Facility Location for Congesting Commuters and Generalizing the Cost-Distance Problem

Thanasis Lianeas[1], Marios Mertzanidis[2], and Aikaterini Nikolidaki[3]

[1]University of West Attica, [2]Purdue University, [3]National Technical University of Athens.
lianeas@corelab.ntua.gr, mmertzan@purdue.edu, aiknikol@mail.ntua.gr

Abstract. In Facility Location problems there are agents that should be connected to facilities and locations where facilities may be opened so that agents can connect to them. We depart from Uncapacitated Facility Location and by assuming that the connection costs of agents to facilities are congestion dependent, we define a novel problem, namely, Facility Location for Congesting (Selfish) Commuters. The connection costs of agents to facilities come as a result of how the agents commute to reach the facilities in an underlying network with cost functions on the edges. Inapproximability results follow from the related literature and thus approximate solutions is all we can hope for. For when the cost functions are nondecreasing we employ in a novel way an approximate version of Caratheodory's Theorem [5] to show how approximate solutions for different versions of the problem can be derived. For when the cost functions are nonincreasing we show how this problem generalizes the Cost-Distance problem [38] and provide an algorithm that for this more general case achieves the same approximation guarantees.

1 Introduction

Facility Location problems are among the well studied problems in Theoretical Computer Science (see, e.g., the related work section) and other fields [41, 51]. In such problems, there are some facilities to be opened in available locations and agents need to connect to them. Opening facilities usually comes with a facility cost and connecting agents to their closest facility comes with connection costs. The goal is to minimize the sum of these costs. The connection costs are usually fixed and ignore congestion effects when connecting to the facilities. In this work, we assume that there are congestion effects when the agents commute to connect to the facilities.

Congestion Games are one of the most studied classes of games in Algorithmic Game Theory. In these games, agents choose subsets of resources and incur costs based on the congestion on the resources they have chosen. In Network Congestion Games, the very basic version of Congestion Games, there is an underlying network and agents choose paths in the network that connect their sources to their targets. Agents are selfish, aiming to minimize their costs without caring for any type of social welfare, which could be the goal of a central organizer. For that reason, it makes sense to examine both outcomes where agents are satisfied by being on shortest paths, thus being at equilibrium, and outcomes where agents are assigned so that the social welfare, in any chosen sense, is minimized.

Departing from Uncapacitated Facility Location we define Facility Location for Congesting/Selfish Commuters. We assume there is an underlying network on the nodes of which facilities may be opened with a node-specific cost. Some nodes of the network are source nodes from where infinitesimally small agents, that form demands, depart to reach facilities. For any given set of facilities, there are many flows that may route the demands to the facilities, with the congestion cost varying among them. The goal is to minimize the sum of the facilities opening costs for the opened facilities plus the routing costs of the demands. When minimizing, similar to Congestion Games, we consider two cases for the flows, asking either to be simply feasible or to be equilibrium flows. In the first case, we say that we deal with Facility Location for Congesting Commuters and in the second case we say that we deal with Facility Location for Selfish Commuters.

The congestion costs are modeled using cost functions that come with the edges. Both cases of nondecreasing and nonincreasing cost functions on the edges have been considered in the Congestion Games

literature. Interestingly, Facility Location for Congesting Commuters with nonincreasing cost functions generalizes the Cost-Distance problem [38]. In this problem, there is an underlying undirected network where demands are to be routed to some target and every edge comes with two types of costs. The first type of cost can be seen as a cost for building the edge and the second as a cost for traversing the edge. The goal is to return among all Steiner trees that connect the sources to the target the one that minimizes the building plus the routing costs.

Contribution: In this work we define Facility Location for Congesting Commuters (*FLCC* in short), a problem generalizing Facility Location Problems by assuming congestion effects that affect the agents when traveling/commuting to connect to their facilities. Moving one step further, we also define Facility Location for Selfish Commuters (*FLSC* in short) where the agents are not centrally controlled and are not assigned to paths in any convenient way. Instead, they are selfish wanting to travel in shortest paths, thus we assume that they ought to be at equilibrium. If we see these problems through an Algorithmic Game Theory lens, it is like having Network Congestion Games where the targets for the demands have to be specified (with some cost) affecting any optimization task we wish to solve.

One can think of many variants for both *FLCC* and *FLSC* since the underlying network may be directed or undirected, may be a single-commodity, a single-source multi-commodity or a multi-commodity network and may have nonincreasing or nondecreasing cost functions, while, in addition, the opening costs for the facilities may be left arbitrary or be constrained, e.g., we may require to have the same opening cost for all facilities. We note that when the cost functions are constants the two problems coincide and reduce to Uncapacitated Facility Location for which hardness of approximation results are known [23]. Thus, we can aim only for approximation algorithms for these problems. One can show (Appendix A) that the techniques used to tackle Facility Location hardness results (Local search and Linear Programming) fail in this more general setting (at least if applied in a straightforward way).

Our first result concerns directed, single-source multi-commodity networks with nondecreasing latency functions and arbitrary opening costs. We employ ideas used in Congestion Games that work when the latency functions are a-Lipschitz, appplying an approximate version of Caratheodory's theorem (Theorem 1) to provide an algorithm (Section 3) that performs well whenever the maximum length among the paths is $o(|V|)$ (Theorem 2). The presented algorithm approximately solves *FLSC*, but a similar and simpler approach can be used to approximately solve *FLCC*. We note that sparsification techniques similar to ours have been used in the past [21, 19] yet this is the first time that such techniques are applied for multi-target instances, where additionally the optimal placement for the targets, i.e., the facilities, has to be determined as well.

Our second result concerns undirected multi-commodity networks with nonincreasing cost functions and common opening costs for the facilities. In this case, as discussed already and as we show in Theorem 4, *FLCC* generalizes the Cost-Distance problem [38] for which Chuzhoy et al.[16] have shown an $\Omega(\log\log|S|)$ hard to approximate result, where S is the set of sources. Thus, this hardness result also holds for our more general setting. We propose an approximation algorithm (Subsection 4) for *FLCC* that is influenced by the algorithm of Meyerson et al. [38] but, in order to make the algorithm work for multicommodity networks and more general latency functions, it uses a novel matching mechanism (for matchings that have to be done therein). This complicates the analysis but is sufficient to provide the same approximation guarantees (Theorem 5). The proposed algorithm works for *FLCC* but can also be used for *FLSC* with the factor of approximation multiplied by $PoA(D)$, i.e., the Price of Anarchy [33] for cost functions in class D [17]. Conceptually, $PoA(D)$ captures how much worse can an equilibrium flow be when compared to the optimal flow for the cost functions used.

Related Work: Facility Location problems are among the most well-studied problems in algorithmic literature. Depending on the costs and constraints of the problem different variants of facility location emerge. Our main problem can be seen as a generalization of the Metric Uncapacitated Facility Location problem.

The first constant factor approximation algorithm for this problem was provided by Shmoys et al. [46]. They provided a guarantee of 3.16 times the optimal cost. Jain and Vazirani [28] use a Primal-Dual technique to achieve a 3-approximate algorithm. Chudak and Shmoys [14] found a $(1+2/e)$-approximate algorithm and finally Li [34] provided a 1.488-approximation algorithm for the problem. As far as inapproximability results are concerned, the work of Guha and Khuller [23] prove that it is NP-Hard to approximate this problem with a factor better than 1.463.

We also examine a k-median variant of our main problem. Charikar et al. [11] were able to provide a $O(\log(n) \cdot \log\log(n))$-approximation algorithm for the k-median facility location, which was obtained by derandomizing and refining an algorithm proposed by Bartal [7, 8]. Charikar et al. [12] were able to provide the first constant approximation algorithm for the k-median problem with a guarantee of $6\frac{2}{3}$. Finally, Arya et al. [3] provided a $3+\epsilon$ approximation algorithm. One can also derive a $(1+\frac{2}{e}-\epsilon)$ inapproximability result by adapting the proof of hardness for the facility location problem provided by Guha and Khuller [23] (this was observed by Jain et al. [27]). Additionally many other variants of facility location problems have been studied such as Multi-Level Facility Location [1, 22], Connected Facility Location [31, 24, 47, 25], and Universal Facility Location [49, 2, 35, 40, 4].

Facility Location problems where congestion effects are present have been studied in the past although in those works the congestion does not affect the connection costs, while, additionally, such works are mainly experimental [26, 36, 18, 6, 45, 20, 10, 50, 29]. Instead, our work is purely theoretical and to the best of our knowledge is the first to consider that the congestion affects the connection costs, and it does so in a complex way similar to that assumed in (network) Congestion Games.

Congestion Games [42] provide a natural model for non-cooperative resource allocation in large-scale communication networks and have been the subject of intensive research in Algorithmic Game Theory. Many variants have been considered in the literature but the one closely related to ours is that of nonatomic network Congestion Games, also known as non atomic Selfish Routing. Through the use of a potential function [42] equilibrium existence is guaranteed and its computation reduces to solving a convex program, which is also the case in our problem.

The selfish behavior of the users may cause inefficiency, measured using the Price of Anarchy [33]. Related to our work, one way to reduce this inefficiency is to find and remove edges that cause the so called Braess Paradox [9, 39], i.e., that removing edges may improve the networks' performance. Braess Paradox seems not an artifact of optimization theory [32, 43, 48, 15] and resolving it has been proved hard to approximate even for simple instances [44]. This line of work relates to our work in two ways. First, resolving the paradox lies under the umbrella of Network Design, which is also the case for Facility Location for Congesting/Selfish Commuters. Second, in this work we employ and generalize techniques that have been applied for resolving the paradox [21, 19].

Facility Location for Congesting/Selfish Commuters generalizes the Cost-Distance Problem. It was introduced by Meyerson et l. [37, 38], and generalizes many important problems, e.g., single-sink buy-at-bulk with variable pipe types between different sets of nodes, facility location with buy-at-bulk type costs on edges, constructing single source multicast trees with good cost and delay properties, and multi-level facility location. In the same work Meyerson et al. proposed the first known $O(\log|S|)$ randomized approximation algorithm for the single-sink case on a given undirected graph, where S is the set of sources. This algorithm was subsequently derandomized by Chekuri et al. [13]. Chuzoy et al. [16] presented an $\Omega(\log\log|S|)$ inapproximability result for the single-sink case, which also holds for our problems.

2 Preliminaries

For both **Facility Location for Congesting Commuters** and **Facility Location for Selfish Commuters** (*FLCC* and *FLSC* in short, respectively) an instance is defined in a similar way and the difference from one problem to the other is an extra constraint regarding the routing. An instance for these problems consists of

a network $G(V, E)$ (directed or undirected) together with a set of source nodes $S \subseteq V$, a set of continuous non-negative cost functions $\{\ell_e\}_{e\in E}$ for the edges, traffic demands $\{w_i\}_{i\in S}$ for the source nodes, consisting of infinitesimal agents, and a set of costs $\{B_i\}_{i\in V}$ that capture the (possibly different) costs for opening facilities on the nodes. For convenience we set $|V| = n$ and $|E| = m$ and in the case of single source instances, i.e., instances where $S = \{s\}$, w.l.o.g. we assume that $w_s = 1$, since cost functions can be adapted appropriately (setting $\ell_e^{new}(x) = \ell_e(w_s x)$).

Cost functions. For every edge e, ℓ_e maps non-negative reals to non-negative reals, i.e. $\ell_e : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$. We care both for nondecreasing and nonincreasing cost functions. For the first part of this work every ℓ_e is assumed to be nondecreasing. For the second part we focus on nonincreasing cost functions, additionally asking for every ℓ_e to be such that $x \cdot \ell_e(x)$ is a nondecreasing concave function. This case can be seen as agents sharing the cost of an edge they use, with the total cost of each edge increasing if more agents use the edge. In each section it will be clearly noted whether we examine the nondecreasing or nonincreasing variant.

Flows. The facilities may open on the network's nodes and act as targets to which the traffic demands travel. Once they are opened, the demands travel from their sources to the targets/facilities forming flows. Formally, let $F \subseteq V$ denote the nodes with opened facilities and $\mathcal{P}_{i,F}$ denote the network's paths that start from node $i \in S$ and end with a node in F. Also, let $\mathcal{P}_F = \cup_{i\in S}\mathcal{P}_{i,F}$. Given F, a flow $x = \{x_p\}_{p\in\mathcal{P}_F}$ is a vector assinging nonnegative reals to the paths of $\mathcal{P}_F$, and it is feasible if for every $i \in S : \sum_{p\in\mathcal{P}_{i,F}} x_p = w_i$. For a flow x and an edge e, we let $x_e = \sum_{p:e\in p} x_p$ denote the amount of flow that x routes through e. To denote a feasible flow for a set of facilities F we write $x(F)$ and we may write x instead of $x(F)$ whenever F is clear from the context.

Costs. Given F and a feasible flow x, we say that a path p is used if for all $e \in p : x_e > 0$. The cost of a path p under x is the sum of the costs of its edges, i.e., $\ell_p(x) = \sum_{e\in p} \ell_e(x_e)$. The total routing cost of a flow x is $RC(x) = \sum_{p\in\mathcal{P}} x_p \cdot \ell_p(x)$ or equivalently $RC(x) = \sum_{e\in E} x_e \cdot \ell_e(x_e)$.

Equilibria. We say that x is a Nash flow if for all $i \in S$, and for any used path $p \in P_{i,F}$ and any path $p' \in P_{i,F}$ we have $\ell_p(x) \leq \ell_{p'}(x)$, i.e., for every source the corresponding demand is routed through minimum cost paths. We say that x is an ϵ-Nash flow if for all $i \in S$, and for any used path $p \in P_{i,F}$ and any path $p' \in P_{i,F}$ we have $\ell_p(x) \leq \ell_{p'}(x) + \epsilon$. Using potential function arguments (for the continuous version of Rosenthal's potential [42]) one can show that, given F, a Nash flow always exists.

Solutions and Objective. Given an *FLCC* or an *FLSC* instance, a solution to it is a set $F \subseteq V$ determining the nodes on which a facility opens. A solution F is feasible if there exists a feasible flow x (given F). The goal for *FLCC* is to find a feasible solution that minimizes the sum of the total routing cost, under the opened facilities and searching among all feasible flows, plus the cost for the facilities that we opened,[1] i.e., among all feasible F and all feasible flows for F find the one minimizing $C(F) := \sum_{e\in E} x_e(F) \cdot \ell_e(x_e(F)) + \sum_{i\in F} B_i$. For *FLSC* the objective is the same, i.e., minimizing the above total cost, but there is an extra constraint that allows only flows $x(F)$ that are equilibria.

The following definitions serve as reference points for the problems we generalize in this work.

Cost-Distance Problem [38]. We are given an undirected graph $G(V, E)$ along with a set of source nodes $S \subseteq V$ which need to be connected to a single sink node $t \in V$. Each node $s \in S$ comes with an associated demand w_s to be routed to t. Each edge $e \in E$ is equipped with a cost c_e and a length l_e. The goal is to find a connected subgraph $G'(V', E')$ of G which contains all the source nodes and the sink and minimizes the sum $\sum_{e\in E'} c_e + \sum_{s_i\in S} w_s \cdot l(s_i, t)$, where $l(s_i, t)$ denotes the length of the shortest s_i-t path in G'.

Facility Location and k-median. The problem we depart from is similar to the Metric Uncapacitated Facility Location Problem. We are given an undirected graph with costs $\{c_e\}_{e\in E}$ on the edges. On every vertex

[1] We mix routing and opening costs but we can assume that, e.g., opening costs are multiplied to be measured in routing costs units.

i there might be a demand d_i and a facility can be opened with opening cost B_i. The goal is to open some facilities where the demands can be connected to, so that the sum of the total facility opening cost and the weighted cost of connecting the demands to the facilities is minimized. For the k-median variant there are no opening costs and the goal is to open (up to) k facilities so that the weighted cost of connecting the demands to the facilities is minimized.

3 Instances with Nondecreasing Functions

In this section we consider *FLCC* and *FLSC* on directed networks with nondecreasing cost functions. One can show (see Appendix A) that the approaches that give approximation algorithms for Facility Location problems fail in our more general case (at least if applied in a straightforward way). Instead, inspired by techniques that have been used in Congestion Games, we restrict our attention on single-source instances, i.e., instances where $|S| = 1$. We additionally assume that all cost functions are a-Lipschitz. In Appendix B we note on why these techniques are of no use in the multi-source case, i.e., when $|S| > 1$.

Our proposed algorithm approximately solves *FLSC* but can be easily adapted so that it works for *FLCC*. It makes use of an approximate version of Caratheodory's theorem to provide an algorithm that solves any such instance and performs well (timewise) whenever the maximum length among the paths is $o(|V|)$, e.g., it runs in polynomial or quasipolynomial time when the length of the paths of the network is bounded by a constant or a polylog factor, respectively. Note that this restricts also the number of paths $|P|$. Since we focus on single-source instances, w.l.o.g. we assume that the total demand is $w_s = 1$.

Caratheodory's theorem states that if a point $x \in \mathbb{R}^d$ lies in the convex hull of a set $X = \{x_1, x_2, \ldots, x_{|X|}\}$ then x can be written as the convex combination of (at most) $d+1$ elements of X. The approximate version of Caratheodory's theorem that we use is the following.

Theorem 1 ([5]). *Let X be a set of vectors $X = \{x_1, ..., x_n\} \subset \mathbb{R}^d$ and $\epsilon > 0$. For every $\mu \in conv(X)$ and $2 \leq p \leq \infty$ there exists an $O(\frac{p \cdot \gamma^2}{\epsilon^2})$-uniform vector $\mu' \in conv(X)$ such that $\|\mu - \mu'\|_p \leq \epsilon$, where $\gamma = \max_{x \in X} \|x\|_p$ and a vector is k-uniform when it can be expressed as an average of k vectors of X with replacements allowed, i.e., as $\sum_{j=1}^{k} \frac{x_{i_1}}{k}$, with $x_{i_1}, \ldots, x_{i_k} \in X$.*

We will apply the theorem for points in $\mathbb{R}^{n+m}$ using the $|| \cdot ||_2$ norm. We consider arbitrary orderings for the m edges and the n nodes of the network and for each path $p \in \mathcal{P}$ we create vector $x_p = (x_{p,1}, \ldots, x_{p,m}, x_{p,m+1}, \ldots, x_{p,m+n})$ where $x_{p,e} = 1$ if $e \in p$ and 0 otherwise, for $1 \leq e \leq m$, and $x_{p,m+v} = 1$ if p ends in facility v and 0 otherwise, for $1 \leq v \leq n$. Let $X = \{x_p\}_{p \in \mathcal{P}}$ be the set of all these created vectors.

Suppose in the optimal solution we have a set of opened facilities F^* and a corresponding Nash flow f^* that uses a set of paths $\mathcal{P}'$. Note that, since $w_s = 1$, f^* lies in the convex hull of $X' = \{x_p : p \in \mathcal{P}'\}$ and thus in the convex hull of X, i.e., $f^* \in conv(X)$. Also, assuming a bound M on the length of the paths of $\mathcal{P}$, $\gamma = \max_{x \in X} \|x\|_2 = \sqrt{1 + \max_{p \in P} |p|} \leq \sqrt{M+1}$, as every vector in X has 1's in at most M edge coordinates and one node coordinate.

Using Theorem 1 for set X, the $\| \cdot \|_2$ norm and $\epsilon_1 > 0$ to be defined later, we can see that there exists $k = O(\frac{M}{\epsilon_1^2})$ paths, say $i_1, i_2, ..., i_k$, such that for flow $\hat{f} = \sum_{j=1}^{k} \frac{x_{i_j}}{k}$ it holds that $\|f^* - \hat{f}\|_2 \leq \epsilon_1 \Rightarrow |f_e^* - \hat{f}_e| \leq \epsilon_1$. Assuming that the cost functions are a-Lipschitz we get $|\ell_e(f^*) - \ell_e(\hat{f})| \leq a \cdot \epsilon_1 \Rightarrow |\ell_p(f^*) - \ell_p(\hat{f})| \leq a \cdot M \cdot \epsilon_1$, for any path $p \in \mathcal{P}$. Choosing $\epsilon_1 = \frac{\epsilon}{2 \cdot a \cdot M}$ and considering any path p used by f^* we get $|L(f^*) - \ell_p(\hat{f})| \leq \frac{\epsilon}{2}$, where $L(f^*)$ is the common path cost at equilibrium f^*. This implies that all used paths' costs are $\frac{\epsilon}{2}$ close to $L(f^*)$ and thus ϵ close to each other. Additionally, for any unused path p' it holds $|\ell_{p'}(f^*) - \ell_{p'}(\hat{f})| \leq \frac{\epsilon}{2}$ and since $\ell_{p'}(f^*) \geq L(f^*)$ we get $\ell_{p'}(\hat{f}) \geq L(f^*) - \frac{\epsilon}{2}$. Thus, $\hat{f}$ is an ϵ-Nash flow.

Putting it all together, flow $\hat{f}$ uses $O(\frac{a^2 \cdot M^3}{\epsilon^2})$ paths, is an ϵ-Nash flow, and for any used path $p \in \mathcal{P}$: $\ell_p(\hat{f}) \leq L(f^*) + \frac{\epsilon}{2}$. Multiplying both parts by $\hat{f}_p$ (i.e., the amount of flow passing through path p in $\hat{f}$) we get $\hat{f}_p \cdot \ell_p(\hat{f}_p) \leq \hat{f}_p \cdot (L(f^*) + \frac{\epsilon}{2})$. Summing over all p that are being used by $\hat{f}$ we get $\sum_p \hat{f}_p \cdot \ell_p(\hat{f}_p) \leq \sum_p \hat{f}_p \cdot (L(f^*) + \frac{\epsilon}{2}) \Rightarrow RC(\hat{f}) \leq (L(f^*) + \epsilon) \cdot \sum_p \hat{f}_p = L(f^*) + \epsilon = RC(f^*) + \frac{\epsilon}{2}$.

Last thing to note is that there exists one such $\hat{f}$ that uses only facilities, say F', that are used in the optimal solution F^*. This is because f^* and thus $\hat{f}$ lies in the convex hull of $X' = \{x_p : p \in \mathcal{P}'\}$, i.e., the used by f^* paths that end up in a node of the network where a facility is opened under F^*. Thus, the facilities opening cost in order to route $\hat{f}$ does not exceed the cost of opening the facilities in F^* implying that in total the routing cost of $\hat{f}$ together with the opening cost of F' is $\leq C(F^*) + \frac{\epsilon}{2}$

Algorithm 1 exhaustively searches among all flows that may arise as k-combinations of paths in G. For each of them, it checks which facilities are opened and whether the flow is an ϵ-Nash flow. It returns the F for which the corresponding sum of routing and facilities cost is minimized, and a corresponding ϵ-Nash flow.

Algorithm 1: The approximation scheme

Input: FLSC Instance G and $\alpha, \epsilon > 0$
Output: Feasible solution F of G and ϵ-Nash flow with cost $C \leq C(F^*) + \frac{\epsilon}{2}$

M:= the maximum length among the paths of G;
$k := O(\frac{a^2 M^3}{\epsilon^2})$;
K:= the set of all different k-combinations of paths of G, with replacements allowed;
Initialize BestCost:=∞, $F :=$ any set of vertices, and $g :=$any flow;
For every combination in K **do**

- Let F' be the solution defined by the combination's paths;
- Check if the combination's flow f, sending $\frac{1}{k}$ units to each path, is an $\epsilon-$Nash flow for F';
- Check if the total routing and facilities cost C is $C \leq BestCost$;
- If the answer is yes for both conditions, $BestCost := C$, $F := F'$ and $g := f$;

return *F and g;*

Theorem 2. *For any FLSC instance with a-Lipschitz cost functions and any $\epsilon > 0$ we can find in time $|\mathcal{P}|^{O\left(\frac{a^2 \cdot M^3}{\epsilon^2}\right)} \cdot O\big(poly(n)\big)$ a feasible solution F and an ϵ-Nash flow (that routes to these facilities) with total cost at most $C(F^*) + \frac{\epsilon}{2}$, where M is an upper bound on the length of the paths*

Proof. By the discussion above we get that in one of the $|\mathcal{P}|^{O\left(\frac{a^2 \cdot M^3}{\epsilon^2}\right)}$ loops of the while loop we will find a feasible F satisfying the theorem. As for the running time, we need to compute the running time of checking whether we have an ϵ-Nash flow and whether we have better total cost.

To check whether we have an ϵ-Nash flow we first consider the network induced by the flow carrying edges, which is a DAG, and check if the longest and the shortest path costs, say c_{min} and c_{max}, differ by no more than ϵ, i.e., $c_{max} - c_{min} \leq \epsilon$. Then we consider the full network and check whether the shortest path cost is $\geq c_{max} - \epsilon$. These can be done in time $O\big(k \cdot poly(n)\big)$ and suffice to verify that we have an ϵ-Nash flow. Since $k \leq |\mathcal{P}|$ we get the $|\mathcal{P}|^{O\left(\frac{a^2 \cdot M^3}{\epsilon^2}\right)} \cdot O\big(poly(n)\big)$ running time.

4 Instances with Nonincreasing Functions

In this section we focus on *FLCC* instances on undirected networks with nonincreasing cost functions where additionally we require $x \cdot \ell_e(x)$ to be increasing and concave. For ease of presentation we will call such functions *good*. As we show, in the optimal solution demands do not split and thus our approach works also for unsplittable demand. Here, we assume a common opening cost $B_i = B$ for any facility i.

We start the section by revealing the structure of the optimal solution (Theorem 3), which is needed for our algorithm's analysis but will also be used to show that this *FLCC* variant generalizes the Cost-Distance problem, in the sense that solving the Cost-Distance problem reduces to solving *FLCC* for good cost functions (Theorem 4). Next we present an algorithm for approximately solving *FLCC* for good cost functions and we close the section with the algorithm's analysis, providing approximation guarantees (Theorem 5).

The Optimal Solution is a Forest A key property on which we base our algorithm is that there exists an optimal solution where the flows form a forest on the graph, i.e., there is an optimal solution that opens k facilities and the edges used by the flows do not form cycles and thus can be seen as k trees rooted in each one of the opened facilities. We prove this in the following lemmas.

Lemma 1. *In any FLCC instance with good cost functions, there exists an optimal flow where there are no directed flow carrying cycles, i.e., cycles in which all demand flows in one direction.*

Proof. Let f^* be an optimal flow. For any cycle C that carries positive flow in the same direction, we can decrease f^* on the edges of the cycle by $\min_{e \in C} f^*_e$ getting a feasible solution with cost no more than the initial cost, since the $x \cdot \ell_e(x)$'s are nondecreasing, and thus with optimal cost. Repeatedly we can eliminate all such cycles and get an optimal acyclic flow.

Lemma 2. *In any FLCC instance with good cost functions, there exists an optimal flow where demands do not split their flows once they have met.*

Proof. Suppose there is a node in an optimal solution where some demand arrives and then a portion x_1 of this demand is routed through path P_1 and another portion x_2 is routed through path P_2 where $P_1 \neq P_2$. We will show that we can get another optimal flow on the same facilities where x_1 and x_2 are routed on the same path. Recall that any optimal solution minimizes the sum $\sum_{e \in E} x_e(F) \cdot \ell_e(x_e(F)) + \sum_{i \in F} B_i$ and thus $\sum_{e \in E} x_e(F) \cdot l_e(x_e(F))$ itself should be minimum among all flows routing to the facilities in F.

If we treat x_1 and x_2 as variables the total cost that is being paid on the edges of path P_1 that are not in P_2 and the total cost that is being paid on the edges of path P_2 that are not in P_1 are respectively: $C_1(x_1) = \sum_{e \in P_1 \setminus P_2} (w'_e + x_1) \cdot l_e(w'_e + x_1)$ and $C_2(x_2) = \sum_{e \in P_2 \setminus P_1} (w'_e + x_2) \cdot l_e(w'_e + x_2)$ where w'_e is the rest of the demand that is being routed through e in the optimal solution at hand.

Since all parts of the sum (i.e., the $(w'_e + x_1) \cdot l_e(w'_e + x_1)$'s) are concave functions (with respect to x_1 or x_2) then also $C_1(x_1)$ and $C_2(x_2)$ are concave functions. Since we have assumed that the flow is optimal, if we take the first derivatives of $C_1(x_1)$ and $C_2(x_2)$ it should hold that $C'_1(x_1) = C'_2(x_2)$. If this was not the case and for example it was $C'_1(x_1) > C'_2(x_2)$ then we could move a small amount of demand from P_1 to P_2 and the total cost would drop.

Using $C'_1(x_1) = C'_2(x_2)$ and that $C_1(x_1)$ and $C_2(x_2)$ are concave functions, we get that for any $d \leq x_1$

$$\frac{C_1(x_1) - C_1(x_1 - d)}{d} \geq C'_1(x_1) = C'_2(x_2) \geq \frac{C_2(x_2 + d) - C_2(x_2)}{d}$$

which gives

$$C_1(x_1) + C_2(x_2) \geq C_1(x_1 - d) + C_2(x_2 + d)$$

implying, by setting $d = x_1$, that we can move all the demand x_1 from P_1 to P_2 without increasing the total cost, remaining at optimal routing cost. We can repeatedly do this for any split of demand to P_1 and P_2 ending up with a flow with optimal cost where the demands route their flows on single paths.

Lemma 3. *In any FLCC instance with good cost functions, there exists an optimal flow for which there are no cycles formed by the edges with positive flow.*

Proof. Let f^* be an acyclic optimal flow where the demands do not split, like the one guaranteed by Lemmas 1 and 2. Consider the flow carrying edges and, in order to reach a contradiction, assume that there is a cycle formed. Since f^* is acyclic there must be a node v where both of the edges of v send flow away from v, contradicting that in f^* the demands do not split.

From Lemmas 1-3 it follows that:

Theorem 3. *In any FLCC instance with good cost functions, there exists an optimal flow for which there are no cycles formed by the edges with positive flow and the demands use single paths and do not split once they have met.*

Next we show that *FLCC* generalizes the Cost-Distance Problem:

Theorem 4. *The Cost-Distance problem polynomially reduces to solving FLCC with good cost functions.*

Proof. Let $G(V, E)$ be the graph, S be the set of source nodes, t be the target node, $\{w_s\}_{s\in S}$ be the demands of the nodes in S and $\{c_e\}_{e\in E}$ and $\{l_e\}_{e\in E}$ be the costs and the lengths of the Cost-Distance problem instance, say I_{cd}, and recall we are searching for a subgraph G' under which the sum $\sum_{e\in E'} c_e + \sum_{s_i\in S} w_s \cdot l(s_i, t)$ is minimized, where $l(s_i, t)$ denotes the length of the shortest s_i-t path in G'.

To create the *FLCC* instance from the Cost-Distance instance I_{cd} we first note that in a solution of I_{cd} the total cost incurred by one edge under the corresponding flow f is $c(e) + l(e) \cdot \sum_{s:e\in P_s} w_s$ where P_s is the route chosen for demand $s \in S$ in f. We can achieve the same cost by using the following cost function in the *FLCC* instance we create:

$$\ell_e(x) = \begin{cases} \frac{c(e)}{x} + l(e) & \text{if } x_e \geq \min_{s\in S}(w_s) \\ \frac{c(e)}{\min_{s\in S}(w_s)} + l(e) & \text{if } x_e < \min_{s\in S}(w_s) \end{cases}$$

If $x_e = 0$ then $x_e \cdot \ell_e(x_e) = 0$. If $x_e \geq \min_{s\in S}(w_s)$ then $x_e \cdot \ell_e(x_e) = c(e) + x_e \cdot l(e)$. Note that for $x \geq \min_{s\in S}(w_s)$ these functions are good. For $x < \min_{s\in S}(w_s)$ we do not really care because no flow less than $\min_{s\in S}(w_s)$ can exist in the optimal solution (recall Theorem 3) and we just want $\ell_e(0)$ to be bounded so as to pay zero cost when no flow passes through e.

The instance of *FLCC* we create has the same graph $G(V, E)$ with cost functions as defined above, a sufficiently (polynomially) large common facility opening cost B, and set of source nodes the set S of I_{cd} augmented by node t, i.e., $S \cup \{t\}$. The demands for the nodes in S are as in I_{cd} and for node t is set equal to the sum of weights of all other demands, i.e., $w_t = \sum_{s\in S} w_s$. B is chosen sufficiently large so that we know that only one facility opens in the optimal solution, since opening a second facility would cost more than the routing costs.

Consider an optimal solution for the *FLCC* instance. If it opens the facility in a node v different from t then we may get a new optimal solution by moving the facility on t, where w_t is already served without any routing cost, and route all the w_i's to v and then to t using the path that t used to send its demand to v. This will not increase the cost of the solution since routing a demand of weight $\sum_{s\in S} w_s$ from t to v (the demand of t), is equal to routing a demand of weight $\sum_{s\in S} w_s$ from v to t (the demands of nodes in S). Thus we may always change the optimal solution of *FLCC* into one where the facility opens at t.

Given a solution of the *FLCC* instance that opens a facility on t, the solution we return for I_{cd} is the subgraph $G(V', E')$ used by the flow of the *FLCC* solution. The optimality of the solution is guaranteed by the fact that minimizing

$$\sum_{e \in E} x_e(F) \cdot l_e(x_e(F)) + \sum_{i \in F} B$$

reduces to minimizing

$$\sum_{e \in E} x_e(F) \cdot l_e(x_e(F)),$$

since only one facility opens, which, by recalling the definition of the cost functions and the discussion following it, is equal to

$$\sum_{e \in E'} [c(e) + x_e(F) \cdot l(e)],$$

which in turn, by rearranging terms, recalling that demands do not split and recalling that the facility is opened at t, is equal to

$$\sum_{e \in E'} c_e + \sum_{s_i \in S} w_s \cdot l(s_i, t),$$

i.e., the objective of the Cost-Distance problem.

The Algorithm An observation that simplifies our approach is that we may focus on the k-median variant of *FLCC*. In this variant there are no opening costs for the facilities, only routing costs, and we may open at most k facilities. Solving the k-median variant for all possible k values, solves the original problem since for any k we can compute the sum of the routing cost of the optimal solution with the original problem's opening cost, which for any k will be $k \cdot B$ (recall, we have a common cost $B_i = B$) and then compare these sums for the n different values of k in order to keep the optimal one.

Below we give an algorithm that takes as input an instance of *FLCC* and returns k facility points for the k-median version of *FLCC*. The algorithm is influenced by the algorithm proposed by Meyerson et al. [38] for the Cost-Distance problem which returns one target point (i.e., one facility, in our setting). However, their matching mechanism and analysis are tailored made to the fact that the total cost incurred by each edge is linear. We introduce a novel matching mechanism which in turn complicates the resulting analysis. For the algorithm we will need the following distance metric: $g(u, v, w) = \sum_{e \in P^w_{u,v}} w \cdot \ell_e(w)$ where $P^w_{u,v}$ is the closest path from u to v with respect to the distance metric $\ell_e(w)$.

To give a high level view of the algorithm, the algorithm works in phases. In every phase it pairs up the demands. For every such pair it routes both demands from their sources to a chosen meeting point, and then it randomly (in a biased way) routes them back to one of the two demand sources. For the next phase, every pair of paired demands is handled as one demand of volume equal to the sum of the two demands and source the randomly chosen source in the previous phase. The algorithm will terminate when some phase ends with k demands, and it will open facilities right on the sources of these demands.

Two important ingredients of the algorithm is how the pairing is done (steps 2(a) and 2(b) below) and how the common source for the merging demands is chosen (step 2(d)ii). The reasons for doing the pairing and the routing this way will become apparent in the next section that analyzes the algorithm. For step 2(b) the requirement for having nodes unmatched is there to avoid forcing to match nodes that are in different rooted trees of the optimal solution (in case these are of odd cardinality). Recall that the optimal solution can be seen as k trees rooted in each one of the opened facilities.

Approximation scheme for nonincreasing cost functions

1. Initialize $S_0 = S$, $w_{0,s} = w_s$ and $i = 0$.

2. While $|S_i| > k$:
   (a) For every pair $u, v \in S_i$ find $K_i(u,v) = \min_{z \in V}\{g(u,z,w_{i,u}) + g(v,z,w_{i,v}) + \frac{w_{i,u}}{w_{i,u}+w_{i,v}} \cdot g(z,u,w_{i,u}+w_{i,v}) + \frac{w_{i,v}}{w_{i,u}+w_{i,v}} \cdot g(z,v,w_{i,u}+w_{i,v})\}$.
   (b) Perform a maximum cardinality minimum cost matching on S_i with respect to the costs K_i under the constraint that, unless you get an empty matching, k nodes must be unmatched.
   (c) Set $S_{i+1} = \{\}$.
   (d) For each matched u, v pair:
      i. Send both demands to the node z that minimized the expression of step 2(a).
      ii. Choose u with probability $\frac{w_{i,u}}{w_{i,u}+w_{i,v}}$, otherwise chose v. Without loss of generality we will assume that we chose u.
      iii. Send the combined $w_{i,u} + w_{i,v}$ demand back to u, add u to S_{i+1} and set $w_{i+1,u} = w_{i,u} + w_{i,v}$.
   (e) Add unmatched nodes to S_{i+1}
   (f) Set $i \leftarrow i + 1$
3. Return as facilities the k nodes that are in S_i and as flows the ones dictated by the above procedure.

The Analysis For our analysis we introduce the metric C_u^i to be the total cost payed due to the movement of demand from source u in phase i. The total cost payed in phase i is $C^i = \sum_{s \in S_i} C_s^i$. Since at every step the number of sources minus k is roughly divided by two and we end up with k (final) sources, it is not difficult to show the following lemma.

Lemma 4. *The algorithm presented terminates after $O(\log |S|)$ phases.*

If we prove that in each phase of the algorithm the expected cost payed (i.e. $\mathbf{E}[C^i]$ is at most the cost of the optimal solution, then combining this fact with Lemma 4 shows that our algorithm is an $O(\log |S|)$-approximate algorithm for the *FLCC* with good functions. We will show that this is true for the first phase and that the expected cost of each phase is less than or equal to the cost of the first phase.

Phase 0 Phase 0 is the first phase of our algorithm where no demands have been aggregated yet. We begin with the following lemma which is a generalization of [38, Lemma 4.2] and reveals some structural property of the optimal solution. Recall, again, that the optimal solution can be seen as k trees rooted in each one of the opened facilities.

Lemma 5. *Let $S' \subseteq S$ be a set of sources that are routed at a facility in node r in the optimal solution. Lets call T the tree that corresponds to the flows from nodes in S' to r. Then there exists a matching like the one done in steps 2(a),(b) of our algorithm where demands only use edges that they use in the optimal flow and at most one source is left unmatched.*

Proof. Intuitively, we perform steps 2(a),(b) conditioned that the paths for computing the $g(\cdot,\cdot,\cdot)$'s are restricted to belong in T. For every $s \in S'$ we take its path towards r at T until it meets with the path of another source. We will call the vertex in which the two paths merge a level one meeting point. We move all sources at their corresponding level one meeting points. In each meeting point with an even number of demands we match them arbitrarily until no demand is left unmatched. In each meeting point with an odd number of demands we do the same thing however now there will be one demand left out. We continue along the path of those level one meeting points with an unmatched demand until their path merges with the path of another level one meeting point. The vertices in which those level one paths meet are called level two meeting points. We continue along the same line of thought until we reach our final meeting point r where at most one demand will be left unmatched.

In the algorithm in steps 2(a)-(b) we search for the best (w.r.t. cost) such matching and we choose as z (in step 2(d)i) the respective meeting point described by the proof of Lemma 5. Thus, the returned matching cannot be worse than the one dictated by Lemma 5 and it will be without loss of generality that in this phase but also in every other phase we can assume that the edges used are exactly those found by the matching (we will route the same demand on paths at most as expensive, in total, as those of the optimal solution). To bound the cost of our matching we just need to bound the cost of sending demands to those meeting points and the expected demand of sending them back again. The first half is accomplished with the following lemma.

Lemma 6. *The cost of sending all demands to their respective meeting points is less than that of the optimal solution.*

Proof. From Lemma 5 there is a subtle implication that becomes very useful right now. Each edge is traversed by demands that traverse that edge in the optimal solution. More specifically suppose that an edge $e \in E$ is traversed by a set S'' of demands. In our matching only one of this demands traverses that edge. Thus the amount of flow f'_e traversing edge e in our matching is less or equal than the amount of flow f_e that traverses e in the optimal solution. And since $x \cdot \ell_e(x)$ is nondecreasing we have that $f'_e \cdot \ell_e(f'_e) \leq f_e \cdot \ell_e(f_e)$. Summing over all edges we get that the total cost is less than the optimal cost.

The next Lemma holds for any phase i of the algorithm.

Lemma 7. *The expected cost of routing demands back from their meeting point to the selected source is less than or equal to the cost paid for sending them to the meeting point.*

Proof. Suppose we are at phase i and let $a_1, a_2, \ldots, a_m$ be the edges along the path of u to the meeting point and $b_1, b_2, \ldots, b_n$ the ones of v. Then the expected cost $\mathbf{E}[C_{\text{back}}]$ of sending them back to a source is the cost of sending them to u times the probability of choosing u plus the probability of sending them to v times the cost of sending them there. In other words we have that

$$\mathbf{E}[C_{\text{back}}] = (w_{i,u}+w_{i,v}) \cdot \frac{w_{i,u}}{w_{i,u}+w_{i,v}} \cdot \sum_{j=1}^{m} \ell_{a_j}(w_{i,u}+w_{i,v}) + (w_{i,u}+w_{i,v}) \cdot \frac{w_{i,v}}{w_{i,u}+w_{i,v}} \cdot \sum_{j=1}^{n} \ell_{b_j}(w_{i,u}+w_{i,v})$$

$$\leq w_{i,u} \cdot \sum_{j=1}^{m} \ell_{a_j}(w_{i,u}) + w_{i,v} \cdot \sum_{j=1}^{m} \ell_{b_j}(w_{i,v})$$

which is exactly the cost of sending those demands to the meeting point. The inequality holds since the demands are positive and the ℓ_e's are nonincreasing.

From Lemmas 6 and 7 we can see that $C^0 \leq 2 \cdot C^*$ where C^* is the total cost of the optimal solution.

Phase i The only thing left to do is to bound the cost of sending demands to their meeting points in phase i of the algorithm. This can be done using the following lemma.

Lemma 8. *The expected cost of routing demands to their meeting points in phase i is less than the cost of the optimal solution.*

Proof. Suppose at phase i that in a node u a set S''_u of demands has been gathered with a total demand $W_u = \sum_{s \in S''_u} w_s$. The probability of u having this demand is $\frac{w_u}{W_u}$. This claim is easy to see because for u to have that demand it must have been selected in all phases with a total probability of

$$\frac{w_u}{w_u + w_{v_1}} \cdot \frac{w_u + w_{v_1}}{w_u + w_{v_1} + w_{v_2}} \cdot \ldots \cdot \frac{\sum_{s \in S''_u} w_s - w_{u_{last}}}{\sum_{s \in S''_u} w_s}$$

and in this product only the first numerator and the last denominator do not cancel out. We once again perform the matching described by Lemma 5 with respect to weights $w_{i,u}$.

We are now going to show that the expected cost of the matching described by Lemma 5 and searched for in steps 2(a)-(b) is bounded by the cost of the optimal solution. For the sake of the argument lets call $g_e(x) = x \cdot \ell_e(x)$ the cost payed for the usage of edge e (recall $g_e(x)$ is a nondecreasing concave function). Suppose that a specific edge e in the optimal solution is used by demands $w_1, w_2, \ldots, w_k$. Thus the total cost payed for edge e in the optimal solution is $g_e\left(\sum_{j=1}^{k} w_j\right)$. We will show that for this arbitrary e it is $\mathbf{E}[C_{i,e}] \leq g_e\left(\sum_{j=1}^{k} w_j\right)$, where $C_{i,e}$ is the cost payed in the matching of phase i by e. Summing over all e will conclude the proof.

In phase i the expected cost payed by edge e because of demand w_j is the cost payed because a total demand of weight W_{w_j} passes through e times the probability that all of the demand with which w_j has been matched, actually passes through e. That probability can be expressed as the probability of W_{w_j} ending up in w_j (which we have shown earlier that is equal to $\frac{w_j}{W_{w_j}}$), times the probability that demand is not matched earlier in the Tree. We will call the later probability p_{e,w_j}. Thus the expected cost payed by edge e due to demand w_j is $\frac{w_j}{W_{w_j}} \cdot p_{e,w_j} \cdot g_e(W_{w_j})$. Also we will call $p_{e,0}$ the probability that no demand passes through e in our matching either because with probability $\prod_{j=1}^{k}\left(1 - \frac{w_j}{W_{w_j}}\right)$ there are no demands in the subtree underneath e or there was an even number of demands in the subtree underneath e and thus all demands have been matched lower in the Tree. So the total expected cost of edge e is $p_{e,0} \cdot g_e(0) + \sum_{j=1}^{k} \frac{w_j}{W_{w_j}} \cdot p_{e,w_j} \cdot g_e(W_{w_j})$. However, one can see that $p_{e,0} + \sum_{j=1}^{k} \frac{w_j}{W_{w_j}} \cdot p_{e,w_j} = 1$ and $g_e(x)$ is concave, so we can use Jensen's inequality [30] for concave functions which leads to the following expression:

$$\mathbf{E}[C_{i,e}] \leq p_{e,0} \cdot g_e(0) + \sum_{j=1}^{k} \frac{w_j}{W_{w_j}} \cdot p_{e,w_j} \cdot g_e(W_{w_j})$$

$$\leq g_e\left(p_{e,0} \cdot 0 + \sum_{j=1}^{k} \frac{w_j}{W_{w_j}} \cdot p_{e,w_j} \cdot W_{w_j}\right) = g_e\left(\sum_{j=1}^{k} w_j \cdot p_{e,w_j}\right) \leq g_e\left(\sum_{j=1}^{k} w_j\right)$$

where the last inequality holds because $p_{e,w_j} \leq 1$ and $g_e(x)$ is nondecreasing. This argument concludes our proof.

Combining the aforementioned lemmas we end up with the following lemma.

Lemma 9. *The routing cost of each phase is on expectation at most the cost of the optimal solution.*

Combining this lemma with the fact that our algorithm has $O(\log |S|)$ phases we get the following theorem.

Theorem 5. *The analyzed algorithm produces an $O(\log |S|)$-approximate solution to FLCC with good cost functions.*

The solution returned by the approximation scheme for nonincreasing cost functions can be used for approximately solving *FLSC*. Under the opened set of facilities, say $\hat{F}$, the routing cost when considering *FLSC*, i.e., when requiring agents to be at equilibrium, is at most $PoA(D)$ times the optimal routing cost. Here, $PoA(D)$ is the Price of Anarchy [33] for cost functions in class D [17], which captures how much worse can an equilibrium flow be when compared to the optimal flow for the cost functions used.

Thus, the total cost of the solution that opens facility set $\hat{F}$ when considering *FLSC* is at most $PoA(D)$ times the total cost of the solution when considering *FLCC*, which in turn is at most $O(\log |S|)$ times the

cost of the optimal solution for *FLCC*. On the other, hand the optimal solution for *FLCC* has total cost at most as the total cost of the optimal solution for *FLSC*. Putting it all together we get the following corollary. We note that our scope is not to provide a PoA(D) bound, rather, it is to provide an approximation scheme for when such a bound is known.

Corollary 1. *The approximation scheme presented outputs an* $[O(\log |S|) \cdot PoA(D)]$*-approximate solution to FLSC with good cost functions.*

5 Conclusion and the Case of Discrete Agents

We introduced Facility Location problems that account for congestion dependent connection costs. Among the many possible variants, we presented algorithms for two of them. Providing results for any of these variants is an interesting direction.

Our results can be of use if one considers discrete agents. One can use the algorithm in Section 3, that cuts the flow in small discrete particles, to solve the corresponding continuous case and then apply some type of rounding to get approximation guarantees. These guarantees will be close to the ones of the continuous case when the number of agents is large enough to allow for small enough cuts in the flow. In Section 4 we showed that in the optimal solution the demands do not split. Thus, the algorithm presented here works also for the discrete agents' case. Last, derandomizing this algorithm could be an interesting extension.

Bibliography

[1] Karen Aardal, Fabián A. Chudak, and David B. Shmoys. A 3-approximation algorithm for the k-level uncapacitated facility location problem. *Information Processing Letters*, 72(5-6):161–167, 1999.

[2] Eric Angel, Nguyen Thang, and Damien Regnault. Improved local search for universal facility location. *Journal of Combinatorial Optimization*, 29, 02 2014.

[3] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. *SIAM Journal on Computing*, 33(3):544–562, 2004.

[4] Manisha Bansal, Naveen Garg, and Neelima Gupta. A 5-approximation for universal facility location. *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, 12 2018.

[5] Siddharth Barman. Approximating nash equilibria and dense subgraphs via an approximate version of carathéodory's theorem. *SIAM J. Comput.*, 47(3):960–981, 2018.

[6] Opher Baron, Oded Berman, and Dmitry Krass. Facility location with stochastic demand and constraints on waiting time. *Manuf. Serv. Oper. Manag.*, 10(3):484–505, 2008.

[7] Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. *Proceedings of 37th Conference on Foundations of Computer Science*, 1996.

[8] Yair Bartal. On approximating arbitrary metrices by tree metrics. *Proceedings of the thirtieth annual ACM symposium on Theory of computing - STOC 98*, 1998.

[9] D. Braess. Über ein paradox aus der Verkehrsplanung. *Unternehmensforschung*, 12:258–268, 1968.

[10] Arghya Chakraborty and Rahul Vaze. Online facility location with timed-requests and congestion, 11 2022.

[11] Moses Charikar, Chandra Chekuri, Ashish Goel, and Sudipto Guha. Rounding via trees: Deterministic approximation algorithms for group steiner trees and k-median. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, page 114–123, New York, NY, USA, 1998. Association for Computing Machinery.

[12] Moses Charikar, Sudipto Guha, Éva Tardos, and David B. Shmoys. A constant-factor approximation algorithm for the k-median problem. *Journal of Computer and System Sciences*, 65(1):129–149, 2002.

[13] Chandra Chekuri, Sanjeev Khanna, and Joseph Naor. A deterministic algorithm for the cost-distance problem. *Symposium on Discrete Algorithms: Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, 7(09):232–233, 2001.

[14] Fabián A. Chudak and David B. Shmoys. Improved approximation algorithms for the uncapacitated facility location problem. *SIAM Journal on Computing*, 33(1):1–25, 2003.

[15] Fan Chung and Stephen J. Young. Braess's paradox in large sparse graphs. In *Internet and Network Economics*, pages 194–208, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[16] Julia Chuzhoy, Anupam Gupta, Joseph (Seffi) Naor, and mitabh Sinha. On the approximability of some network design problems. *ACM Trans. Algorithms*, 4(2), May 2008.

[17] José R. Correa, Andreas S. Schulz, and Nicolás E. Stier Moses. Selfish routing in capacitated networks. *Math. Oper. Res.*, 29(4):961–976, 2004.

[18] Teodora Dan and Patrice Marcotte. Competitive facility location with selfish users and queues. *Oper. Res.*, 67(2):479–497, 2019.

[19] Sotirios Dimos, Dimitris Fotakis, Thanasis Lianeas, and Kyriakos Sergis. Escaping braess's paradox through approximate caratheodory's theorem. *Inf. Process. Lett.*, 179:106289, 2023.

[20] Amir Eshaghi, Hamed Jahani, Abdollah Aghaie, and Dmitry Ivanov. A multi-layer congested facility location problem with consideration of impatient customers in a queuing system. *IFAC-PapersOnLine*, 52:2279–2284, 01 2019.

[21] Dimitris Fotakis, Alexis C. Kaporis, and Paul G. Spirakis. Efficient methods for selfish network design. *Theoretical Computer Science*, 448:9–20, 2012.

[22] S. Guha, A. Meyerson, and K. Munagala. Hierarchical placement and network design problems. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, FOCS '00, page 603, USA, 2000. IEEE Computer Society.

[23] Sudipto Guha and Samir Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithms*, 31(1):228–248, 1999.

[24] Anupam Gupta, Jon Kleinberg, Amit Kumar, Rajeev Rastogi, and Bulent Yener. Provisioning a virtual private network: A network design problem for multicommodity flow. *33rd Proc. STOC*, pages 389–398, 01 2001.

[25] Anupam Gupta, Amit Kumar, and Tim Roughgarden. Simpler and better approximation algorithms for network design. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '03, page 365–372, New York, NY, USA, 2003. Association for Computing Machinery.

[26] Mohammad Taghi Hajiaghayi, Mohammad Mahdian, and Vahab S. Mirrokni. The facility location problem with general cost functions. *Networks*, 42(1):42–47, 2003.

[27] Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing lp. *Journal of the ACM*, 50(6):795–824, 2003.

[28] Kamal Jain and Vijay Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *Journal of The ACM - JACM*, 48, 03 2001.

[29] Ata Jalili Marand and Pooya Hoseinpour. A congested facility location problem with strategic customers. *European Journal of Operational Research*, 318(2):442–456, 2024.

[30] J. L. W. V. Jensen. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta Mathematica*, 30:175–193, 1906.

[31] David Karger and Michael Minkoff. Building steiner trees with incomplete global knowledge. *Annual Symposium on Foundations of Computer Science - Proceedings*, pages 613–623, 02 2000.

[32] F. Kelly. *The mathematics of traffic in networks*. In *The Princeton Companion to Mathematics* (Editors: T. Gowers, J. Green and I. Leader). Princeton University Press, 2008.

[33] Elias Koutsoupias and Christos Papadimitriou. Worst-case equilibria. *16th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 404– 413, 1999.

[34] Shi Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. *Information and Computation*, 222:45–58, 2013. 38th International Colloquium on Automata, Languages and Programming (ICALP 2011).

[35] Mohammad Mahdian and Martin Pal. Universal facility location. *Springer, Berlin, Heidelberg*, 2003.

[36] Vladimir Marianov, Miguel Ríos, and Manuel José Icaza. Facility location for market capture when users rank facilities by shorter travel and waiting times. *European Journal of Operational Research*, 191(1):32–44, 2008.

[37] A Meyerson, K Munagala, and S Plotkin. Cost-distance: two metric network design. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 624–624. IEEE Computer Society, 2000.

[38] Adam Meyerson, Kamesh Munagala, and Serge A. Plotkin. Cost-distance: Two metric network design. *SIAM J. Comput.*, 38(4):1648–1659, 2008.

[39] J. D. Murchland. Braess's paradox of traffic flow. *Transportation Res.*, 4:391–394, 1970.

[40] Vinayaka Pandit. *Local Search Heuristics For Facility Location Problems*. PhD thesis, Department of Computer Science and Engineering Indian Institute of Technology Delhi, 2004.

[41] Masoud Hekmatfar Reza Zanjirani Farahani. *Facility Location. Concepts, Models, Algorithms and Case Studies*. Springer-Verlag Berlin, 2009.

[42] Robert W. Rosenthal. A class of games possessing pure-strategy nash equilibria. *International Journal of Game Theory*, 7, 12 1973.
[43] T. Roughgarden. *Selfish Routing and the Price of Anarchy*. MIT press, 2005.
[44] Tim Roughgarden. On the severity of braess's paradox: Designing networks for selfish users is hard. *Journal of Computer and System Sciences*, 72(5):922 – 953, 2006. Special Issue on FOCS 2001.
[45] Mehdi Seifbarghy and Aida Mansouri. Modelling and solving a congested facility location problem considering systems' and customers' objectives. *International Journal of Industrial and Systems Engineering*, 22:281, 01 2016.
[46] D. B. Shmoys, E. Tardos, and K.I. Aardal. Approximation algorithms for facility location problems. In *Proceedings of 29th Annual ACM Symposium on Theory of Computing*, pages 265–274, 1997.
[47] Chaitanya Swamy and Amit Kumar. Primal–dual algorithms for connected facility location problems. *Algorithmica*, 40:245–269, 01 2004.
[48] Gregory Valiant and Tim Roughgarden. Braess's paradox in large random graphs. *Random Struct. Algorithms*, 37(4):495–515, 2010.
[49] Jens Vygen. From stars to comets: Improved local search for universal facility location. *Oper. Res. Lett.*, 35:427–433, 07 2007.
[50] Xin Wang, Jiemin Zhao, Chun Cheng, and Mingyao Qi. A multi-objective fuzzy facility location problem with congestion and priority for drone-based emergency deliveries. *Computers & Industrial Engineering*, 179:109167, 2023.
[51] Horst W. Hamacher Zvi Drezner. *Facility Location. Applications and Theory*. Springer-Verlag Berlin, 2001.

A On the Applicability of Facility Location Techniques

Why Local Search does not Work in our Setting

One natural idea would be to try and solve this problem using local search algorithms. Consider the following example for instances where the cost function of an edge is a polynomial of bounded degree d. Since in our problem we have constant facility costs we are going to examine local steps that do not take into consideration the amount of demand each facility accommodates. We are going to use the **open** local step where one facility is opened and the demand is rerouted optimally, the **close** local step where an open facility is closed and the demand is rerouted optimally and the **swap** where an opened facility is closed while a closed facility is opened simultaneously and the demand is the rerouted optimally.

In the example illustrated in figure 1 we have nodes $c_1, c_2, ..., c_k$ all having unit demand. The cost of opening a facility on o_i is ϵ and the cost functions of edges (c_i, o_i) are constant and equal to 1. We also have another possible facility node S with cost of opening a facility $(k-1)^{d+1}$. The cost functions of edges $e = (c_i, S)$ is $l_e(x) = x^d$. The optimal solution is opening all o_i facilities with cost $k \cdot \epsilon$ and route each demand c_j to its corresponding facility o_j. Now consider the following locally optimal solution where only S is open. Obviously we cannot close S. We cannot open o_i because it would not change the routing cost and just add an extra ϵ to the facility cost. Also, we cannot swap S with o_i. If we did this then the facility cost would be just ϵ but the routing cost would be $3 \cdot (k-1) + 1 + (k-1) \cdot (k-1)^d$ since all demand need to be routed at o_i. Thus having S as our only facility is a locally optimal solution with a local ratio of $O(k^d)$.

To get more intuition on the difficulty of this approach, following the techniques in the related literature that are usually employed to provide approximation guarantees, we would state a list of local steps and try to find out information concerning a locally optimal solution. We will have to use the fact that no local step can improve the cost of the locally optimal state in order to bound its cost. However we need good estimations concerning the new routing costs once a local step is made. Since the edge cost functions depend on the

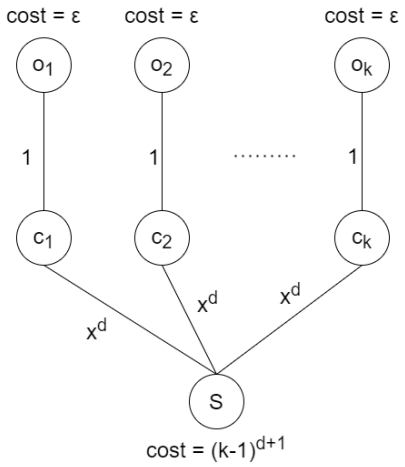

Fig. 1: A counter example for the use of Local Search

amount of traffic they receive, then we cannot come up with an estimate concerning the routing cost while taking into consideration only demands that are being affected due to the local step. In other words, in order to estimate the increase or decrease in the total routing cost we need to take into account all of the demand that have passed through each edge up until we reached our final state. This fact increases the complexity and new techniques must be created in order to tackle this problem.

Why Linear Programming does not Work in our Setting

A second natural idea would be to find some underlying structure of the optimal solution in order to come up with an LP-rounding or primal-dual algorithm. To do so in this section we will try and simplify our problem. We will assume that the facility cost is the same for all nodes and equal to B (i.e. $f_j = B, \forall j \in V$). We will also assume that our cost functions are linear functions with no negative coefficients (i.e. $l_e(x) = a \cdot x + b$ where $a, b \geq 0, \forall e \in E$).

Consider the following natural formulation of the problem:

$$min \sum_{e \in E} (a_e \cdot x_e^2 + b_e \cdot x_e) + \sum_{e \in F} z_e \cdot B$$

$$\begin{aligned} s.t. \sum_{e \in \delta^-(v)} x_e = \sum_{e \in \delta^+(v)} x_e, \quad & \forall v \in V \\ x_e \leq z_e \cdot \sum_{s \in S} w_s, & \forall e \in F \\ x_e = w_s & \forall e \in S \\ z_e \in \{0, 1\} & \forall e \in F \end{aligned}$$

In the above formulation we treat demands as edges that lead into the graph and carry already w_s demand, and facilities as edges that leave from the graph. The variable z_e becomes 1 when the corresponding facility is opened and 0 otherwise. Thus the second inequality of our restrictions ensures that only open facilities are being used. This linear program has an integer restriction and thus, a priori, cannot be solved in polynomial time (unless P=NP). However this restriction is essential to the formulation. Consider the relaxation where we just demand that $0 \leq z_e \leq 1$. This problem can be solved in polynomial time however its optimal solution offer no valuable information because of the following observation.

When relaxing the integrality constraint on z_e, since we have a minimization problem z_e will receive the smallest possible value and thus we will have $x_e = z_e \cdot \sum_{s \in S} w_s \Rightarrow z_e = \frac{x_e}{\sum_{s \in S} w_s} \Rightarrow \sum_{e \in F} z_e \cdot B = \sum_{e \in F} \frac{x_e}{\sum_{s \in S} w_s} \cdot B = \frac{B}{\sum_{s \in S} w_s} \cdot \sum_{e \in F} x_e$. However all demands must be served by a facility so $\sum_{e \in F} x_e =$

$\sum_{s\in S} w_s \Rightarrow \frac{B}{\sum_{s\in S} w_s} \cdot \sum_{e\in F} x_e = B$. This tells us that no mater how many facilities are opened and no matter how the demand will be arranged to them, the facility cost will always be equal to B. So in order to minimize the routing cost a fractional facility will be opened at every demand bringing the routing cost down to zero. Since the fractional problem returns always the same solution with the same cost, it is of no use for a better approximation than the naive. In other words the intergrality gap of this non-linear program is $O(n)$. Our problem in general becomes really easy when intergrality constraints are removed and thus LP based solutions have big integrality gaps and thus the inherently cannot provide valuable information.

B Non-extendability of the Sparsification Technique to the Multi-Source Setting.

Suppose we have multiple demand points and the cost of opening a facility is common and equals B. First of all we cannot try to use sparsification techniques for each demand point separately because this would lead us to an exhaustive search that is exponential in the amount of demand points. However its is obvious that the number of facilities is bounded by the number of demands. So if we could perform exhaustive search exponential in the amount of demands we could simply check all possible facility combination in the first place thus rendering the use of sparsification techniques useless.

Thus it becomes evident that we have to treat the flow as a whole in order for sparsification techniques to add value to our analysis. However in doing so the results returned by the exhaustive search might not be a feasible solution of the problem. It might also not take into consideration many small demands and close facilities that are crucial to the optimal solution. So once we have used those techniques we then have to find set of facilities that will accommodate large amount of demands which was the exact problem we began with. To better illustrate this argument consider the following example that captures the essence of the difficulty of the problem.

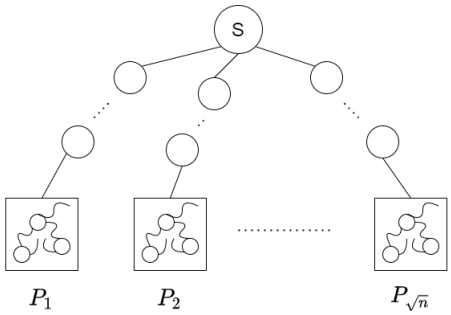

Fig. 2: An example where sparsification techniques are rendered useless

If we do not use sparsification techniques the best algorithm we are aware of is the naive $O(n)$-approximate algorithm. Consider the example depicted in figure 2. For any n we can create $\sqrt{n}$ different problems where each one of them has $\sqrt{n}$ nodes and $\frac{r}{\sqrt{n}}$ demand (where r is the total demand of the problem). We will call these problems $P_1, P_2, ..., P_{\sqrt{n}}$. Suppose that each problem P_j has C_j^* cost in its optimal solution. We also create a node S and we connect it with each problem with a long chain of nodes. Due to the restrictions of sparsification techniques those chains can be of length at most $b_1 \cdot \log^{b_2}(n)$ where b_1, b_2 are constants. This ensures that the optimal solution of the final graph is to solve optimally each problem separately. So the total number of nodes is $\sqrt{n} \cdot \sqrt{n} + \sqrt{n} \cdot b_1 \cdot \log^{b_2}(n) + 1 = O(n)$. If we tried to solve each problem separately the best we could do without the use of sparsification techniques would be to use our naive approximate algorithm and end up with a total cost of $\sqrt{n} * OPT$.

Now we will examine how well sparsification techniques perform in this general problem. Sparsification techniques will examine at most k paths and thus will open at most k facilities. For the exhaustive search of the sparsification techniques to end in quasi-polynomial time we want $k = \textit{poly}(\log(n))$. Thus the sparsification technique will yield a result that addresses at most k from the $\sqrt{n}$ problems.

If we do not open any more facilities then we will have to route demand from $\sqrt{n} - k$ facilities to S and then to the k addressed problems. This obviously creates a cost that is greater than that of the naive algorithm of opening just one facility in S and routing all demands there. So it becomes apparent that we have to open new facilities.

The only possible alternative is to solve each of the $\sqrt{n} - k$ unaddressed problems separately using our naive $O(n)$-approximation algorithm. Without loss of generality suppose that $P_1, P_2, ..., P_{\sqrt{n}-k}$ are the unaddressed problems. Solving them naively we get $\sqrt{n}\cdot C_1^* + \sqrt{n}\cdot C_2^* + ... + \sqrt{n}\cdot C_{\sqrt{n}-k}^* = \sqrt{n}\cdot\sum_{j=1}^{\sqrt{n}-k} C_j^*$ cost. We will try to find the approximation ratio of our solution. Obviously we can bound from underneath the approximation ratio if we make the following assumptions:

- The k addressed problems are solved optimally.
- The cost of the k addressed problems is the maximum possible.
- The cost of the $\sqrt{n} - k$ unaddressed problems is the minimum possible.

We can observe that each problem must open at least one facility and opening one facility in each node is a possible solution (recall that B is the cost of opening a facility). Thus $B \leq C_j^* \leq \sqrt{n} \cdot B$ for all j. Thus we get that the approximation ratio of our algorithm is:

$$A = \frac{\sqrt{n} \cdot \sum_{j=1}^{\sqrt{n}-k} C_j^* + \sum_{j=\sqrt{n}-k+1}^{\sqrt{n}} C_j^*}{\sum_{j=1}^{\sqrt{n}-k} C_j^* + \sum_{j=\sqrt{n}-k+1}^{\sqrt{n}} C_j^*} \geq \frac{\sqrt{n} \cdot \sum_{j=1}^{\sqrt{n}-k} B + \sum_{j=\sqrt{n}-k+1}^{\sqrt{n}} \sqrt{n} \cdot B}{\sum_{j=1}^{\sqrt{n}-k} B + \sum_{j=\sqrt{n}-k+1}^{\sqrt{n}} \sqrt{n} \cdot B}$$

$$\Rightarrow A \geq \frac{\sqrt{n} \cdot (\sqrt{n} - k) + \sqrt{n} \cdot k}{\sqrt{n} - k + \sqrt{n} \cdot k} = \frac{\sqrt{n}}{1 - k/\sqrt{n} + k} \geq \frac{\sqrt{n}}{1 + k}$$

Thus $A \geq \frac{\sqrt{n}}{1+k} \geq O(n^{\frac{1}{2}-\lambda})$ for any $\lambda > 0$. The second inequality holds since $k = \textit{poly}(\log(n))$. So, especially for large n we have not made any significant progress regarding the $O(\sqrt{n})$ naive algorithm we could have implemented in the first place. Also it is not obvious how we will distinguish those different problems in more difficult settings. Additionally with some hyper parameter tuning such as adjusting the total number of different problems and pose some restraints on the cost of the optimal solution of each problem P_j we can achieve a tighter bound on A. Nevertheless this analysis is simple enough to be easily understood but also perfectly illustrates our main point.