese
Department of Industrial Engineering and Center for Mathematical Modeling, Universidad de Chile, Chile
awiese@dii.uchile.cl

Abstract

In the *Highway* problem, we are given a path with n edges (the highway), and a set of m drivers, each one characterized by a subpath and a budget. For a given assignment of edge prices (the tolls), the highway owner collects from each driver the total price of the associated path when it does not exceed drivers's budget, and zero otherwise. The goal is to choose the prices to maximize the total profit. A PTAS is known for this (strongly NP-hard) problem [Grandoni,Rothvoss-SODA'11,SICOMP'16].

In this paper we study the *limited supply* generalization of Highway, that incorporates capacity constraints. Here the input also includes a capacity $u_e \geq 0$ for each edge e; we need to select, among drivers that can afford the required price, a subset such that the number of drivers that use each edge e is at most u_e (and we get profit only from selected drivers). To the best of our knowledge, the only approximation algorithm known for this problem is a folklore $O(\log m)$ approximation based on a reduction to the related Unsplittable Flow on a Path problem (UFP). The main result of this paper is a PTAS for limited supply highway.

As a second contribution, we study a natural generalization of the problem where each driver i demands a different amount d_i of capacity. Using known techniques, it is not hard to derive a QPTAS for this problem. Here we present a PTAS for the case that drivers have uniform budgets. Finding a PTAS for non-uniform-demand limited supply highway is left as a challenging open problem.

2012 ACM Subject Classification Theory of computation → Packing and covering problems; Theory of computation → Packing and covering problems

Keywords and phrases approximation algorithms, pricing problems, highway problem, unsplittable flow on a path

Digital Object Identifier 10.4230/LIPIcs.ESA.2019.51

Funding *Fabrizio Grandoni*: Partially supported by the SNSF Excellence Grant 200020B_182865/1.
Andreas Wiese: Partially supported by the Fondecyt Regular grant 1170223.

1 Introduction

In the *Highway* problem we are given a path graph $G = (V, E)$ with n edges (the *highway*) and a set D of m drivers. Each driver i is characterized by a subpath P_i of G, and by a budget $B_i \in \mathbb{N}^+$. We have to fix a price $p_e \geq 0$ one each edge e (the same for all drivers). Then, for each driver i, we get a profit of $p(i) := \sum_{e \in P_i} p_e$ (i.e., the total price over the edges *used* by i), provided that $p(i) \leq B_i$, and otherwise 0. Intuitively, each driver wishes to travel along subpath P_i, but it is not going to do that if the total requested price exceeds her budget. Our goal is to choose the prices to maximize the total profit from all drivers.

It is not hard to imagine applications for this problem, besides the obvious one suggested by its name. For example, highway edges might represent links of a (high-bandwidth) telecommunication network. Alternatively, one might interpret the highway as a period of

27th Annual European Symposium on Algorithms (ESA 2019).
Editors: Michael A. Bender, Ola Svensson, and Grzegorz Herman; Article No. 51; pp. 51:1–51:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

time, and the edges as time slots: now drivers are clients who need a service for a given interval of time.

Highway is well studied. It was shown to be weakly NP-hard in [9] via a reduction from *Partition*, and strongly NP-hard in [20] via a reduction from *Max-2-SAT*. There is a simple $O(\log m)$-approximation that works for much more general instances. This was improved to $O(\log n)$ in [3] using ideas in [26], and to $O(\log n/\log\log n)$ in [22]. A QPTAS for the problem was presented in [21]. Finally, a PTAS was given in [25].

In this paper we study the *Limited Supply Highway* problem (Ls-Highway), which is a natural generalization of Highway with capacity constraints. Here we are additionally given an integral capacity $u_e \in \mathbb{N}^+$ for each edge e. A solution is now given by a price $p_e \geq 0$ on each edge e plus a subset $S \subseteq D$ of drivers that satisfy the following *capacity constraint*: the total number of selected drivers that use each edge e is at most u_e, i.e. $|\{i \in S : e \in P_i\}| \leq u_e$. The profit from each driver is defined in the same way as in Highway, however now we obtain profit only from the selected drivers S. Observe that there might be drivers that can afford to pay for the considered prices and are still excluded (i.e., they cannot take the highway) due to capacity constraints. Capacity constraints make sense in some of the mentioned applications, e.g., optimal networks might have insufficient bandwidth to accommodate all candidate users and the authority handling the network could exclude some of these users (regardless of their budget). The same argument applies to a company selling a limited resource, such as computational power, over time slots. The best known approximation for Ls-Highway is, to the best of our knowledge, a folklore $O(\log m)$ approximation based on a reduction to the related *Unsplittable Flow on Path* problem (UFP). Details about this reduction are given later.

In this paper we also consider a *non-uniform demand* generalization of Ls-Highway, next denoted as NuLs-Highway, where each driver i has a demand $d_i \in \mathbb{N}$. W.l.o.g., we can assume that $d_i \leq \min_{e\in P_i}\{u_e\}$ (otherwise driver i can be discarded). Now the subset S of selected drivers has to satisfy $\sum_{i\in S: e\in P_i} d_i \leq u_e$ for each edge e. In particular, Ls-Highway is the special case of NuLs-Highway where $d_i = 1$ for all i. Essentially the same reduction to UFP as mentioned above provides a $O(\log m)$ approximation also for NuLs-Highway.

1.1 Our Results and Technique

The main result of this paper is a PTAS for Ls-Highway (see Section 2).

▶ **Theorem 1.** *There is a deterministic PTAS for Ls-Highway.*

Our starting point is a hierarchical decomposition of G into subpaths (called *intervals*) of different levels as introduced in [25]. The whole path G forms the (only) interval of level 0. Then G is subdivided into $\Gamma = O_\epsilon(1)$ subintervals of level 1 such that for each subinterval the sum of the prices of the edges in the optimal solution is identical. Note that this decomposition depends on the unknown optimal solution and cannot be inferred directly from the input. Recursively, each interval of level ℓ is subdivided into Γ subintervals of level $\ell+1$ with the latter property. A driver is said to be in level ℓ if its path is contained in an interval of level ℓ but not in an interval of level $\ell+1$. The PTAS in [25] guesses this decomposition recursively. First, it guesses the partition of G into intervals of level 1. This implies which drivers are of level 0 and – using some additional arguments – for which of them the total price of the edges of their respective paths exceeds the budget. In more detail, using some shifting arguments one ensures that essentially each driver of level 0 crosses at least $1/\epsilon$ intervals of level 1 completely (and at most two such intervals partially). Since all intervals of level 1 have the same total price in the optimal solution, up to a factor of $1+\epsilon$ this implies the amount that each driver of level 0 would have to pay if it is contained the optimal set of drivers. Then *all* drivers of level 0 are selected whose budget is not exceeded. Afterwards, the algorithm continues recursively in the intervals of level 1. Importantly, in order to process an interval of a level ℓ, one does not need to know which drivers of smaller levels were selected previously. Instead, each arising subproblem can be described by an interval and a level. Therefore, the number of possible subproblems is bounded by a polynomial and the whole algorithm can easily be embedded into a polynomial time dynamic program.

In Ls-Highway this sitution is drastically different. When we want to process an interval of level ℓ then it is not clear that we want to select all drivers of level ℓ whose budget is not exceeded since we might want to use the available edge capacity for drivers of larger levels instead. Also, we need to know the previously selected drivers of smaller levels since they might use capacity on the edges that we then cannot use for drivers of level ℓ anymore. Unfortunately, there is an exponential number of possibilities for which drivers have been selected before and hence we would get a super-polynomial number of possible subproblems. One could use the profiling technique in [5] in order to ensure that there are only a polynomial number of possibilities for the capacity taken by drivers from *each* previous level (with a small loss in the profit). However, since the number of levels is $\Omega(\log n)$ this yields a quasi-polynomial number of combinations for the used capacity from all levels together which is still too much.

At this point our main idea comes into play. We would like that the path of each driver of each level ℓ' starts and ends at the boundary vertex of an interval of level $\ell' + 1$. Then, when we process an interval of level ℓ it would be easy to describe the total capacity taken away from drivers of smaller levels: the total number of such drivers would suffice, knowing that each of them spans the entire interval. Therefore, consider the drivers of level ℓ that start or end in the middle of an interval G' of level $\ell + 1$, let us say there are m' such drivers. For each of them we try to delete a minimal set of drivers of level $\ell + 1$ or larger such that each edge of G' is used by at least one deleted driver. If we succeed then this frees up one unit of capacity along each edge of G' for each considered driver of level ℓ, and we can use this extra space to *forget* the actual portion of G' that is spanned by this driver. In other terms, we can imagine that its path spans the entire subinterval G'. If we do not succeed to delete enough drivers using some edge e then we allocate all remaining capacity on e to the drivers of level ℓ. In other words, the remaining capacity on each edge of G' is reduced *by* m' or *to zero*. Hence, when we process an interval G' of level $\ell + 1$ in our recursion then one number in $\{0, ..., m\}$ suffices to describe by how much the capacity on each edge in G' is reduced due to drivers from smaller levels. We perform this deletion procedure for each level and each interval. This allows us then to devise a polynomial time dynamic program that computes a solution whose profit is at least as large as the profit of the remaining drivers.

To bound the cost of the above deletion step, by losing a factor $1+\epsilon$ in the approximation ratio the construction in [25] ensures that the path of each driver i of a level ℓ spans at least $\Omega(1/\epsilon)$ intervals of level $\ell + 1$ and hence its profit is by a factor $\Omega(1/\epsilon)$ larger than the sum of the edge prices in an interval of level $\ell + 1$. Up to constant factors, the latter is the total profit of the drivers of level at least $\ell + 1$ that we delete for i in the procedure above. Therefore, the total profit due to the deleted tasks can be charged to i, losing only a factor of $1 + O(\epsilon)$.

The non-uniform demand case

Given the above PTAS, it is natural to address the non-uniform version of the problem. In particular:

▷ Question 2. Is there a PTAS for NuLs-Highway?

Using the hierarchical decomposition from above and ideas from [5] it is not hard to derive a QPTAS for the problem, i.e., a $(1+\epsilon)$-approximation that runs in quasi-polynomial time (at least for quasi-polynomially bounded capacities). Let $U_{max} = \max_{e \in G}\{u_e\}$ be the largest capacity.

▶ **Theorem 3.** *For any constant $\epsilon > 0$, there is a deterministic algorithm that computes a $(1+\epsilon)$-approximation for NuLs-Highway in time $(n \log U_{\max})^{O_\epsilon(\log U_{\max} \log m)}$.*

Also, using a folklore reduction to UFP one can obtain a $O(\log m)$-approximation for NuLs-Highway.

▶ **Lemma 4.** *There is a polynomial-time deterministic $O(\log m)$-approximation for NuLs-Highway.*

We were able to design a PTAS for the interesting special case of uniform budgets (see Section 3). Suppose that each driver has a budget of B. We partition G into blocks of total price B/ϵ each and ensure via a shifting argument that the path of essentially each driver is contained in some block. We guess this partition via a dynamic program step by step. For each block, on a high level we show that there is a near-optimal solution in which only $O_\epsilon(1)$ edges within the block have a non-zero price and hence we can guess these edges and their prices in polynomial time. The problem of selecting the drivers yields an instance of UFP in which each task uses one of the latter $O_\epsilon(1)$ edges. We invoke the known PTAS [23] for this case and obtain a $(1+\epsilon)$-approximation overall.

▶ **Theorem 5.** *There is a deterministic PTAS for NuLs-Highway in the special case that the budgets of all drivers are identical.*

1.2 Other Related Work

The *tollbooth* problem is the generalization of the highway problem where G is a tree. A $O(\log n)$ approximation was developed in [20], and later improved to $O(\log n/\log\log n)$ in [22]. Cygan et al. [18] present a $O(\log\log n)$ approximation for the case of uniform budgets. The tollbooth problem is APX-hard [26].

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 price 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 [26]. This bound was refined in [9] to $O(\log L + \log B)$, where L denotes the maximum number of items in a bundle and B the maximum number of bundles containing a given item. Chalermsook et al. [12] showed that this problem is hard to approximate within $\log^{1-\epsilon} n$ for any constant $\epsilon > 0$. A $O(L)$ approximation is given in [3]. The latter approximation factor is asymptotically the best possible for constant values of L unless P = NP as recently proved by Chalermsook et al. [13].

Elbassioni et al. [19] studied the limited-supply highway and tollbooth problems, however for *non-single-minded* drivers. Limited-supply pricing problems have also been studied in their *envy-free* version [26]: the goal here is to compute a maximum-profit pricing so that each client that can afford her bundle actually gets it. Cheung and Swamy [17] provided a $O(\log U)$ approximation for envy-free limited-supply highway with uniform capacities U.

Observe that our algorithm does not guarantee envy-freeness. We also remark that requiring envy-freeness can substantially decrease the optimal profit, hence studying limited-supply pricing problems without this additional constraint makes sense in many applications.

The NuLs-Highway problem has several aspects in common with the well-studied *Unsplittable Flow on a Path* problem (UFP). In this problem we are given a path graph $G = (V, E)$, with edge capacities $\{u_e\}_{e \in E}$, and a set of tasks T, where each task i is characterized by a demand d_i, a subpath P_i of G, and a weight w_i. The goal is to select a maximum weight subset S of tasks such that the total demand $\sum_{i \in S: e \in P_i} d_i$ of selected tasks using each edge e is at most u_e. The current best approximation for this problem is $5/3 + \epsilon$ [24], improving on earlier results [2, 10, 6, 27, 15, 11, 8, 4]. The problem also admits a QPTAS [5, 7]. There is also a line of research on finding LP relaxations with small integrality gap for UFP [1, 11, 14].

1.3 Preliminaries

For any positive integer q let $[q] := \{1, 2, \ldots, q\}$. We are given an $\epsilon > 0$ and assume w.l.o.g. that $1/(2\epsilon)$ is integral and $\epsilon \leq 1/2$. Let (OPT, p^*) denote an optimum solution to the considered instance, with drivers OPT and prices p^*, and opt be its profit. W.l.o.g., OPT contains only drivers with strictly positive profit. Standard reductions (see e.g. [25]) imply the following.

▶ **Lemma 6.** *By losing a factor $1+\epsilon$ in the approximation, we can reduce in polynomial time a given instance of Ls-Highway to an instance of the same problem with $O(m^2/\epsilon)$ edges such that: (1) Budgets are integers between 1 and $\frac{m}{\epsilon}$; (2) Optimal prices take values in $\{0, 1\}$.*

Given the above reduction, we can assume that the sum P^* of the optimal prices is known by trying all the $O(m^2/\epsilon)$ possibilities.

For each edge e let $D_e := \{i \in D : e \in P_i\}$ be the drivers whose path contains e, and, for a subpath G', $D(G') := \{i \in D : P_i \subseteq G'\}$ be the drivers whose path is contained in G'. Given prices p and a subpath G', we let $p(G') = \sum_{e \in G'} p_e$. Given a driver i and prices p, we let the associated profit $\mathrm{pro}(i, p)$ be $p(P_i)$ if this quantity is at most B_i, and 0 otherwise. For a subset of drivers S, $\mathrm{pro}(S, p) = \sum_{i \in S} \mathrm{pro}(i, p)$ is the total profit of those drivers. In case of non-uniform demands, we define $d(S') := \sum_{i \in S'} d_i$.

2 A PTAS for Ls-Highway

In this section we present our PTAS for Ls-Highway.

2.1 Hierarchical decomposition

Consider the input instance after applying the preprocessing step from Lemma 6, with optimal solution (OPT, p^*). We next describe how to extract an almost optimal solution $\mathrm{OPT}' \subseteq \mathrm{OPT}$ with a convenient structure. Here we use the same construction as in [25].

Let $\Gamma = (1/\epsilon)^{1/\epsilon}$ and $\gamma = 1/(2\epsilon)$. We add dummy edges on the right of G (w.l.o.g. having a price of 1 each in the optimal solution) such that we can assume that $P^* = \Gamma^{\ell^*}$ for some integer $\ell^* = O_\epsilon(\log m)$. Since $n \leq \frac{m^2}{\epsilon}$ we can guess in polynomial time the number of dummy edges that we need and the resulting value of P^*. Let $x \in \{1, \ldots, P^*\}$ and $y \in \{1, \ldots, 1/\epsilon\}$ be two parameters to be fixed later. We append $P^* \cdot ((1/\epsilon)^y - 1) - x$ additional dummy edges to the left of G and x additional dummy edges to the right of G, resp., and we assume w.l.o.g. that p^* assigns a price of 1 to each one of them. To simplify the notation, we denote by G the resulting path, by p^* the resulting prizing, and by P^* the sum of prices in p^*. Observe that now $P^* = \Gamma^{\ell^*}(1/\epsilon)^y$.

Based on p^*, we define a hierarchical decomposition of G into nested subpaths (*invervals*). The starting point is the interval G of level 0. Given an interval G' of level ℓ, we partition it into Γ subintervals $G'_1, \ldots, G'_\Gamma$ of level $\ell+1$, with uniform price. Observe that intervals of level ℓ have price $P^\ell := P^*/\Gamma^\ell = \Gamma^{\ell^*-\ell}(1/\epsilon)^y$. We stop the recursion at intervals of level ℓ^*, which have constant price $(1/\epsilon)^y = O_\epsilon(1)$.

For each interval G' we denote by $\ell(G')$ its level. We say that a driver i is at level ℓ if P_i is fully contained in an interval of level ℓ but in no interval of level $\ell+1$. For each driver i, we let $\ell(i)$ be its level and $q(i)$ be the number of intervals of level $\ell(i)+1$ which are fully contained in P_i. Based on the above decomposition and notation, we define an approximate profit function pro* for each driver i of level $\ell(i) < \ell^*$ as follows

$$\mathrm{pro}^*(i) = \begin{cases} 0 & \text{if } q(i) < \gamma \text{ or } q(i) \cdot P^{\ell(i)+1} > B_i \\ q(i) \cdot P^{\ell(i)+1} & \text{otherwise.} \end{cases} \tag{1}$$

For drivers i of level ℓ^*, we use the standard definition of profit, i.e. $\mathrm{pro}^*(i) = p^*(P_i)$ for $p^*(P_i) \leq B_i$, and $\mathrm{pro}^*(i) = 0$ otherwise. Intuitively, pro* counts the profit of a driver i in level ℓ only if P_i spans many subintervals of level $\ell+1$, i.e., at least γ many. For counting the profit, we ignore the two subintervals that P_i only partially overlaps with. Since P_i gets the full profit of at least γ subintervals, the difference is only a factor of $1 + O(\epsilon)$. On the other hand, it could be that $\mathrm{pro}^*(i) > 0$ but i's budget is exceeded. In this case it is still true that $\mathrm{pro}(i, p^*) \geq \mathrm{pro}^*(i)$ if i had a budget of $\frac{\gamma+2}{\gamma}B_i \leq (1+O(\epsilon))B_i$. Therefore, intuitively we will pretend in the sequel that all drivers have a (larger) budget of $\frac{\gamma+2}{\gamma}B_i$ and repair this by scaling all edge prices at the very end. As usual, for $S \subseteq D$, $\mathrm{pro}^*(S) = \sum_{i\in S} \mathrm{pro}^*(i)$. We next let $\mathrm{OPT}' \subseteq \mathrm{OPT}$ be the drivers $i \in \mathrm{OPT}$ with strictly positive $\mathrm{pro}^*(i)$ (hence of profit at least $\gamma P^{\ell(i)+1}$).

▶ **Lemma 7** ([25]). *There exist values of x and y such that* $\mathrm{pro}^*(\mathrm{OPT}) = \mathrm{pro}^*(\mathrm{OPT}') \geq (1 - O(\epsilon))\,opt$.

In the following we assume that the input graph is preprocessed according to the pair (x, y) given by Lemma 7: this is w.l.o.g. since we can try all the constantly many options. By pro* we will denote the approximate profit function given by this choice.

2.2 A Structured Solution

At this point we introduce the most critical and novel idea in our PTAS. We extract from OPT' a large profit subset OPT'' that is even more structured. Intuitively, our goal is to limit the interaction between drivers of different levels. More formally, in OPT'' for each interval G' of some level ℓ' there exists a value m' such that on each edge e of G' the drivers of level ℓ' or larger use at most $\max\{0, u_e - m'\}$ units of capacity and the drivers of levels $\ell' - 1$ or smaller use the remaining capacity. Our algorithm will later select the drivers in the order of their levels, from small to large. Hence, in order to describe the capacity available on G' for the drivers level ℓ' or larger it suffices to know m' for which there are only $m+1$ possibilities. This will be useful to define our algorithm as a polynomial time dynamic program.

Initially we set $\mathrm{OPT}'' = \mathrm{OPT}'$. Then we gradually move some drivers from OPT'' to a set of *deleted* drivers DEL. We will guarantee that the profit of deleted drivers is a small fraction of the profit of drivers that are still in OPT'' at the end of the process.

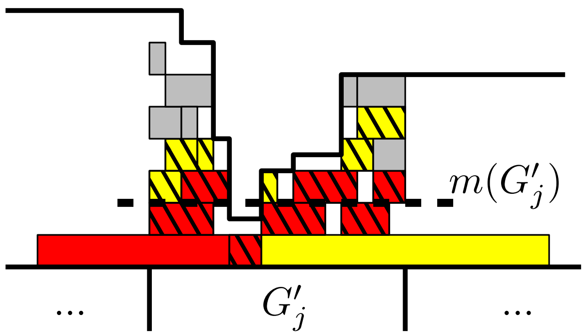

Figure 1 For the red driver we delete a set of short drivers whose paths are contained in G'_j, depicted in striped red, that together completely cover G'_j. For the yellow driver we do not find such a set among the remaining drivers and delete drivers (depicted in striped yellow) that cover a maximal set of edges in G'_j. The gray drivers remain in the solution.

It is convenient to describe the construction of DEL in terms of a recursive procedure *delete*. This procedure, described in Figure 2, takes as input a tuple (G', ℓ', m') where G' is a subpath, $\ell' \in \{0, \dots, \ell^*\}$ is a level, and $m' \in \{0, \dots, m\}$ is some capacity. Note that w.l.o.g. we can assume that $u_e \leq m$ for each edge e. Furthermore, OPT$''$ and DEL are considered as global variables. We initialize $(\mathrm{OPT}'', \mathrm{DEL})$ to $(\mathrm{OPT}', \emptyset)$, and run $delete(G, 0, 0)$.

The high-level idea behind *delete* is as follows. Intuitively, G' is some interval of level ℓ', and m' is some uniform capacity that is *reserved* along G' to allocate drivers from previous levels whose path overlaps with G'. We remark that m' might exceed the capacity u_e available on some edge $e \in G'$, in which case drivers from level ℓ' or larger cannot use edge e (in other words, the *residual capacity* on edge e is $\max\{0, u_e - m'\}$). Consider the subdivision of G' into subintervals $G'_1, \dots, G'_\Gamma$. Let us focus on a specific G'_j, and consider the drivers of level ℓ' in $\mathrm{OPT}'' \cap D(G')$ whose path intersects G_j, let us denote them by $\mathrm{OPT}''_{\ell'}(G'_j)$. Let $\mathrm{OPT}''_{\ell',part}(G'_j)$ and $\mathrm{OPT}''_{\ell',span}(G'_j)$ be the subset of them with $G'_j \not\subseteq P_i$ and $G'_j \subseteq P_i$, resp. In order to define the residual capacity for drivers in $D(G'_j)$ the drivers in $\mathrm{OPT}''_{\ell',span}(G'_j)$ are not problematic: they use a uniform amount of capacity along G'_j. In order to handle the problematic drivers $\mathrm{OPT}''_{\ell',part}(G'_j)$, the procedure *delete* removes some drivers from $\mathrm{OPT}'' \cap D(G'_j)$ of level $\ell' + 1$ or larger. This leaves some free capacity that can be used to *ignore* the exact extend by which each $i \in \mathrm{OPT}''_{\ell',part}(G'_j)$ overlaps with G'_j. Ideally, for each $i \in \mathrm{OPT}''_{\ell',part}(G'_j)$, we would like to find a minimal set of drivers $\mathrm{DEL}_i(G'_j) \subseteq \mathrm{OPT}'' \cap D(G'_j)$ that spans G'_j, i.e., such that each edge of G'_j is used by at least one driver in $\mathrm{DEL}_i(G'_j)$. However, there might not be enough drivers available for this in which case we rather take one such set with the largest possible span of edges of G'_j. This process is illustrated in Figure 1. After deleting all drivers in the sets $\mathrm{DEL}_i(G'_j)$ for all $i \in \mathrm{OPT}''_{\ell',part}(G'_j)$ we can safely set the (residual) capacity on edge e for drivers of level larger than ℓ' (whose path is contained in G'_j) to $\max\{0, u_e - m' - m_{\ell'}(G'_j)\}$ where $m_{\ell'}(G'_j) := |\mathrm{OPT}''_{\ell'}(G'_j)|$. We then recurse in each subinterval G'_j of G' by calling $delete(G'_j, \ell' + 1, m' + m_{\ell'}(G'_j))$. We stop the recursion once we reach an interval of level ℓ^* in which case we do not delete any further drivers.

Consider OPT$''$ at the end of the root call $delete(G, 0, 0)$. This is obviously a feasible solution (being a subset of OPT$'$). Let us show that it has large profit.

▶ **Lemma 8.** *We have that* $\mathrm{pro}^*(\mathrm{OPT}'') \geq (1 - O(\epsilon))\mathrm{pro}^*(\mathrm{OPT}')$

Proof. Let us show that $\mathrm{pro}^*(\mathrm{DEL}) \leq \frac{4}{\gamma}\mathrm{pro}^*(\mathrm{OPT}'') = O(\epsilon)\mathrm{pro}^*(\mathrm{OPT}'')$. We use a charging

$delete(G', \ell', m')$

if $\ell' = \ell^*$ **then**
  halt;
Let $G'_1, \ldots, G'_\Gamma$ be the partition of G' into subintervals of level $\ell + 1$;
for $j = 1, \ldots, \Gamma$ **do**
  Let $\mathrm{OPT}''_{\ell'}(G'_j) := \{i \in \mathrm{OPT}'' \cap D(G') : \ell(i) = \ell', E(P_i) \cap E(G'_j) \neq \emptyset\}$;
  Let $m_{\ell'}(G'_j) = |\mathrm{OPT}''_{\ell'}(G'_j)|$;
  Let $\mathrm{OPT}''_{\ell',part}(G'_j) := \{i \in \mathrm{OPT}''_{\ell'}(G'_j) : G'_j \not\subseteq P_i\}$;
  for every $i \in \mathrm{OPT}''_{\ell',part}(G'_j)$ **do**
    Let $E_i(G'_j)$ be the edges used by $\mathrm{OPT}'' \cap D(G'_j)$;
    Let $\mathrm{DEL}_i(G'_j)$ be a minimal subset of $\mathrm{OPT}'' \cap D(G'_j)$ that spans $E_i(G'_j)$;
    Set $\mathrm{OPT}'' \leftarrow \mathrm{OPT}'' \setminus \mathrm{DEL}_i(G'_j)$ and $\mathrm{DEL} \leftarrow \mathrm{DEL} \cup \mathrm{DEL}_i(G'_j)$;
  $delete(G'_j, \ell' + 1, m' + m_{\ell'}(G'_j))$;

Figure 2 Procedure to build the sets OPT'' and DEL.

argument. Consider an interval G' of level ℓ' and one of its subintervals G'_j. Note that, by construction, the total price over G' and G'_j is $P^{\ell'}$ and $P^{\ell'+1} = P^{\ell'}/\Gamma$, respectively. Consider any $i \in \mathrm{OPT}''_{\ell',part}(G'_j)$. Observe that i cannot be deleted in the next recursive calls, hence it finally belongs to OPT''. Let us charge the loss due to the removal of $\mathrm{DEL}_i(G'_j)$ to i.

By the minimality of $\mathrm{DEL}_i(G'_j)$, each edge $e \in G'_j$ can be used by at most two drivers in $\mathrm{DEL}_i(G'_j)$. It thus follows that $\mathrm{pro}^*(\mathrm{DEL}_i(G'_j)) \leq 2p^*(G'_j) \leq 2P^{\ell'+1}$. On the other hand, $\mathrm{pro}^*(i) \geq \gamma P^{\ell'+1}$, hence $\mathrm{pro}^*(\mathrm{DEL}_i(G'_j)) \leq \frac{2}{\gamma}\mathrm{pro}^*(i)$. Observe that each driver i in OPT'' of level ℓ' can be charged by at most two sets $\mathrm{DEL}_i(G'_a)$ and $\mathrm{DEL}_i(G'_b)$, associated with the (at most) two subintervals G'_a and G'_b of level $\ell' + 1$ that partially overlap with P_i (since the subintervals that are fully spanned by P_i do not charge i). It follows that

$$\mathrm{pro}^*(\mathrm{DEL}) = \sum_{G'_j, i} \mathrm{pro}^*(\mathrm{DEL}_i(G'_j)) \leq \frac{2}{\gamma} \sum_{G'_j, \ell', i \in \mathrm{OPT}''_{\ell',part}(G'_j)} \mathrm{pro}^*(i) \leq \frac{4}{\gamma}\mathrm{pro}^*(\mathrm{OPT}'').$$

◀

One can show that, if in the recursion above a call $delete(G', \ell', m')$ arises, then in OPT'' on each edge e of G' the drivers of level ℓ' or larger use at most $\max\{0, u_e - m'\}$ units of capacity. We will use this property in our dynamic program below.

2.3 Dynamic program

We describe an algorithm that computes a solution with a profit of at least $\mathrm{pro}^*(OPT'')$, pretending that each driver i has an increased budget of $\frac{\gamma+2}{\gamma}B_i$. Afterwards, we scale down the prices by a factor $\frac{\gamma+2}{\gamma}$ in order to respect the original budgets. Together with Lemma 8 this yields an approximation factor of $1 + O(\epsilon)$. For the sake of simplicity, in the sequel we will compute only the value of the desired solution while a straightforward extension yields an algorithm that finds the corresponding set of drivers and also the pricing for the edges.

A natural idea is to define a recursive algorithm that *guesses* the hierarchical decomposition into intervals and the values $m(G'_j)$ corresponding to OPT''. Suppose we are given a tuple (G', ℓ', m') consisting of an interval G', a level ℓ', and an integer m'. The reader may imagine that in the hierarchical decomposition above G' is of level ℓ' and m' units of capacity are taken away on each edge of G' due to drivers of levels smaller than ℓ'. If $\ell' < \ell^*$ we guess the corresponding subdivision into subintervals $G'_1, \ldots, G'_\Gamma$ of level $\ell' + 1$ (each of them having length at least $P^{\ell'+1}$), and the associated values $m(G'_j)$, i.e., we try all possibilities for them. Via a reduction to UFP we select the drivers of level ℓ': for each driver i with $P_i \subseteq G'$ but $P_i \not\subseteq G'_j$ for each G'_j, we introduce a task i' with path $P_{i'} := P_i$ and demand $d_{i'} := 1$. For $q(i)$ being the number of intervals G'_j with $G'_j \subseteq P_i$ we define

$$w_{i'} := \begin{cases} 0 & \text{if } q(i) < \gamma \text{ or } q(i) \cdot P^{\ell'+1} > B_i \\ q(i) \cdot P^{\ell'+1} & \text{otherwise.} \end{cases} \tag{2}$$

Observe that to guarantee that we get a profit of $w_{i'}$ from driver i we would need that i has a budget of at least $\frac{\gamma+2}{\gamma} B_i$. This can be fixed at the end by scaling down prices by a factor $\frac{\gamma+2}{\gamma}$ (with a small profit loss). We define the edge capacities by $u'_e := \min\{m(G'_j), \max\{0, u_e - m'\}\}$ for each edge e in a subinterval G'_j. Since all drivers have unit demand this instance of UFP can be solved exactly in polynomial time (see, e.g., [16]). Then we recurse on each interval G'_j such that the corresponding subproblem consists of the tuple $(G'_j, \ell'+1, m' + m(G'_j))$. Finally, the solution for (G', ℓ', m') is the most profitable solution obtained in this way over all of the guesses above.

If we are given a tuple (G', ℓ', m') with $\ell' = \ell^*$ (the reader may again imagine that G' is an interval of level ℓ^*) then we guess directly the optimal pricing p^* which is one of the polynomially many options to assign a total price of $(1/\epsilon)^y = O_\epsilon(1)$ to the edges of G' such that each edge gets a price in $\{0, 1\}$. Selecting the drivers yields again an instance of UFP. For each driver i with $P_i \subseteq G'$ we introduce a task i' with path $P_{i'} := P_i$, demand $d_{i'} := 1$, and weight $w_{i'} = p(B_i)$ if $p(B_i) \leq B_i$ and $w_{i'} = 0$ otherwise. Each edge e has a capacity of $u'_e := \max\{0, u_e - m'\}$. Again, since all drivers have unit demand we can solve this instance of UFP in polynomial time [16]. The solution for (G', ℓ', m') is then the most profitable solution over all guesses. We return the solution to $(G, 0, 0)$.

As it is described above, this algorithm does not have polynomial running time since in each subproblem we enumerate a polynomial number of guesses and the recursion depth is $\Omega(\log n)$. However, each recursive call is specified by a tuple (G', ℓ', m') and there are only a polynomial number of those. Hence, we can embed our algorithm into a polynomial time dynamic program, see Figure 3. For each cell (G', ℓ', m') denote by $DP(G', \ell', m')$ the value stored in it.

Let us consider the recursive partition of G that corresponds to $(G, 0, 0)$, i.e., the intervals of the partition achieving the maximum in Line 7 and recursively their subpartitions achieving the maximum in their respective subproblems. Let $\mathcal{G}$ be the intervals in this partition. For a given $G' \in \mathcal{G}$ we let $\ell'(G')$ and $m'(G')$ denote the associated values of ℓ' and m'. Furthermore, if $\ell'(G') < \ell^*$, let $\{G'_j\}_j$ and $\{m(G'_j)\}_j$ be the corresponding values achieving the maximum in Step 7. By $\mathrm{ALG}(G')$ we denote the UFP solution corresponding to the cell $(G', \ell'(G'), m'(G'))$ that achieves the maximum value in Step 7 or 12. Note that the solution associated with $(G, 0, 0)$ is $\mathrm{ALG} = \cup_{G' \in \mathcal{G}} \mathrm{ALG}(G')$. By p^{ALG} we denote the pricing induced by the values p achieving the maximum in Step 12.

▶ **Lemma 9.** *ALG respects the capacity constraints.*

Proof. For a given $G' \in \mathcal{G}$, let $\mathrm{ALG}^{\downarrow}(G') := \cup_{G'' \in \mathcal{G}: G'' \subseteq G'} \mathrm{ALG}(G'')$ be the union of all the UFP solutions corresponding to subintervals contained in G' (G' included). We will show by induction on decreasing values of $\ell'(G')$ that $\mathrm{ALG}^{\downarrow}(G')$ is a feasible solution w.r.t. residual capacities $\max\{0, u_e - m'(G')\}$ i.e.

$$|D_e \cap \mathrm{ALG}^{\downarrow}(G')| \leq \max\{0, u_e - m'(G')\}, \quad \forall e \in G'.$$

The claim then follows since $m'(G) = 0$ and $\mathrm{ALG}^{\downarrow}(G) = \mathrm{ALG}$.

compute $DP(G', \ell', m')$

1: **if** $\ell' < \ell^*$ **then**
2: **for** all possible subdivisions of G' into subpaths $G'_1, \dots, G'_\Gamma$ of length at least $P^{\ell'+1}$ each **do**
3: **for** all possible values $m(G'_j) \in \{0, \dots, m\}$, $j = 1, \dots, \Gamma$ **do**
4: construct the UFP instance I' associated with $(G', m', \{G'_j\}_j, \{m(G'_j)\}_j)$;
5: solve I' optimally, let $wufp(I')$ be the resulting profit
6: compute
$$w(G', m', \{G'_j\}_j, \{m(G'_j)\}_j) := wufp(I') + \sum_{j=1}^{\Gamma} DP(G'_j, \ell' + 1, m' + m(G'_j))$$
7: $DP(G', \ell', m') \leftarrow$ largest value $w(G', m', \{G'_j\}_j, \{m(G'_j)\}_j)$ computed in Step 6
8: **if** $\ell' = \ell^*$ **then**
9: **for** all possible assignments $p = \{0, 1\}^{E(G')}$ with $p(G') = P^{\ell^*} = (1/\epsilon)^y$ **do**
10: construct the UFP instance I' associated with (G', m', p);
11: solve I' optimally, let $wufp(I')$ be the resulting profit
12: $DP(G', \ell', m') \leftarrow$ largest value $wufp(I')$ computed in Step 11

Figure 3 Dynamic program to approximate Ls-Highway. Here G' denote a subpath of G of length at least $P^{\ell'}$, $\ell' \in \{0, \dots, \ell^*\}$ a level, and $m' \in \{0, \dots, m\}$ a capacity.

For the base case $\ell'(G') = \ell^*$ this is true by the definition of the edges capacities for the case that $\ell' = \ell^*$. Suppose next the claim is true up to the value $\ell' + 1$, and consider G' with $\ell'(G') = \ell'$. Consider any edge $e \in G'_j$, for some $j \in [\Gamma]$.

$$|D_e \cap \text{ALG}^{\downarrow}(G'_j)| \le \max\{0, u_e - m'(G') - m(G'_j)\}.$$

By construction and the definition of the edge capacities for the case that $\ell' < \ell^*$ we have

$$|D_e \cap \text{ALG}(G')| \le \min\{m(G'_j), \max\{0, u_e - m'(G')\}\}.$$

Thus

$$\begin{aligned} &|D_e \cap \text{ALG}^{\downarrow}(G')| = |D_e \cap \text{ALG}(G')| + |D_e \cap \text{ALG}^{\downarrow}(G'_j)| \\ \le &\min\{m(G'_j), \max\{0, u_e - m'(G')\}\} + \max\{0, u_e - m'(G') - m(G'_j)\} \\ \le &\max\{0, u_e - m'(G')\}, \end{aligned}$$

where the last inequality follows easily by distinguishing the cases $u_e - m'(G') \le 0$, $0 < u_e - m'(G') \le m(G'_j)$, and $u_e - m'(G') > m(G'_j)$. ◀

The proof of the following lemma follows by constraining the choices of the algorithm in order to mimic the construction of OPT$''$. The crucial step is to show that, for a subproblem (G', ℓ', m') (corresponding to an interval G' in the hierarchical decomposition due to Section 2.1) the drivers in OPT$'' \cap D(G')$ of level ℓ' (denote them by OPT$''(G', \ell')$) define a feasible solution for the associated UFP instance whose weight is precisely pro*(OPT$''(G', \ell)$) by the definition of the tasks weights of these instances.

▶ **Lemma 10.** $DP(G, 0, 0) = \text{pro}^*(ALG) \ge \text{pro}^*(\text{OPT}'')$.

Finally, we scale down the price on each edge by a factor $\frac{\gamma+2}{\gamma} \le 1 + O(\epsilon)$, i.e. we return the solution (ALG, $\frac{\gamma}{\gamma+2} p^{\text{ALG}}$). This way, all drivers in ALG respect the original budgets and we achieve a profit almost as large as $DP(G, 0, 0)$.

▶ **Lemma 11.** $\text{pro}(ALG, \frac{\gamma}{\gamma+2} p^{ALG}) \ge \frac{\gamma}{\gamma+2} DP(G, 0, 0)$.

Proof. It is sufficient to show that, after scaling prices, the budget of each driver $i \in \mathrm{ALG}$ is not exceeded. It then follows that the profit associated with i is precisely $\frac{\gamma}{\gamma+2}$ times its weight $w_{i'}$ in the corresponding UFP instance. W.l.o.g. we can assume that $w_{i'} > 0$. If the level ℓ of i is ℓ^*, the claim is trivial since $\mathrm{pro}(i, p^{\mathrm{ALG}}) = w_{i'}$. In other words, the budget of i is respected even without scaling the prices. Otherwise, with the usual notation, by definition we have that $q(i) \geq \gamma$ and $q(i) \cdot P^\ell \leq B_i$. By construction the total price associated with i is

$$p^{\mathrm{ALG}}(P_i) \leq (q(i) + 2)P^\ell \leq \frac{\gamma+2}{\gamma} q(i) P^\ell \leq \frac{\gamma+2}{\gamma} B_i.$$

Hence $\frac{\gamma}{\gamma+2} p^{\mathrm{ALG}}$ does not violate the budget of i as required. ◀

Our algorithm runs in polynomial time since we have a polynomial number of DP-cells and the computation for each takes polynomial time. Now the proof of Theorem 1 follows immediately from Lemmas 8-11.

3 A PTAS for NuLs-Highway with Uniform Budgets

In this section we present a PTAS for NuLs-Highway when all drivers have the same budget B. It is not hard (modulo technicalities) to extend our result to the case that the ratio of largest to smallest budget is upper bounded by a given constant.

We will use the following folklore result for the highway problem (and more generally for item pricing problems), that immediately extends to Ls-Highway and NuLs-Highway.

▶ **Lemma 12. (Close To Budget Lemma)** *Given any $\alpha \in (0, 1]$, in any optimal solution (OPT, p^*) to NuLs-Highway at least a fraction $(1 - \alpha)$ of the profit is due to drivers whose profit is at least α times their budget.*

Proof. Assume by contradiction the claim is not true, and consider the drivers $i \in S \subseteq \mathrm{OPT}$ whose profit in p^* is less than $\alpha \cdot B_i$. Then the pricing p^*/α achieves a profit larger than OPT from S, a contradiction. ◀

Let us first show that a solution with a convenient structure exists. Let (OPT, p^*) be an optimal solution. Using Lemma 6 we can assume that the price of each edge is in $\{0, 1\}$ and that $B \in \{1, ..., m/\epsilon\}$. Let P^* be the sum of the optimal prices, and h^* be the smallest integer such that $P^* \leq (h^* - 1)\frac{B}{\epsilon}$. We guess P^* and hence we then also know h^*. For a choice of $x \in \{0, \ldots, \frac{1}{\epsilon}B - 1\}$ to be defined later, we append x edges to the left of the input graph G, and $y = (h^* - 1)\frac{B}{\epsilon} - P^* + \frac{1}{\epsilon}B - x$ edges to its right. W.l.o.g. we assume that each new edge has a price of 1 in (OPT, p^*). For simplicity, we still denote by G the resulting graph, by p^* its optimal pricing, by P^* the total price of all edges and we define $h^* := P^*\epsilon/B$. Observe that $p^*(G) = h^*\frac{B}{\epsilon}$.

By $\mathrm{OPT}' \subseteq \mathrm{OPT}$ we denote the drivers i whose profit in p^* is at least $\epsilon \cdot B_i$. By applying Lemma 12 with $\alpha = \epsilon$ one has that $\mathrm{pro}(\mathrm{OPT}', p^*) \geq (1 - \epsilon)\mathrm{opt}$. We next define a solution (APX, p^{apx}), with $\mathrm{APX} \subseteq \mathrm{OPT}'$. Subdivide G in h^* subpaths B_j (*blocks*) with total price exactly $\frac{B}{\epsilon}$ each. Discard all drivers $i \in \mathrm{OPT}'$ whose path P_i contains edges in two different blocks: let APX be the remaining drivers. Subdivide each B_j into $1/\epsilon^3$ subpaths $B_{j,k}$ of optimal price exactly $\epsilon^2 B$ each (*sub-blocks*). In p^{apx} set the price of the rightmost edge in each sub-block to $\frac{1}{1+\epsilon} \cdot \epsilon^2 B$ and the price of any other edge to zero (note that we use fractional prices even though we assumed the optimal solution to have prices in $\{0, 1\}$).

Let us show that the profit of the new solution is large enough for a proper choice of x.

▶ **Lemma 13.** *There is a choice of x in the above construction such that* $\mathrm{pro}(APX, p^{apx}) \geq (1 - O(\epsilon))\, opt$.

We devise now a dynamic program that computes a solution with a profit of at least $\mathrm{pro}(\mathrm{APX}, p^{apx})$. Intuitively, it guesses step by step the above partition into blocks. Then for each block B it guesses its partition into subblocks, sets a price of $\frac{1}{1+\epsilon} \cdot \epsilon^2 B$ to the rightmost edge of each subblock and computes a subset of drivers from $D(B)$ maximizing the profit from the selected drivers. The problem of selecting these drivers yields special instances of UFP (one for each block) in which there are $1/\epsilon^3$ special edges (the edges with non-zero price) such that each input task uses at least one of them (all other drivers yield zero profit and can be discarded). We invoke the known PTAS for this special case [23].

Formally, first we guess the value for $x \in \{0, \ldots, \frac{1}{\epsilon}B - 1\}$ due to Lemma 12. Observe that $B \leq m/\epsilon$ due to Lemma 5 and hence there are only m/ϵ^2 options for x. We start by preprocessing the instance as described before for the considered x: let N be the final number of edges, and let us label them from 1 to N from left to right. Let $G_{\ell,r}$ be the subpath of G with leftmost edge ℓ and rightmost edge r. The DP table is indexed by pairs (r, h) where r is some edge and $h \in \{1, \ldots, h^*\}$. Intuitively, the value of $DP(r, h)$ is the maximum profit that is achievable by drivers whose path is contained in $G_{1,r}$ in the following way:

1. We divide $G_{1,r}$ into h blocks B_j, subdivide each block into $1/\epsilon^3$ sub-blocks, and assign the price $\frac{\epsilon^2 B}{1+\epsilon}$ to the rightmost edge of each sub-block (and 0 otherwise).
2. We select a set of drivers such that the path of each driver is fully contained in some block.

As usual, we can associate to $DP(r, h)$ a specific solution of the same profit. At the end, we output the solution in the cell $DP(N, h^*)$.

Consider a given DP-cell $DP(r, h)$. For all values ℓ with $1 \leq \ell < r$ we do the following: we partition $G_{1,r}$ into a block $G_{\ell,r}$ and a remaining part $G_{1,\ell-1}$. We consider all the possible $O(n^{1/\epsilon^3})$ ways to subdivide $G_{\ell,r}$ into sub-blocks $B^1_{\ell,r}, \ldots, B^{1/\epsilon^3}_{\ell,r}$ such that none of them is empty. For any such choice, we define a UFP instance $UFP(\{B^k_{\ell,r}\}_k)$ as follows. The graph is $G_{\ell,r}$, with the corresponding edge capacities u_e, $e \in G_{\ell,r}$. For each driver i with $P_i \subseteq G_{\ell,r}$, we define a task i' with path $P_{i'} := P_i$ and demand $d_{i'} := d_i$. We define its weight $w_{i'}$ as follows: assign price $\frac{\epsilon^2 B}{1+\epsilon}$ to the rightmost edge in each sub-block; set $w_{i'}$ to the total price on the edges of $P_{i'}$ if this is at most B_i, and 0 otherwise. We discard a task i' if with $w_{i'} = 0$.

Note that in this instance of UFP each task must use one of the $1/\epsilon^3$ edges with non-zero price. We invoke the PTAS in [23, Theorem 3.3] for this special case. Let $\mathrm{alg}(\ell, r)$ be the maximum weight of any computed UFP solution for this choice of ℓ (over all partitions $B^1_{\ell,r}, \ldots, B^{1/\epsilon^3}_{\ell,r}$). Observe that this value depends only on ℓ and r. If $r - \ell + 1 < 1/\epsilon^3$ then there can be no partition in which all sub-blocks are non-empty and therefore we set $\mathrm{alg}(\ell, r) = 0$. Given the above quantities, we define

$$DP(r, h) := \max_{1 \leq \ell < r} \{\mathrm{alg}(\ell, r) + DP(\ell - 1, h - 1)\}$$

where we define $DP(0, h) = 0$ for all h and $DP(r, 0) = 0$ for all r. Finally, we output the solution in the DP-cell $DP(N, h^*)$.

References

1 Aris Anagnostopoulos, Fabrizio Grandoni, Stefano Leonardi, and Andreas 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 Aris Anagnostopoulos, Fabrizio Grandoni, Stefano Leonardi, and Andreas Wiese. A mazing $2+\varepsilon$ approximation for unsplittable flow on a path. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 26–41, 2014.

3 Maria-Florina Balcan and Avrim Blum. Approximation algorithms and online mechanisms for item pricing. In *ACM Conference on Electronic Commerce*, pages 29–35, 2006.

4 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.

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

6 Amotz Bar-Noy, Reuven Bar-Yehuda, Ari Freund, Joseph Naor, and Baruch Schieber. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM*, 48(5):1069–1090, 2001.

7 Jatin Batra, Naveen Garg, Amit Kumar, Tobias Mömke, and Andreas Wiese. New approximation schemes for unsplittable flow on a path. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 47–58, 2015.

8 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.

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

10 Gruia Calinescu, Amit Chakrabarti, Howard J. Karloff, and Yuval Rabani. Improved approximation algorithms for resource allocation. In *International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 401–414, 2002.

11 Amit Chakrabarti, Chandra Chekuri, Anupam Gupta, and Amit Kumar. Approximation algorithms for the unsplittable flow problem. *Algorithmica*, 47(1):53–78, 2007.

12 Parinya Chalermsook, Julia Chuzhoy, Sampath Kannan, and Sanjeev 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.

13 Parinya Chalermsook, Bundit Laekhanukit, and Danupon 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.

14 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.

15 Chandra Chekuri, Marcelo Mydlarz, and F. Bruce Shepherd. Multicommodity demand flow in a tree and packing integer programs. *ACM Transactions on Algorithms*, 3(3), 2007.

16 Chandra Chekuri, Marcelo Mydlarz, and F Bruce Shepherd. Multicommodity demand flow in a tree and packing integer programs. *ACM Transactions on Algorithms (TALG)*, 3(3):27, 2007.

17 Maurice Cheung and Chaitanya Swamy. Approximation algorithms for single-minded envy-free profit-maximization problems with limited supply. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 35–44, 2008.

18 Marek Cygan, Fabrizio Grandoni, Stefano Leonardi, Marcin Pilipczuk, and Piotr Sankowski. A path-decomposition theorem with applications to pricing and covering on trees. In *European Symposium on Algorithms (ESA)*, pages 349–360, 2012.

19 Khaled M. Elbassioni, Mahmoud Fouz, and Chaitanya Swamy. Approximation algorithms for non-single-minded profit-maximization problems with limited supply. In *Internet and Network*

Economics - 6th International Workshop, WINE 2010, Stanford, CA, USA, December 13-17, 2010. Proceedings, pages 462–472, 2010.

20 Khaled M. Elbassioni, Rajiv Raman, Saurabh Ray, and René Sitters. On profit-maximizing pricing for the highway and tollbooth problems. In *International Symposium on Algorithmic Game Theory (SAGT)*, pages 275–286, 2009.

21 Khaled M. Elbassioni, René Sitters, and Yan Zhang. A quasi-PTAS for profit-maximizing pricing on line graphs. In *European Symposium on Algorithms (ESA)*, pages 451–462, 2007.

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

23 Fabrizio Grandoni, Tobias Mömke, Andreas Wiese, and Hang Zhou. To augment or not to augment: Solving unsplittable flow on a path by creating slack. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2411–2422, 2017.

24 Fabrizio Grandoni, Tobias Mömke, Andreas Wiese, and Hang Zhou. A $(5/3 + \epsilon)$-approximation for unsplittable flow on a path: placing small tasks into boxes. In *ACM Symposium on Theory of Computing (STOC)*, pages 607–619, 2018.

25 Fabrizio Grandoni and Thomas Rothvoß. Pricing on paths: A PTAS for the highway problem. *SIAM J. Comput.*, 45(2):216–231, 2016.

26 Venkatesan Guruswami, Jason D. Hartline, Anna R. Karlin, David Kempe, Claire Kenyon, and Frank McSherry. On profit-maximizing envy-free pricing. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1164–1173, 2005.

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