Revenue Maximization in Transportation Networks

Kshipra Bhawalkar
Google Research, Mountain View, CA, USA

Kostas Kollias
Google Research, Mountain View, CA, USA

Manish Purohit
Google Research, Mountain View, CA, USA

Abstract

We study the joint optimization problem of pricing trips in a transportation network and serving the induced demands by routing a fleet of available service vehicles to maximize revenue. Our framework encompasses applications that include traditional transportation networks (e.g., airplanes, buses) and their more modern counterparts (e.g., ride-sharing systems). We describe a simple combinatorial model, in which each edge in the network is endowed with a curve that gives the demand for traveling between its endpoints at any given price. We are supplied with a number of vehicles and a time budget to serve the demands induced by the prices that we set, seeking to maximize revenue. We first focus on a (preliminary) special case of our model with unit distances and unit time horizon. We show that this version of the problem can be solved optimally in polynomial time. Switching to the general case of our model, we first present a two-stage approach that separately optimizes for prices and routes, achieving a logarithmic approximation to revenue in the process. Next, using the insights gathered in the first two results, we present a constant factor approximation algorithm that jointly optimizes for prices and routes for the supply vehicles. Finally, we discuss how our algorithms can handle capacitated vehicles, impatient demands, and selfish (wage-maximizing) drivers.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis

Keywords and phrases Pricing, networks, approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2021.26

Category APPROX

1 Introduction

The increasing popularity of ride-sharing systems has inspired renewed interest on questions of pricing and routing transportation requests in networks [2, 4, 6, 12, 16]. Typically, such ride sharing platforms have an abundance of data at their disposal, which offers them a good understanding of the market. These data offer insights which can lead to reasonable estimates of the supply of drivers expected at a given time, as well as the number of customers who would be interested in taking a given trip at a given time and price. Similar data is available for more traditional transportation companies, such as airlines and bus agencies. In all these settings it is natural to ask the question:

How do we maximize revenue in a transportation network, given a) a supply of vehicles and b) demand curves for the possible trips?

This question appears at face value to be (primarily) a pricing problem. While this is true to a large extent, there is a latent scheduling/routing aspect of how one can serve these demands with an available supply of vehicles. This connection implies that any approach in this setting has to address difficulties encountered both in pricing and in routing problems.

Various efforts have been made at tackling aspects of pricing and routing in ride-sharing platforms. These include queueing approaches [4], mechanism design [6, 12, 16], and Markov chain models [2]. In this work we formulate and study a simple combinatorial model of the

Transportation Network Pricing problem. In our model, the problem is studied on a graph where distances are symmetric (i.e., the distance from node u to node v is the same as the distance from node v to node u) but demands are asymmetric (i.e., the demand from node u to node v at a price p is not necessarily the same as the demand from node v to node u at the same price p). We consider this assumption to be reasonable in real world scenarios but also note that our results hold within a constant approximation when the distance between any u and v is within constant bounds of the distance between v and u. An available supply of k vehicles can move from node to node in the graph serving demands in the process.

It is not hard to observe that pricing decisions are interconnected with routing decisions. Knowing how many times vehicles will travel from u to v gives insights on how to price the trip from u to v. In such a case, we would want to charge as much as possible while still maintaining a demand high enough to utilize the vehicles that make the trip. Similarly, knowing that a specific trip has a large number of customers who are willing to travel at a high price hints that we should send a large number of vehicles their way. This interconnection makes the problem challenging and interesting.

1.1 Our Contributions

In Section 3 we attempt to disentangle the pricing component from the routing aspect and understand their difficulties separately. We explore the pricing and supply assignment aspect by abstracting away the routing component in a special case of the model. We show that this pricing and assignment version of the problem can be solved in polynomial time. In Section 4, we transition to the general graph model and present an approach that handles pricing and routing as separate stages, achieving a logarithmic approximation to the revenue in the process. In Section 5, applying the insights gathered in the first two, we bring the pricing and routing components back together in a joint optimization stage and provide a constant factor approximation algorithm for general graphs. In Section 6 of the paper we explain how our solution can handle selfish drivers with a small loss in the approximation factor. In Section 7, we show that our techniques generalize to the setting where edges have different demands depending on the time of the day, a setting that can also handle impatient demands that disappear after a certain period. Finally, in Section 8, we discuss how the capacitated version of the problem reduces to the unit capacity case.

1.2 Related Work

Various previous works study pricing in ride-sharing systems. The papers most closely related to ours are [6, 16] who also study a network with price dependent demand curves and seek to maximize revenue. A significant difference in our model is that we consider a general network with arbitrary distances as opposed to the unit distances studied in these two models. The work in [6] considers an infinite supply setting and proves that price discrimination can significantly improve revenue over uniform pricing. The work in [16] considers drivers with preferences for one location over the other and takes a mechanism design approach to achieve incentive compatibility. We note that in our final section we also consider a special, well-motivated, form of driver preferences: wage maximization.

Other papers study more dynamic aspects of ride-sharing platforms such as spatial imbalance and temporal variation [12], dynamic pricing [8], Markov models [2], and queueing models [4]. Other studies focus on market segmentation [1, 3] and car pooling aspects [14].

On the routing side, our work is related to the vehicle routing problem [10, 11] and, more closely, to prize collecting traveling salesperson problems in graphs. Most relevant is work on the *orienteering* problem, the best known algorithms for variants of which are given in [15] and [9]. The work in [15] achieves a 2 approximation for single path orienteering on undirected graphs via a primal-dual algorithm. The work in [9] presents dynamic programming based algorithms, following up on work in [5, 7]. A particular result from [7] that is relevant in our proofs is that undirected orienteering with k paths can be approximated within a factor 3.

2 Model and Preliminaries

In this section we define the specifics of TRANSPORTATION NETWORK PRICING.

Consider a set of locations V and the possible trips between them $E = V \times V$. Let l_e be the length (in time) of trip $e \in E$. We assume trip times are symmetric and $l_e = l_{e'}$ for $e = (u, v)$ and $e' = (v, u)$. For each trip e, we are also given a demand curve $d_e(p)$ that gives the number of agents who are willing to pay a price p for trip e. Naturally, we assume that $d_e(p)$ is a non-increasing function of p. For convenience, for each trip e, we also define the price curve

$$p_e(d) = \max\{p \mid d_e(p) \geq d\}$$

as the maximum price p such that at least d agents are willing to pay p for the trip e. To serve these demands, we have a supply of k service vehicles who can move from location to location and transport the demands. The total trips a service vehicle can make are limited by a time horizon T which is an upper bound on the total length of trips a service vehicle can do. For simplicity and without loss of generality we assume that all edge lengths, demand values, and possible prices are integers.

A solution consists of: (a) a price q_e for each trip e and (b) a path P_i of length at most T for each service vehicle i. A path is a sequence of trips $P = \{e_1, e_2, \ldots, e_m\}$ such that the destination of trip e_j is the source of trip e_{j+1}. The set of service vehicle paths induces a supply s_e for trip e, defined as the number of times e appears in all paths (note that a path might repeat e multiple times). The revenue for trip e is then equal to:

$$r_e = q_e \min\{s_e, d_e(q_e)\}.$$

Our objective is to provide prices and paths that maximize the total revenue:

$$R = \sum_{e \in E} r_e.$$

We assume throughout the paper that the number of agents k and the time horizon T are both polynomial in the size of the graph G. We will design (approximation) algorithms that are polynomial in k, T, and the size of the graph. Figure 1 illustrates a simple instance of the TRANSPORTATION NETWORK PRICING problem and its optimal solution.

A key component of our algorithms is the *revenue function* of an edge $e \in V \times V$ that expresses the maximum amount of revenue that can be obtained from the edge e for a given supply. Mathematically, we define

$$r_e(\ell) = \max_{0 \leq j \leq \ell} \{j \cdot p_e(j)\}$$

.

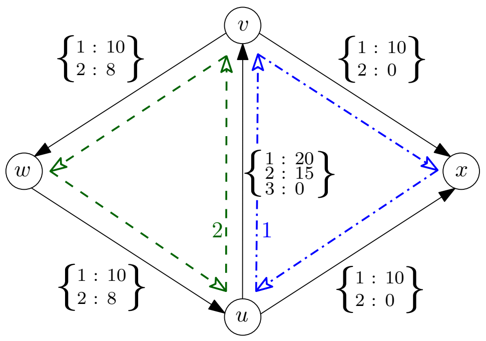

Figure 1 An instance of the TRANSPORTATION NETWORK PRICING problem. Each edge in the digraph represents a trip e of unit length. The maps labeling each edge represent the corresponding price curve, for instance, $p_{(u,v)}(1) = 20$ and $p_{(u,v)}(2) = 15$. For a time horizon of $T = 3$ and $k = 3$ service vehicles, the figure illustrates an optimal solution that assigns two vehicles to the path $\langle (u,v),(v,w),(w,u)\rangle$ and one vehicle to the path $\langle (u,v),(v,x),(x,u)\rangle$ for a total revenue of 82.

3 Node Model: One Trip Per Vehicle

We begin with a warm-up setting in which each vehicle only makes one trip and our decisions amount to pricing edges and assigning vehicles to them. To fit this framework in our model, we can think of the special case with unit edge lengths and a unit time horizon. Since edges have no interaction with each other in this setting, we may equivalently think of them as simply unconnected nodes. For ease of notation, for this special case, we define a demand curve $d_i(\cdot)$ for each node i. The price curve $p_i(\cdot)$ and revenue curve $r_i(\cdot)$ are defined similarly. We term this special case as the TRANSPORTATION NODE PRICING PROBLEM. We show that the problem is poly-time solvable.

▶ **Theorem 1.** *TRANSPORTATION NODE PRICING can be solved in polynomial time when the number k of service vehicles is polynomial in the size of the graph.*

Proof. Consider an arbitrary ordering of the nodes. Let $\mathrm{Opt}(i,j)$ denote the revenue extracted by the optimal solution for the first i nodes with j vehicles. Thus $\mathrm{Opt}(n,k)$ denotes the revenue extracted by the optimal solution for an instance. The following recurrence shows how one can compute this optimal solution via dynamic programming.

$$\mathrm{Opt}(i,j) = \max_{\ell \in \{0,\dots,j\}} \{\mathrm{Opt}(i-1, j-\ell) + r_i(\ell)\} \tag{1}$$

Intuitively, the recurrence searches over all possible number of vehicles to assign to the i^{th} node and solves the residual problem optimally. While $\mathrm{Opt}(n,k)$ only yields the optimal revenue, it is also easy to obtain the actual optimal solution by tracing the path taken by the dynamic program. ◀

As a side-note, we prove that the problem is NP-Hard when k is super-polynomial. We note though that a FPTAS is possible with an approach similar to the one for KNAPSACK.

▶ **Theorem 2.** *TRANSPORTATION NODE PRICING is NP-Hard.*

Proof. We will prove this by reducing KNAPSACK to our problem. The KNAPSACK problem has a collection of n items with sizes s_i and values v_i for $i = 1,2,\dots,n$ and a knapsack of size B. The goal is to pack items of total size at most B and maximize the total value picked.

Our reduction is as follows. For each item i, we construct a node i with the following demands: for a given large number L, the demand for the trip to i is one when the price is Ls_iv_i and Ls_i when the price is v_i. There are no others interested in the trip to i. In other words, we set $r_i(1) = Ls_iv_i$ and $r(Ls_i + 1) = Ls_iv_i + v_i$. The total supply of vehicles is $k = n + LB$.

Observe that in the induced Transportation Node Pricing instance there are, in effect, two possible prices for each node: either set price $p_i = Ls_iv_i$ and serve the unique customer at that price, or set $p_i = v_i$ and serve all $Ls_i + 1$ customers. This means we have the option to either extract total revenue Ls_iv_i spending supply 1 or spend an additional supply of Ls_i to extract an extra v_i. When L is high enough, it is clear that any optimal solution spends the first n of the $n + LB$ supply units to secure the Ls_iv_i from every node, before considering any of the additional v_i's. Then the solution will have to allocate the remaining LB supply units to get additional revenue v_i from any node i to which it allocates Ls_i. This is precisely the original Knapsack problem where all the sizes are scaled by L, which implies that any optimal solution to this Transportation Node Pricing instance recovers an optimal solution to the corresponding Knapsack instance. ◀

4 Separate Price & Route Optimization

In this section we consider the general Transportation Network Pricing model and present an approach that first attempts to determine prices and then to compute routes for the supply vehicles. We show that this algorithm achieves a logarithmic approximation. This section is of interest in itself, but also a warm-up for various aspects we will encounter in our main technical result in the next section (a constant approximation for the same problem that jointly optimizes for prices and routes) such as a reduction to the Undirected Orienteering Problem:

▶ **Definition 3.** *In the Undirected Orienteering problem we are given an undirected graph $G = (V, E)$ with costs on the edges and values on the nodes, and a cost budget T. We seek to find k paths each of cost at most T that maximize the total value of nodes visited.*

Our algorithm proceeds in two steps as follows:

Guess a revenue target and set prices accordingly. We guess a target revenue $\tilde{r} \in \mathbb{R}$ and attempt to extract a revenue of $\tilde{r}$ from each edge e in the network. For every edge e, set the price that would achieve the revenue target $\tilde{r}$ with the smallest supply possible. If $\tilde{r}$ is not achievable on some edge e, give up on e and set an infinite price.

Construct and solve an undirected orienteering instance. Since prices have been determined, each crossing of an edge by a supply vehicle extracts a known revenue. We formulate and solve an Undirected Orienteering instance based on this information. Good constant factor approximation algorithms are known for Undirected Orienteering, something that is the raison d'etre of the graph transformation we perform in this stage. In more detail, we construct an auxiliary undirected graph in which we move the value from edges (i.e., the trips in the transportation graph) to new nodes that we introduce between the trip's endpoints. Every time such a node is visited will represent the corresponding trip being performed once. Hence, the value of such a node is equal to the price set for the corresponding trip. The edge lengths in the auxiliary graph are scaled so that the paths returned by the orienteering algorithm can be converted into sequences of trips in the original network.

4.1 Price Setting

Our algorithm begins with a guess $\tilde{r}$ that is the revenue we will try to extract from every edge (i.e., every trip) in the network. Let $\mathcal{R} = \{r_e(\ell)\}_{e \in E, 0 \leq \ell \leq kT}$ denote the set of all possible revenue values that can be extracted from any edge (note that no trip can be made more than kT times even if all service vehicles perform that one trip). The algorithm will ultimately be run for all possible guesses $\tilde{r} \in \mathcal{R}$. Since $|\mathcal{R}| \leq n^2 kT$, trying all possible revenues in $\mathcal{R}$ can be done in polynomial time.

Once $\tilde{r}$ is fixed, the price that we should set at any edge e can be computed as follows. Let $s_e = \min\{\ell : r_e(\ell) \geq \tilde{r}\}$ be the minimum supply we need to extract value $\tilde{r}$ at e. Then $q_e = p_e(s_e)$ is the price we set for edge e. We now make the following claim.

▶ **Lemma 4.** *Let (q^*, P^*) be an optimal solution with q^* the vector of prices and P^* the paths of the service vehicles. Also, let $\tilde{r}$ be the guess that, with induced prices $\tilde{q}$ and the same paths P^*, maximizes the revenue among all guesses. The revenue extracted by solution $(\tilde{q}, P^*)$ is an H_m-approximation to the revenue extracted by the solution (q^*, P^*), where m is the number of edges in the graph and H_m the m-th harmonic number.*

Proof. Order the edges as $1, 2, \ldots, m$, in order of non-increasing revenue extracted in (q^*, P^*). Call these revenues $r_1^*, r_2^*, \ldots, r_m^*$. Consider the guess $\tilde{r} = r_j^*$ for our algorithm. Fix the paths P^* and set the price that achieves $\tilde{r}$ in each edge e (or infinite price if not possible) as per our algorithm. Consider any edge $e \leq j$. We know that $\tilde{r} = r_j^*$ is achievable on these edges, since the optimal solution extracts at least that on each one. Moreover, we have enough supply to achieve r_j^* on these edges, since P^* allocates enough supply for at least that much. Hence, when the guess is $\tilde{r} = r_j^*$, the solution $(\tilde{q}, P^*)$ extracts value at least jr_j^*.

Let $j^* = \arg\max_j \{jr_j^*\}$ be the guess that yields the maximum value. Thus, we have $j^* r_{j^*}^* \geq j' r_{j'}^*$, for all $j' = 1, 2, \ldots, m$. Then:

$$R^* = \sum_{j'=1}^{m} r_{j'}^* \leq \sum_{j'=1}^{m} \frac{j^* r_{j^*}^*}{j'} = H_m j^* r_{j^*}^* \leq H_m R,$$

with R^* the optimal revenue and R the revenue of $(\tilde{q}, P^*)$. This proves the lemma.

We also proceed to prove that this factor is tight. Consider the case when $r_j^* = 1/j$. Every jr_j^* is unit, whereas their sum is H_m. ◀

4.2 Construction of the Orienteering Instance

Given the prices fixed in the previous stage of the algorithm, our goal is to route the k supply vehicles so that they can extract as much value as possible. This task is similar to the Undirected Orienteering problem, for which a 2-approximation algorithm exists [15]. The main difference in our setting is that the values are on the trips between nodes and not on the nodes. Moreover, these trips are directed. We now describe a transformation to the graph that handles these issues with a small loss in approximation.

We construct an undirected graph $H = (N, A)$ with values on the nodes and costs on the edges as follows. We begin with the nodes of the input transportation network V. All these nodes have value 0. For every ordered pair of nodes $e = (u, v) \in E$, we construct $\min\{kT, d_e(q_e)\}$ nodes $z_e^i, i = 1, 2, \ldots, \min\{kT, d_e(q_e)\}$, with value q_e, i.e., the price of the trip from u to v. For every such node z_e^i, we add an (undirected) edge between it and u and an (undirected) edge between it and v. Both these edges have length equal to l_e, the length of the trip from u to v. Figure 2 shows an example of the construction of the orienteering instance.

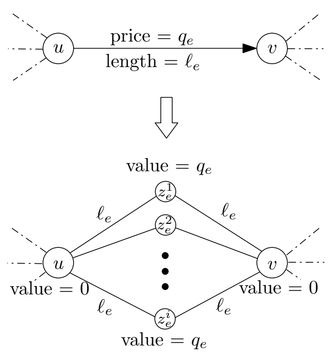

Figure 2 Construction of the orienteering instance with fixed prices.

▶ **Lemma 5.** *Every path of length at most T^* in the input graph G that extracts revenue R can be expressed as a path of length at most $2T^*$ in H that picks value R.*

Proof. Let P be a path of length at most T^* in G. Consider the order in which nodes are visited by the path P in G. We visit the same nodes in the same order in the graph H to obtain a path P'. The i^{th} time we cross an edge from u to v (in G), we go via the intermediate node z_e^i in H. For i larger than $d_e(q_e)$, we go via any of the intermediate nodes (since they all have already been visited). Since the original path P in G has length at most T^*, and every edge $e = (u, v)$ of length l_e in G corresponds to a walk $(u \to z_e^i \to v)$ of length $2l_e$ in H, the new path P' in H has a total length of at most $2T^*$. Let us now compute the value picked up by the path P' in H. Let s_e be the number of times that path P passes through edge e. Then, by definition, the total revenue extracted by P is given by:

$$R = \sum_e q_e \min\{s_e, d_e(q_e)\}.$$

On the other hand, for every edge $e = (u, v)$ in G, by construction the path P' passes through $\min\{s_e, d_e(q_e)\}$ distinct intermediate vertices (z_e^i) each having value q_e. Thus path P' picks up value at least R in H. ◀

▶ **Lemma 6.** *Every path of length at most $\tilde{T}$ in graph H that picks value R can be expressed as a path of length at most $\tilde{T}$ in G that extracts at least revenue R.*

Proof. Let P' be a path in H of length at most $\tilde{T}$ that picks value R. Let v be the first vertex on path P'. Then the path P' departs from node v, visits an intermediate node z_e^i (where $e = (u, v)$ or $e = (v, u)$) and either returns back to v or moves to the opposite node u. In the former case, it pays a cost of $2l_e$ and extracts value q_e. We can construct a path P in G in exactly the same way as follows - starting from node v, visit node u and come back to v paying a total cost of $2l_e$ (since lengths are symmetric) and extracting at least q_e revenue. In the latter case P' visits $v \to z_e^i \to u$ and again pays a cost of $2l_e$ and extracts a revenue of q_e (unless of course all intermediate nodes z_e^i have already been visited earlier). In this case, if $e = (v, u)$, then we simply cross from node v to node u in the path P to earn revenue q_e and a cost of only l_e. On the other hand, if $e = (u, v)$, then in path P, we first take edge (v, u), then take (u, v), and then again take (v, u) so that we end up on the same node on both P and P'. In this step, path P extracts a revenue of at least q_e but pays a cost of $3l_e$.

Let P'_{rev} denote a path in H that is the reverse of P', i.e., it visits the same set of nodes but in the reverse order. Let P_{rev} be the path in G constructed as above starting from P'_{rev}. By construction, both P and P_{rev} extract a revenue of at least R. However, since for any step $v \to z_e^i \to u$ in P', exactly one of P and P_{rev} pay a cost of l_e while the other pays $3l_e$. Thus, we have:

$$\mathsf{length}(P) + \mathsf{length}(P_{\text{rev}}) = \sum_e 4l_e = 2\mathsf{length}(P')$$

and hence at least one of P and P_{rev} have length of at most $\tilde{T}$, proving the lemma. ◀

▶ **Lemma 7.** *For a set of fixed prices, solving the* Undirected Orienteering *problem on graph H with budget T and translating the paths of graph H to paths of graph G as in Lemma 6, gives a 6-approximation to revenue.*

Proof. By Lemma 5 we get that each one of the optimal paths in G can be expressed as a path of length at most $2T$ in H. We solve Undirected Orienteering with a budget of T. We note that the optimal solution with budget T will have at least half the value of the optimal solution with budget $2T$ since we can simply take the better half. This implies the optimal solution for the instance we solve will have value at least half the optimal revenue. By the fact that Undirected Orienteering with k paths can be solved within a 3-approximation, our paths in H will be within 6 of the optimal revenue. Applying Lemma 6 completes the proof. ◀

Putting Lemma 4 with Lemma 7 together, we get the main theorem of the section. More precisely, Lemma 4 suggests that some prices given by our first stage are such that the optimal paths for them will give an H_m-approximation to revenue. Lemma 7 proves that, when we try these prices, we will find paths that approximate the optimal paths within a factor 6. We then get the following theorem.

▶ **Theorem 8.** *Our* separate pricing & routing optimization *algorithm gives a $6H_m$-approximation to revenue, where m is the number of edges.*

5 Joint Price and Route Optimization

In this section we use the insights obtained in the previous two sections to come up with a joint pricing and routing optimization algorithm. The algorithm in effect combines the main ideas of the previous two approaches to price and route at the same time. The algorithm proceeds in the following stages.

Concave approximate revenue curve construction. First, we process all demand curves to obtain the corresponding revenue functions $r_e(\ell)$ (recall that these give the maximum possible revenue that can be achieved at edge e with supply ℓ), which in turn we process to obtain *approximate* revenue functions $\hat{r}_e(\ell)$ that are concave. We prove that we can always find a concave function that satisfies $r_e(\ell) \le \hat{r}_e(\ell) \le 2r_e(\ell)$ for every ℓ. The main reason for performing this step is that the concave approximate revenue functions $\hat{r}_e(\ell)$ satisfy the nice property that the marginal increase:

$$\Delta\hat{r}_e(\ell) = \hat{r}_e(\ell) - \hat{r}_e(\ell - 1)$$

that is caused by the ℓ-th supply on edge e is decreasing. This proves useful when we place these marginal contributions as values to be collected from a graph in the second stage. We will also refer to $\hat{r}_e(\cdot)$ as the *perceived* revenue.

Auxiliary graph construction. The main idea of the second stage of our algorithm is to construct an auxiliary graph that, similarly to our approach in the previous section, a) is undirected, b) has values only on nodes, and c) there is an equivalence between paths in the auxiliary graph and sequences of trips in the input transporation network. Again, the value is moved from edge e, to a collection of nodes $z_e^i, i = 1, 2, \ldots, kT$, that are introduced between its endpoints. This time however, the values of these nodes are not the same. Instead, the value of z_e^ℓ is precisely the marginal perceived revenue $\Delta\hat{r}_e(\ell)$. The transformation of edge lengths is exactly as in the previous section. We then proceed as in the previous section, to solve the induced Undirected Orienteering instance and translate the paths of the auxiliary graph H to paths of the input graph G. Once this is done, the paths induce supplies on the edges which we can use to infer the prices.

5.1 Concave Approximate Revenue Functions

As we also discussed in the preliminary node model, we can express the maximum revenue we can extract from an edge, given supply ℓ, as:

$$r_e(\ell) = \max_{0 \le j \le \ell} \{ j p_e(j) \},$$

with $p_e(j)$ the maximum price that induces demand at least j. As can be seen in Figure 3, we note that $r_e(\cdot)$ need not be a concave function.

However, we can define a concave function $\hat{r}_e(\cdot)$ as the concave envelope of $r_e(\cdot)$. In other words, $\hat{r}_e(\cdot)$ is the lowest-valued concave function such that $\hat{r}_e(\ell) \ge r_e(\ell)$. Concretely, let $[\ell_1, \ell_2]$ be a maximal interval such that the function $r_e(\cdot)$ in this interval is bounded above by the linear interpolation of $r_e(\ell_1)$ and $r_e(\ell_2)$. Then $\forall \ell \in [\ell_1, \ell_2]$, $\hat{r}_e(\ell)$ is obtained by linearly interpolating between $(\ell_1, r_e(\ell_1)$ and $(\ell_2, r_e(\ell_2))$, i.e.,

$$\hat{r}_e(\ell) = \left(\frac{r_e(\ell_2) - r_e(\ell_1)}{\ell_2 - \ell_1} \right) (\ell - \ell_1) + r_e(\ell_1)$$

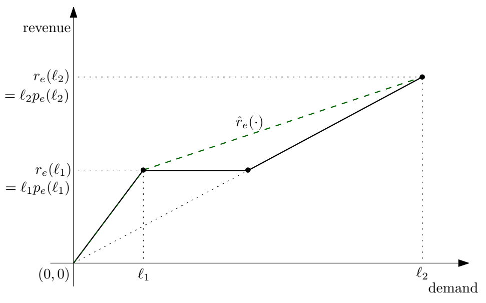

Figure 3 Example revenue function and its concave approximation. The bold line shows the original revenue function $r_e(\cdot)$ for some edge e, and the green dashed line shows its concave approximation $\hat{r}_e(\cdot)$.

We now show that $\hat{r}_e(\cdot)$ point-wise approximates $r_e(\cdot)$ within a factor of 2.

▷ Claim 9. For all $0 \le \ell \le k$, $\hat{r}_e(\ell) \le 2 r_e(\ell)$

Proof. Let $[\ell_1, \ell_2]$ be a maximal interval such that $\hat{r}_e(\ell) > r_e(\ell)$, $\forall \ell \in (\ell_1, \ell_2)$. Note that by definition of ℓ_2, we have $r_e(\ell_2) = \ell_2 p_e(\ell_2)$. Otherwise, if $r_e(\ell_2) = j p_e(j)$ for some $j < \ell_2$, then we have $r_e(\ell_2) = r_e(j)$ and we cannot have $\hat{r}_e(j) > r_e(j)$. Now, since $p_e(\cdot)$ is a non-increasing function, we have

$$r_e(\ell_1) = \max_{0 \le j \le \ell_1} \{ j p_e(j) \} \ge \ell_1 p_e(\ell_2) \tag{2}$$

Hence, we have the following,

$$\frac{r_e(\ell_2) - r_e(\ell_1)}{\ell_2 - \ell_1} \le \frac{r_e(\ell_2) - \ell_1 p_e(\ell_2)}{\ell_2 - \ell_1} \tag{3}$$

$$= \frac{\ell_2 p_e(\ell_2) - \ell_1 p_e(\ell_2)}{\ell_2 - \ell_1} = p_e(\ell_2) \tag{4}$$

Now, for any $\ell \in (\ell_1, \ell_2)$, by definition of $\hat{r}_e(\cdot)$ we have,

$$\hat{r}_e(\ell) = \left(\frac{r_e(\ell_2) - r_e(\ell_1)}{\ell_2 - \ell_1}\right)(\ell - \ell_1) + r_e(\ell_1) \tag{5}$$

$$\le p_e(\ell_2)(\ell - \ell_1) + r_e(\ell_1) \tag{6}$$

$$\le 2\max\{\ell p_e(\ell_2), r_e(\ell_1)\} \tag{7}$$

However, since the revenue function $r_e(\cdot)$ is non-decreasing and the price function $p_e(\cdot)$ is non-increasing, we have

$$r_e(\ell) \ge \max\{\ell p_e(\ell_2), r_e(\ell_1)\} \tag{8}$$

The claim now follows from equations (7) and (8). ◁

5.2 Construction of the Orienteering Instance

As in the previous section, we will construct an undirected graph $H = (N, A)$ with values on the nodes and costs on the edges. Here also, we begin with the nodes of the input transportation network V which again have value 0. For every ordered pair of nodes $e = (u, v) \in E$, we construct kT nodes $z_e^i, i = 1, 2, \ldots, kT$. The value of z_e^i is $\Delta\hat{r}_e(i)$, i.e., the marginal increase in perceived total revenue (as given by the concave approximate revenue functions $\hat{r}_e(\cdot)$) offered by the i-th trip from u to v. For every such node z_e^i, we add an (undirected) edge between it and u and an (undirected) edge between it and v. Both these edges have length equal to l_e, the length of the trip from u to v. Figure 4 shows an example of the construction of the orienteering instance.

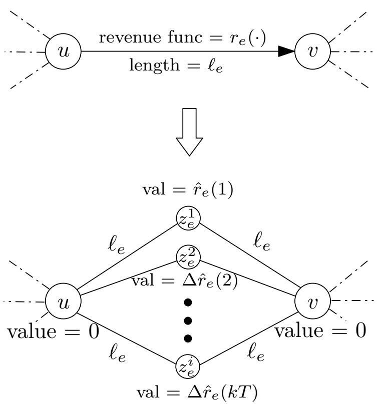

Figure 4 Construction of the orienteering instance with approximate revenues.

▶ **Lemma 10.** *Every path of length at most T^* in the input graph G that extracts perceived revenue R can be expressed as a path of length at most $2T^*$ in H that picks value R.*

Proof. Let P be a path in G of length at most T^* that extracts a perceived revenue of R. We construct a path P' in H as follows - when P uses the edge (u, v) for the i^{th} time, our path in H moves from node u to node v via the intermediate node $z^i_{(u,v)}$. Thus, if P passes through an edge e exactly l times to extract a perceived revenue of $\hat{r}_e(l)$, the path P' also picks up a value of $\sum_{i=1}^{l} \Delta\hat{r}_e(i) = \hat{r}_e(l)$. Thus P' also picks up a total value of R. The length argument follows precisely as in Lemma 5. ◀

▶ **Lemma 11.** *Every path of length at most $\tilde{T}$ in graph H that picks value R can be expressed as a path of length at most $\tilde{T}$ in G that extracts perceived revenue at least R.*

Proof. The path construction and length arguments follow exactly as in Lemma 6. The revenue argument is as follows: Say the path P' in H visits x intermediate nodes corresponding to edge $e = (u, v)$. Then our constructed path P in G crosses edge e at least x times and extracts a perceived revenue of at least $\hat{r}_e(x)$. On the other hand, since $\Delta\hat{r}_e(\cdot)$ is the non-increasing, the value earned by P' from edge e is at most $\sum_{i=1}^{x} \Delta\hat{r}_e(i) = \hat{r}_e(x)$. ◀

Lemmas 10 and 11 together imply that the Undirected Orienteering problem with k paths is equivalent to the Transportation Network Pricing problem with concave revenue functions up to a factor of 2 in the approximation ratio. We can thus directly use a 3-approximation algorithm for Undirected Orienteering as in Lemma 7 to obtain a 6-approximation to the Transportation Network Pricing with concave revenue functions. However, since arbitrary revenue functions can be approximated within a factor of 2 by concave functions, Claim 9 then yields our main result.

▶ **Theorem 12.** *Our* joint pricing & routing optimization *algorithm gives a 6-approximation to revenue for concave revenue functions and a 12-approximation to revenue for general revenue functions.*

6 Selfish drivers

An additional layer of complexity in the ride-sharing context is added by the fact that drivers are independent and will not follow the paths dictated by our algorithm when this is not the behavior that maximizes their total wages. In this section we discuss the presence of selfish drivers in the Transportation Network Pricing setting. We assume that wages are a fixed α fraction of the revenue (i.e., the drivers and the platform share the earnings with a fixed ratio) and argue that, for any given prices, letting the drivers reach an equilibrium is within a factor 2 of the optimal path selections. In this sense, with a small loss in the approximation factor, we may use our algorithms to compute prices assuming the drivers will comply, advertise them, and let the drivers reach an equilibrium.

For the purposes of our argument, we will need to introduce some additional notation and definitions. First, for simplicity of exposition, we set $\alpha = 1$, i.e., assume the drivers receive all revenue. Let x^i_e be the number of times driver i crosses edge e and $x^i = (x^i_e)_{e\in E}$ the vector for driver i over all edges which we will refer to as the driver's *strategy*. Let x be the vector of all driver strategies. Let d_e be the demand under the current price vector (note that, for simplicity, we have dropped dependence of d_e on q_e in the notation, since prices are considered fixed throughout this section) and $s_e(x)$ the supply under x. We are now ready to define the (expected) wage of i on e as:

$$w^i_e(x) = x^i_e q_e \min\left\{1, \frac{d_e}{x_e}\right\}.$$

The interpretation of this expression is that i has probability 1 to get a ride (and hence a payment of q_e) every time she crosses the edge when the demand is at least the supply and probability d_e/x_e when the demand is less than the supply. Another interpretation is that drivers share the total revenue on the edge $r_e = q_e \min\{s_e(x), d_e\}$ proportionally to the number of times they cross it.

A collection of strategies is a *Nash equilibrium* when for every driver i, it is the case that a unilateral deviation to some other vector y^i, induced by a different path selection will not increase her total wages:

$$\sum_{e\in E} w_e^i(x) \geq \sum_{e\in E} w_e^i(x^{-i}, y^i),\ \forall y^i.$$

The *price of anarchy* is the ratio of the total wages in the optimal solution over the total wages in the worst Nash equilibrium. We get the following observation.

▶ **Observation 13.** *The price of anarchy in the* Transportation Network Pricing *problem after prices have been fixed is* 2*.*

Proof. The upper bound follows by the fact that the game we have described is a utility game with a submodular utility function (since the drivers cover demands with their path selections). The upper bound then follows from the main result in [18].

The lower bound follows from the following simple instance of the Transportation Node Pricing submodel. Node 1 has a single demand which is priced at $1+\epsilon$. Node 2 has $1/\epsilon$ demands priced at ϵ. There are $1/\epsilon$ drivers in the game. If all of them head to node 1, their expected wage will be $\epsilon + \epsilon^2$, which is larger than the ϵ they can get from a ride at node 2. Hence, this is a Nash equilibrium with total wages $1+\epsilon$. The optimal solution assigns 1 driver to node 1 and the rest of them to node 2 for total wages 2. ◀

Hence, we reach the conclusion that, in the presence of selfish drivers, our approximation will be a factor 2 away of the ones achieved by our algorithms, i.e., we achieve a 24-approximation using our joint price and route optimization algorithm.

7 Transportation Network Pricing with Dynamic Demands

In this section we consider a natural extension of the Transportation Network Pricing problem where the demands on an edge can now vary as a function of time. We let $d_e(p,t)$ denote the demand on edge e at time t when the price is p. The demand that applies is determined at the moment in time when an agent starts traversing an edge. For ease of notation, we assume that the lengths on edges are specified in the units of time. Since all edge lengths are integral we can assume that the demand changes only at integral time steps. In this section we prove the following theorem.

▶ **Theorem 14.** *The Tranportation Network pricing problem with dynamic demands can be solved in time polynomial in $n, k,$ and T to obtain an approximation of $O(\log n)$.*

To obtain the best possible result, we proceed in two steps. In step 1, we reduce the Transportation Network Pricing with dynamic demand problem to single agent Transportation Network Routing with Unit Time Windows. In step 2, we reduce the single agent Transportation Network Routing with Unit Time Windows problem to Directed orienteering with Unit Time Windows problem. In the end, we obtain an approximation factor of $O(\log n)$.

7.1 Step 1: Transportation Network Pricing to Transportation Network Routing

This reduction is similar to section 5.2. We present an additional step where we reduce the problem from k agents to a single agent.

First since the demand varies with time, we redevelop some of the notation to depend on time. The price curve $p_e(d,t) = \max\{p|d_e(p,t) \geq d\}$ is the price at which the demand is at least d. The revenue from assigning l agents to edge e at time t is $r_e(l,t) = max_{0\leq j\leq l}\{jp_e(j,t)\}$. We approximate the revenue curve using a concave function $\hat{r}_e(t,i)$ constructed similar to Lemma 9 with the guarantee that $r_e(l,t) \leq \hat{r}_e(l,t) \leq 2r_e(l,t)$.

Using the concave revenue functions $\hat{r}$, we reduce the problem to one of constructing paths on a graph. We will call this the Transportation Network Routing with Unit Time Windows problem.

▶ **Definition 15.** *In Transportation Network Routing with Unit Time Windows problem, we are given a directed graph $G(V,E)$ with values v_e and time window of unit length $[t_e, t_e+1]$ associated with each edge. The goal is to find k paths such that each path has length at most T and the sum of values v_e of all edges that appear in at least one path is maximized. The value v_e on an edge is only collected if the path starts on the edge e during the time window $[t_e, t_e+1]$. If the same edge e appears in multiple paths, its value v_e is collected only once.*

Given our input instance $G = (V,E)$ of the Transportation Network Pricing with Dynamic Demands problem, we construct an instance $G' = (V,E')$ of the Tranportation Network Routing with Unit Time Windows problem as follows. This graph has the same set of vertices V. For each edge (u,v) in the original graph G, we construct kT parallel edges between u and v. The value of the (l,t)'th edge for $l \in \{0,1,\ldots k\}$ and $t \in 0,1,\ldots,T-1$ is $\Delta\hat{r}(l,t) = \hat{r}(l,t+1) - \hat{r}(l,t)$. Then similar arguments as section 5.2 guarantee that an α-approximation to Transportation Network Routing with Unit Time Windows problem yields a 2α-approximation to Transportation Network Pricing with Dynamic Demands. Note that since the paths map one-to-one in time between the two instances the time windows do not create any new challenge.

Next we show that an approximation algorithm for the Transportation Network Routing with Time Windows problem with one agent can be used to obtain a slightly worse approximation for k agents. The proof is similar to an analogous result by [7] for orienteering problem.

▶ **Theorem 16.** *An α-approximation algorithm for the transportation network problem with time windows for a single agent can be used to obtain an $(\alpha+1)$-approximation for the transportation network problem with time windows for k agents.*

Proof. Given an α-approximation algorithm for the transportation network problem for a single agent, we use it repeatedly to solve the problem for k agents. After the algorithm has selected path A_i for the i'th agent. We set the value on all edges used by the path A_i to zero before calling the algorithm for the next agent. This ensures that all paths constructed by the algorithm are edge disjoint. Let $O = (O_1, O_2, \ldots, O_k)$ denote the optimal solution with k agents decomposed into the k agents' paths. Let Δ_i denote the edges from path O_i that have already been used by some path A_j (where $j < i$) by algorithm before path A_i is chosen. There is a feasible path using all the edges of $O_i \setminus \Delta_i$. Thus we have that $v(A_i) \geq \frac{1}{\alpha}\{v(O_i) - v(\Delta_i)\}$. Summing these over all agents, $\alpha v(A) \geq v(O) - v(\Delta)$. Moreover $v(\Delta) \leq v(A)$. Hence we conclude that $(\alpha+1)v(A) \geq v(O)$. ◀

With this result, it suffices to obtain an approximation for the TRANSPORTATION NETWORK ROUTING WITH TIME WINDOWS problem for a single agent.

7.2 Step 2: Transportation Network Routing to Directed Orienteering

We next reduce TRANSPORTATION NETWORK ROUTING WITH TIME WINDOWS to directed orienteering with fixed start locations and unit time windows.

▶ **Definition 17.** *In DIRECTED ORIENTEETING WITH UNIT TIME WINDOWS AND FIXED START we are given a directed graph $G = (V, E)$ with costs on the edges and values on the nodes, and a cost budget T. There is also a time-window of unit length associated with each node. The value from a node is only collected if it is visited within the time window. We seek to find a path of cost at most T that starts at node $s \in V$ such that the value collected is maximized.*

This problem can be solved in polynomial time to obtain an approximation of $O(\log n)$. This follows from [9] that provide an approximation of $O(\alpha)$ where α is approximation for directed orienteering, [13] that provides an $O(\beta \log n)$ approximation for directed orienteering where β is the integrality gap for asymmetric TSP, and [17] that provides a constant factor integrality gap for asymmetric TSP.

We start with the TRANSPORTATION NETWORK ROUTING instance $G' = (V, E')$ with length l_e, value v_e and unit time window $[t_e, t_e + 1]$ associated with each edge. We construct a directed graph $H = (N, A)$ with values on the nodes and costs on the edges. The set of vertices $N = V \cup I$. The set V is the set of original vertices. The set I is the set of *intermediate* vertices, with one vertex z_e for each edge e in E'. In the graph H, for each edge $e = (u, v) \in E'$, we add an edge (u, z_e) of length l_e and an edge (z_e, v) of length zero. We associate value v_e and time window $[t_e + l_e, t_e + l_e + 1]$ with each intermediate node z_e and value 0 with nodes in V.

We prove the following lemmas to obtain the final result:

▶ **Lemma 18.** *Any path of length at most T^* in graph G' that picks value V can be expressed as a path of length T^* in H that picks value V*

Proof. Consider edge $e = (u, v)$ in the path. We map it to the edges $(u, z_e), (z_e, v)$. The path collects the value if it starts traversing the edge during $[t_e, t_e + 1]$. In the orienteering instance the path will get to the intermediate node z_e in time window $[t_e + l_e, t_e + l_e + 1]$ so the same value v_e can be collected. ◀

For mapping a solution in graph H to a solution in graph G' the main blocker is that the path may start or end at one of the z_e nodes. To tackle this we call the orienteering problem with a fixed start node. We prove the following lemma.

▶ **Lemma 19.** *Given a path of length T^* that starts at a node a non-intermediate node s in H and collects value V, we can construct a path of length T^* in the graph G' with value V starting at node s in graph G'*

Proof. If the path in H ends at an intermediate node z_e, it can be extended to the next non-intermediate node without increasing its length. We can assume that the path begins and ends in non-intermediate nodes. After that there is one-to-one mapping between the portions of the path. An intermediate node z_e only connects to the end node v of the edge e. So we can always find pairs of segments $(u, z_e), (z_e, v)$ in the path. These can be mapped to $e = (u, v)$ in graph G'. The value v_e is the same, the lengths of the segments are the same and the time window $[t_e + l_e, t_e + l_e + 1]$ in the graph H maps to $[t_e, t_e + 1]$ which is precisely when the constructed path will begin traversing edge e. ◀

To complete the proof of Theorem 14, we need to call the orienteering subroutine with all possible start nodes in the set V. We can choose the best solution among those and it will be an $O(\log n)$-approximation to the optimal solution to the SINGLE AGENT TRANSPORTATION NETWORK ROUTING problem. Trying different start nodes does not degrade the running time by more than a factor of n.

8 Capacitated Vehicles

For the sake of simplicity, we have studied the problem in terms of unit capacity vehicles that can serve a single demand when crossing an edge. We now explain that a simple transformation can reduce the capacitated version of the problem where each vehicle can serve up to a fixed number c of demands to the unit capacity case. The main idea is as follows: We will transform any given demand curve into an equivalent one such that a) for any given price, the number of buyers that is willing to buy is a multiple of c and b) the revenue functions remain intact. Achieving that would then allow us to change the units of measurement by a factor c and have each unit of demand correspond to a number of buyers equal to the vehicle capacity, in effect recovering the unit capacity model. Note that the revenue functions that give the optimal revenue of an edge as a function of the supply assigned to it are the only input given to our main algorithms. This implies our approximation results are preserved by such a reduction.

Consider a given price curve $p_e(\cdot)$. For any given integer s, let,

$$\rho_s = \max_{(s-1)c<d\leq s\cdot c} d\cdot p_e(d), \tag{9}$$

be the maximum revenue obtained when using exactly s service vehicles. Our modified curve is such that:

$$\hat{d}_e(p) = s\cdot c, \text{ for all } p\in\left(\frac{\rho_{s+1}}{(s+1)c}, \frac{\rho_s}{s\cdot c}\right]. \tag{10}$$

The following lemma proves that the modified demand curve is well defined.

▶ **Lemma 20.**

$$\frac{\rho_{s+1}}{(s+1)c} \leq \frac{\rho_s}{s\cdot c}.$$

Proof. Note that $\rho_s \geq p_e(s\cdot c)s\cdot c$, since using $d = s\cdot c$ is an option in (9). Also, $\rho_{s+1} \leq p_e(s\cdot c)(s+1)c$, since the highest price for which supply $s+1$ is needed is at most $p_e(s\cdot c)$ and the highest demand for which supply $s+1$ is needed is $(s+1)c$. The two inequalities can be combined to give:

$$\frac{\rho_{s+1}}{(s+1)c} \leq p_e(s\cdot c) \leq \frac{\rho}{s\cdot c},$$

which completes the proof. ◀

The demand curve (10) by definition satisfies the property that only a multiple of c buyers will show up under any price. Then, the j-th such group of c buyers can be replaced with a single buyer with value ρ_j/j. By Lemma 20 these ρ_j/j values are nonincreasing, as necessary for demand curves. Moreover, the optimal revenue obtained by any given number of supply vehicles remains the same, which suggests the revenue curves are unchanged and our transformation is completed as desired.

References

1 Reza Alijani, Siddhartha Banerjee, Sreenivas Gollapudi, Kostas Kollias, and Kamesh Munagala. The segmentation-thickness tradeoff in online marketplaces. *POMACS*, 3(1):18:1–18:26, 2019.

2 Siddhartha Banerjee, Daniel Freund, and Thodoris Lykouris. Pricing and optimization in shared vehicle systems: An approximation framework. In *Proceedings of the 2017 ACM Conference on Economics and Computation, EC '17, Cambridge, MA, USA, June 26-30, 2017*, page 517, 2017.

3 Siddhartha Banerjee, Sreenivas Gollapudi, Kostas Kollias, and Kamesh Munagala. Segmenting two-sided markets. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, pages 63–72, 2017.

4 Siddhartha Banerjee, Ramesh Johari, and Carlos Riquelme. Pricing in ride-sharing platforms: A queueing-theoretic approach. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation, EC '15, Portland, OR, USA, June 15-19, 2015*, page 639, 2015.

5 Nikhil Bansal, Avrim Blum, Shuchi Chawla, and Adam Meyerson. Approximation algorithms for deadline-tsp and vehicle routing with time-windows. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 166–174, 2004.

6 Kostas Bimpikis, Ozan Candogan, and Daniela Sabán. Spatial pricing in ride-sharing networks. *Operations Research*, 67(3):744–769, 2019.

7 Avrim Blum, Shuchi Chawla, David R. Karger, Terran Lane, Adam Meyerson, and Maria Minkoff. Approximation algorithms for orienteering and discounted-reward TSP. *SIAM J. Comput.*, 37(2):653–670, 2007.

8 Juan-Camilo Castillo, Dan Knoepfle, and Glen Weyl. Surge pricing solves the wild goose chase. In *Proceedings of the 2017 ACM Conference on Economics and Computation, EC '17, Cambridge, MA, USA, June 26-30, 2017*, pages 241–242, 2017.

9 Chandra Chekuri, Nitish Korula, and Martin Pál. Improved algorithms for orienteering and related problems. *ACM Trans. Algorithms*, 8(3):23:1–23:27, 2012.

10 George B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 6(1):80–91, 1959.

11 Gilbert Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59:345–358, 1992.

12 Hongyao Ma, Fei Fang, and David C. Parkes. Spatio-temporal pricing for ridesharing platforms. In *Proceedings of the 2019 ACM Conference on Economics and Computation, EC 2019, Phoenix, AZ, USA, June 24-28, 2019.*, page 583, 2019.

13 Viswanath Nagarajan and R. Ravi. The directed orienteering problem. *Algorithmica*, 60(4):1017–1030, 2011.

14 Michael Ostrovsky and Michael Schwarz. Carpooling and the economics of self-driving cars. In *Proceedings of the 2019 ACM Conference on Economics and Computation, EC 2019, Phoenix, AZ, USA, June 24-28, 2019.*, pages 581–582, 2019.

15 Alice Paul, Daniel Freund, Aaron Ferber, David B. Shmoys, and David P. Williamson. Prize-collecting TSP with a budget constraint. In *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, pages 62:1–62:14, 2017.

16 Duncan Rheingans-Yoo, Scott Duke Kominers, Hongyao Ma, and David C. Parkes. Ridesharing with driver location preferences. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 557–564, 2019.

17 Ola Svensson, Jakub Tarnawski, and László A. Végh. A constant-factor approximation algorithm for the asymmetric traveling salesman problem. *J. ACM*, 67(6):37:1–37:53, 2020.

18 Adrian Vetta. Nash equilibria in competitive societies, with applications to facility location, traffic routing and auctions. In *43rd Symposium on Foundations of Computer Science (FOCS 2002), 16-19 November 2002, Vancouver, BC, Canada, Proceedings*, page 416, 2002.