Local-Search based Approximation Algorithms for Mobile Facility Location Problems

(Extended Abstract)

Sara Ahmadian* Zachary Friggstad* Chaitanya Swamy*

Abstract

We consider the *mobile facility location* (MFL) problem. We are given a set of facilities and clients located in a common metric space $G = (V, c)$. The goal is to move each facility from its initial location to a destination (in V) and assign each client to the destination of some facility so as to minimize the sum of the movement-costs of the facilities and the client-assignment costs. This abstracts facility-location settings where one has the flexibility of moving facilities from their current locations to other destinations so as to serve clients more efficiently by reducing their assignment costs.

We give the first *local-search based* approximation algorithm for this problem and achieve the best-known approximation guarantee. Our main result is $(3 + \epsilon)$-approximation for this problem for any constant $\epsilon > 0$ using local search. The previous best guarantee for MFL was an 8-approximation algorithm due to Friggstad and Salavatipour [12] based on LP-rounding. Our guarantee *matches* the best-known approximation guarantee for the k-median problem. Since there is an approximation-preserving reduction from the k-median problem to MFL, any improvement of our result would imply an analogous improvement for the k-median problem. Furthermore, *our analysis is tight* (up to $o(1)$ factors) since the tight example for the local-search based 3-approximation algorithm for k-median can be easily adapted to show that our local-search algorithm has a tight approximation ratio of 3. Our results extend to the weighted generalization wherein each facility i has a non-negative weight w_i and the movement cost for i is w_i times the distance traveled by i.

In contrast to the k-median problem, the local search procedure that moves, at each step, a constant number of facilities (to chosen destinations) and assigns each client to the nearest destination, is known to have an unbounded locality gap. Our local-search algorithm is a natural and simple variant, where we only select the destinations of the facilities in each step and optimally rematch the facilities to these destinations (which might entail moving all facilities). One of the chief novelties in the analysis is that in order to generate a suitable collection of local-search moves whose resulting inequalities yield the desired bound on the cost of a local-optimum, we define a tree-like structure that (loosely speaking) functions as a "recursion tree", using which we spawn off local-search moves by exploring this tree to a constant depth.

1 Introduction

Facility location problems have been widely studied in the Operations Research and Computer Science communities (see, e.g., [23] and the survey [18]), and have a wide range of applications. In its simplest version, *uncapacitated facility location* (UFL), we are given a set of facilities or service-providers with opening costs, and a set of clients that require service, and we want to open some facilities and assign clients to open facilities so as to minimize the sum of the facility-opening and client-assignment costs. An oft-cited prototypical example is that of a company wanting to decide where to locate its warehouses/distribution centers so as to serve its customers in a cost-effective manner.

We consider facility-location problems that abstract settings where facilities are mobile and may be relocated to destinations near the clients in order to serve them more efficiently by reducing the client-assignment costs. More precisely, we consider the *mobile facility location* (MFL) problem introduced by [10, 12], which generalizes the classical k-median problem (see below). We are given a complete graph $G = (V, E_G)$ with costs $\{c(u, v)\}$ on the edges, a set $\mathcal{D} \subseteq V$ of clients with each client j having d_j units of demand, and a set $\mathcal{F} \subseteq V$ of k initial facility locations. We use the term facility i to denote the facility whose initial location is $i \in \mathcal{F}$. A solution S to MFL moves each facility i to a final location $s_i \in V$ (which could be the same as i), incurring a *movement*

*{sahmadian,zfriggstad,cswamy}@math.uwaterloo.ca. Dept. of Combinatorics and Optimization, Univ. Waterloo, Waterloo, ON N2L 3G1. Supported in part by NSERC grant 327620-09. The second and third authors are also supported by the third author's Ontario Early Researcher Award.

cost $c(i, s_i)$, and assigns each client j to a final location $s \in S$, incurring *assignment cost* $d_j c(j, s)$. The total cost of S is the sum of all the movement costs and assignment costs. More formally, noting that each client will be assigned to the location nearest to it in S, we can express the cost of S as

$$\mathsf{MFL}(S) := \sum_{i \in \mathcal{F}} c(i, s_i) + \sum_{j \in \mathcal{D}} d_j c(j, \sigma(j))$$

where $\sigma(v)$ (for any node v) gives the location in S nearest to v (breaking ties arbitrarily). We assume throughout that the edge costs form a metric. We use the terms nodes and locations interchangeably.

Mobile facility location falls into the genre of *movement problems* introduced by Demaine et al. [10]. In these problems, we are given an initial configuration in a weighted graph specified by placing "pebbles" on the nodes and/or edges; the goal is to move the pebbles so as to obtain a desired final configuration while minimizing the maximum, or total, pebble movement. MFL was introduced by Demaine et al. as the movement problem where facility- and client- pebbles are placed respectively at the initial locations of the facilities and the clients; in the final configuration every client-pebble should be co-located with some facility-pebble.

Our results. We give the first *local-search based* approximation algorithm for this problem and achieve the best-known approximation guarantee. Our main result is a $(3+\epsilon)$-approximation for this problem for any constant $\epsilon > 0$ using a simple local-search algorithm. This improves upon the previous best 8-approximation guarantee for MFL due to [12], which is based on LP-rounding and is not combinatorial.

The local-search algorithm we consider is quite natural and simple. Observe that given the final locations of the facilities, we can find the minimum-cost way of moving facilities from their initial locations to the final locations by solving a minimum-cost perfect-matching problem (and the client assignments are determined by the function σ defined above). Thus, we concentrate on determining a good set of final locations. In our local-search algorithm, at each step, we are allowed to swap in and swap out a fixed number (say p) of locations. Clearly, for any fixed p, we can find the best local move efficiently (since the cost of a set of final locations can be computed in polytime). Note that we do not impose any constraints on how the matching between the initial and final locations may change due to a local move. It is important to allow this flexibility, as it is known [12] that the local-search procedure that moves, at each step, a constant number of facilities to chosen destinations has an unbounded approximation ratio.

Our main contribution is a tight analysis of this local-search algorithm (Section 4). Our guarantee *matches* (up to $o(1)$ terms) the best-known approximation guarantee for the k-median problem. Since there is an approximation-preserving reduction from the k-median problem to MFL [12]—choose arbitrary initial facility locations and give each client a huge demand D—any improvement of our result would imply an analogous improvement for the k-median problem. (In this respect, our result is a noteworthy exception to the prevalent state of affairs for various other generalizations of UFL and k-median—e.g., the data placement problem [3], {matroid-, red-blue-} median [20, 15, 8, 5], k-facility-location [11, 14]—where the best approximation ratio for the problem is worse by a noticeable factor (compared to UFL or k-median); [13] is another exception.) Moreover, *our analysis is tight* (up to $o(1)$ factors) because by suitably setting D in the reduction of [12], we can ensure that our local-search algorithm for MFL coincides with the local-search algorithm for k-median in [2] which has a tight approximation ratio of 3.

We also consider a weighted generalization of the problem (Section 5), wherein each facility i has a weight w_i indicating the cost incurred per-unit distance moved and the cost for moving i to s_i is $w_i c(i, s_i)$. (This can be used to model, for example, the setting where different facilities move at different speeds.) Our analysis is versatile and extends to this weighted generalization to yield the same performance guarantee. For the further generalization of the problem, where the facility-movement costs may be arbitrary and unrelated to the client-assignment costs (for which a 9-approximation can be obtained via LP-rounding; see "Related work"), we show that local search based on multiple swaps has a bad approximation ratio (Appendix A).

Our techniques. The analysis of our local-search procedure requires various novel ideas. As is common in the analysis of local-search algorithms, we identify a set of test swaps and use local optimality to generate suitable inequalities from these test swaps, which when combined yield the stated performance guarantee. One of the difficulties involved in adapting standard local-search ideas to MFL is the following artifact: in MFL, the cost of "opening" a set S of locations is the cost of the min-cost perfect matching of $\mathcal{F}$ to S, which, unlike other facility-location problems, is a highly non-additive function of S (and as mentioned above, we need to allow for the matching from $\mathcal{F}$ to S to change in non-local ways). In most facility-location problems with opening costs for which local search is known to work, we may always swap in a facility used by the global optimum (by possibly swapping out another facility) and easily bound the resulting change in *facility cost*, and the main consideration is to decide how to reassign clients

following the swap in a cost-effective way; in MFL we do not have this flexibility and need to carefully choose how to swap facilities so as to ensure that there is a good matching of the facilities to their new destinations after a swap *and* there is a frugal reassignment of clients.

This leads us to consider long relocation paths to rematch facilities to their new destinations after a swap, which are of the form $(\ldots, s_i, o_i, s_{i'}, \ldots)$, where s_i and o_i are the locations that facility i is moved to in the local and global optimum, S and O, respectively, and $s_{i'}$ is the S-location closest to o_i. By considering a swap move involving the start and end locations of such a path Z, we can obtain a bound on the movement cost of all facilities $i \in Z$ where s_i is the start of the path or o_i serves a large number of clients. To account for the remaining facilities, we break up Z into suitable intervals, each containing a constant number of unaccounted locations which then participate in a multi-location swap. This *interval-swap* move does not at first appear to be useful since we can only bound the cost-change due to this move in terms of a significant multiple of (a portion of) the cost of the local optimum! One of the novelties of our analysis is to show how we can *amortize* the cost of such expensive terms and make their contribution negligible by considering multiple different ways of covering Z with intervals and averaging the inequalities obtained for these interval swaps. These ideas lead to the proof of an approximation ratio of 5 for the local-search algorithm (Section 3).

The tighter analysis leading to the 3-approximation guarantee (Section 4) features another noteworthy idea, namely that of using "recursion" (up to bounded depth) to identify a suitable collection of test swaps. We consider the tree-like structure created by the paths used in the 5-approximation analysis, and (loosely speaking) view this as a recursion tree, using which we spawn off interval-swap moves by exploring this tree to a constant depth. To our knowledge, we do not know of any analysis of a local-search algorithm that employs the idea of recursion to generate the set of test local moves (used to generate the inequalities that yield the desired performance guarantee). We believe that this technique is a notable contribution to the analysis of local-search algorithms that is of independent interest and will find further application.

Related work. As mentioned earlier, MFL was introduced by Demaine et al. [10] in the context of movement problems. Friggstad and Salavatipour [12] designed the first approximation algorithm for MFL. They gave an 8-approximation algorithm based on LP rounding by building upon the LP-rounding algorithm of Charikar et al. [7] for the k-median problem; this algorithm works only however for the unweighted case. They also observed that there is an approximation-preserving reduction from k-median to MFL.

Chakrabarty and Swamy [5] observed that MFL, even with arbitrary movement costs is a special case of the matroid median problem [20]. Thus, the 9-approximation algorithm devised for matroid median independently by [8] and [5], yields a 9-approximation algorithm for MFL with arbitrary movement costs.

There is a wealth of literature on approximation algorithms for (metric) uncapacitated and capacitated facility location (UFL and CFL), the k-median problem, and their variants; see [25] for a survey on UFL. Whereas constant-factor approximation algorithms for UFL and k-median can be obtained via a variety of techniques such as LP-rounding [26, 21, 7, 8], primal-dual methods [16, 17], local search [19, 6, 2], all known $O(1)$-approximation algorithms for CFL (in its full generality) are based on local search [19, 28, 4]. We now briefly survey the work on local-search algorithms for facility-location problems.

Starting with the work of [19], local-search techniques have been utilized to devise $O(1)$-approximation algorithms for various facility-location problems. Korupolu, Plaxton, and Rajaraman [19] devised $O(1)$-approximation for UFL, and CFL with uniform capacities, and k-median (with a blow-up in k). Charikar and Guha [6], and Arya et al. [2] both obtained a $(1+\sqrt{2})$-approximation for UFL. The first constant-factor approximation for CFL was obtained by Pál, Tardos, and Wexler [24], and after some improvements, the current-best approximation ratio now stands at $5+\epsilon$ [4]. For the special case of uniform capacities, the analysis in [19] was refined by [9], and Aggarwal et al. [1] obtain the current-best 3-approximation. Arya et al. [2] devised a $(3+\epsilon)$-approximation algorithm for k-median, which was also the first constant-factor approximation algorithm for this problem based on local search. Gupta and Tangwongsan [14] (among other results) simplified the analysis in [2]. We build upon some of their ideas in our analysis.

Local-search algorithms with constant approximation ratios have also been devised for various variants of the above three canonical problems. Mahdian and Pál [22], and Svitkina and Tardos [27] consider settings where the opening cost of a facility is a function of the set of clients served by it. In [22], this cost is a non-decreasing function of the number of clients, and in [27] this cost arises from a certain tree defined on the client set. Devanur et al. [11] and [14] consider k-facility location, which is similar to k-median except that facilities also have opening costs. Hajiaghayi et al. [15] consider a special case of the matroid median problem that they call the red-blue median problem. Most recently, [13]

considered a problem that they call the k-median forest problem, which generalizes k-median, and obtained a $(3+\epsilon)$-approximation algorithm.

2 The local-search algorithm

As mentioned earlier, to compute a solution to MFL, we only need to determine the set of final locations of the facilities, since we can then efficiently compute the best movement of facilities from their initial to final locations, and the client assignments. This motivates the following local-search operation. Given a current set S of $k = |\mathcal{F}|$ locations, we can move to any other set S' of k locations such that $|S \setminus S'| = |S' \setminus S| \leq p$, where p is some fixed value. We denote this move by swap$(S \setminus S', S' \setminus S)$. The local-search algorithm starts with an arbitrary set of k final locations. At each iteration, we choose the local-search move that yields the largest reduction in total cost and update our final-location set accordingly; if no cost-improving move exists, then we terminate. (To obtain polynomial running time, as is standard, we modify the above procedure so that we choose a local-search move only if the cost-reduction is at least ϵ(current cost).)

3 Analysis leading to a 5-approximation

We now analyze the above local-search algorithm and show that it is a $\big(5+o(1)\big)$-approximation algorithm. For notational simplicity, we assume that the local-search algorithm terminates at a local optimum; the modification to ensure polynomial running time degrades the approximation by at most a $(1+\epsilon)$-factor.

THEOREM 3.1. *Let F^* and C^* denote respectively the movement and assignment cost of an optimal solution. The total cost of any local optimum is at most $\big(3+o(1)\big)F^* + \big(5+o(1)\big)C^*$.*

Although this is not the tightest guarantee that we obtain, we present this analysis first since it introduces many of the ideas that we build upon in Section 4 to prove a tight approximation guarantee of $\big(3+o(1)\big)$ for the local-search algorithm. For notational simplicity, we assume that all d_j values are 1. All our analyses carry over trivially to the case of non-unit (integer) demands since we can think of a client j having d_j demand as d_j co-located unit-demand clients.

Notation and preliminaries We use $S = \{s_1, \ldots, s_k\}$ to denote the local optimum, where facility i is moved to final location $s_i \in S$. We use $O = \{o_1, \ldots, o_k\}$ to denote the (globally) optimal solution, where again facility i is moved to o_i. Throughout, we use s to index locations in S, and o to index locations in O. Recall that, for a node v, $\sigma(v)$ is the location in S nearest to v. Similarly, we define $\sigma^*(v)$ to be the location in O nearest to v. For notational similarity with facility location problems, we denote $c(i, s_i)$ by f_i, and $c(i, o_i)$ by f_i^*. (Thus, f_i and f_i^* are the movement costs of i in S and O respectively.) Also, we abbreviate $c\big(j, \sigma(j)\big)$ to c_j, and $c\big(j, \sigma^*(j)\big)$ to c_j^*. Thus, c_j and c_j^* are the assignment costs of j in the local and global optimum respectively. (So $\mathsf{MFL}(S) = \sum_{i\in\mathcal{F}} f_i + \sum_{j\in\mathcal{D}} c_j$.) Let $D(s) = \{j \in \mathcal{D} : \sigma(j) = s\}$ be the set of clients assigned to the location $s \in S$, and $D^*(o) = \{j \in \mathcal{D} : \sigma^*(j) = o\}$. For a set $A \subseteq S$, we define $D(A) = \bigcup_{s\in A} D(s)$; we define $D^*(A)$ for $A \subseteq O$ similarly. Define $\mathsf{cap}(s) = \{o \in O : \sigma(o) = s\}$. We say that s *captures* all the locations in $\mathsf{cap}(s)$. The following basic lemma will be used repeatedly.

LEMMA 3.1. *For any client j, we have $c\big(j, \sigma(\sigma^*(j))\big) - c\big(j, \sigma(j)\big) \leq 2c_j^*$.*

Proof. Let $s = \sigma(j)$, $o = \sigma^*(j)$, $s' = \sigma(o)$. The lemma clearly holds if $s' = s$. Otherwise, $c(j, s') - c(j, s) \leq c(j, o) + c(o, s') - c(j, s) \leq c_j^* + c(o, s) - c(j, s) \leq c_j^* + c(o, j) = 2c_j^*$ where the second inequality follows since s' is the closest location to o in S. ∎

To prove the approximation ratio, we will specify a set of local-search moves for the local optimum, and use the fact that none of these moves improve the cost to obtain some inequalities, which will together yield a bound on the cost of the local optimum. We describe these moves by using the following digraph. Consider the digraph $\widehat{G} = \big(\mathcal{F} \cup S \cup O, \{(s_i, i), (i, o_i), (o_i, \sigma(o_i))\}_{i\in\mathcal{F}}\big)$. We decompose $\widehat{G}$ into a collection of node-disjoint (simple) paths $\mathcal{P}$ and cycles $\mathcal{C}$ as follows. Repeatedly, while there is a cycle C in our current digraph, we add C to $\mathcal{C}$, remove all the nodes of C and recurse on the remaining digraph. After this step, a node v in the remaining digraph, which is acyclic, has: exactly one outgoing arc if $v \in S$; exactly one incoming and one outgoing arc if $v \in \mathcal{F}$; and exactly one incoming, and at most one outgoing arc if $v \in O$. Now we repeatedly choose a node $v \in S$ with no incoming arcs, include the maximal path P starting at v in $\mathcal{P}$, remove all nodes of P and recurse on the remaining digraph. Thus, each triple (s_i, i, o_i) is on a unique path or cycle in $\mathcal{P} \cup \mathcal{C}$. Define $\mathsf{center}(s)$ to be $o \in O$ such that (o, s) is an arc in $\mathcal{P} \cup \mathcal{C}$; if s has no incoming arc in $\mathcal{P} \cup \mathcal{C}$, then let $\mathsf{center}(s) = \mathsf{nil}$.

We will use $\mathcal{P}$ and $\mathcal{C}$ to define our swaps. For a path $P = (s_{i_1}, i_1, o_{i_1}, \ldots, s_{i_r}, i_r, o_{i_r}) \in \mathcal{P}$, define $\mathsf{start}(P)$ to be s_{i_1} and $\mathsf{end}(P)$ to be o_{i_r}. Notice that $\sigma(o_{i_r}) \notin P$. For each $s \in S$, let $\mathcal{P}_c(s) = \{P : \mathsf{end}(P) \in \mathsf{cap}(s)\}$, $T(s) = \{\mathsf{start}(P) : P \in \mathcal{P}_c(s)\}$, and $H(s) = \{\mathsf{end}(P) : P \in \mathcal{P}_c(s)\} = \mathsf{cap}(s) \setminus \mathsf{center}(s)$. Note that $|\mathcal{P}_c(s)| = |T(s)| = |H(s)| = |\mathsf{cap}(s)| - 1$ for any

$s \in S$ with $|\mathsf{cap}(s)| \geq 1$. For a set $A \subseteq S$, define $T(A) = \bigcup_{s\in A} T(s),\ H(A) = \bigcup_{s\in A} H(s),\ \mathcal{P}_c(A) = \bigcup_{s\in A} \mathcal{P}_c(s)$.

A basic building block in our analysis, involves a *shift* along an $s \rightsquigarrow o = o_{i'}$ sub-path Z of some path or cycle in $\mathcal{P} \cup \mathcal{C}$. This means that we swap out s and swap in o. We bound the cost of the matching between $\mathcal{F}$ and $S \cup \{o\} \setminus \{s\}$ by moving each initial location $i \in Z,\ i \neq i'$ to $\sigma(o_i) \in Z$ and moving i' to $o_{i'}$. Thus, we obtain the following simple bound on the increase in movement cost due to this operation:

$$\begin{aligned}\mathsf{shift}(s,o) &= \sum_{i\in Z}(f_i^* - f_i) + \sum_{i\in Z: o_i \neq o} c\big(o_i, \sigma(o_i)\big) \\ &\leq 2\sum_{i\in Z} f_i^* - c(o,\sigma(o)).\end{aligned} \tag{3.1}$$

The last inequality uses the fact that $c\big(o_i,\sigma(o_i)\big) \leq c(o_i,s_i) \leq f_i^* + f_i$ for all i. For a path $P \in \mathcal{P}$, We use $\mathsf{shift}(P)$ as a shorthand for $\mathsf{shift}\big(\mathsf{start}(P), \mathsf{end}(P)\big)$.

3.1 The swaps used, and their analysis We now describe the local moves used in the analysis. We define a set of swaps such that each $o \in O$ is swapped in to an extent of at least one, and at most two. We classify each location in S as one of three types. Define $t = \lfloor p^{1/3} \rfloor$. We assume that $t \geq 3$.

1. S_0: locations $s \in S$ with $|\mathsf{cap}(s)| = 0$.
2. S_1: locations $s \in S \setminus S_0$ with $|D^*(\mathsf{center}(s))| \leq t$ or $|\mathsf{cap}(s)| > t$.
3. S_2: locations $s \in S$ with $|D^*(\mathsf{center}(s))| > t$ and $0 < |\mathsf{cap}(s)| \leq t$.

Also define $S_3 := S_0 \cup \{s \in S_1 : |\mathsf{cap}(s)| \leq t\}$ (so $s \in S_3$ iff $|\mathsf{cap}(s)| \leq t,\ |D^*(\mathsf{center}(s))| \leq t\}$).

To gain some intuition, notice that it is easy to generate a suitable inequality for a location $s \in S_0$: we can "delete" s (i.e., if $s = s_i$, then do swap(s,i)) and reassign each $j \in D(s)$ to $\sigma(\sigma^*(j))$ (i.e., the location in S closest to the location serving j in O). The cost increase due to this reassignment is at most $\sum_{j\in D(s)} 2c_j^*$, and so this yields the inequality $f_i \leq \sum_{j\in D(s)} 2c_j^*$. (We do not actually do this since we take care of the S_0-locations along with the S_1-locations.) We can also generate a suitable inequality for a location $s \in S_2$ (see Lemma 3.3) since we can swap in $\mathsf{cap}(s)$ and swap out $\{s\} \cup T(s)$. The cost increase by this move can be bounded by $\sum_{P\in\mathcal{P}_c(s)} \mathsf{shift}(P)$ and $c\big(s, \mathsf{center}(s)\big)$, and the latter quantity can be charged to $\frac{1}{t}\sum_{j\in D^*(\mathsf{center}(s))}(c_j + c_j^*)$; our definition of S_2 is tailored precisely so as to enable this latter charging argument. Generating inequalities for the S_1-locations is more involved, and requires another building block that we call an interval swap (this will also take care of the S_0-locations), which we define after proving Lemma 3.3. We start out by proving a simple bound that one can obtain using a cycle in $\mathcal{C}$.

LEMMA 3.2. *For any cycle $Z \in \mathcal{C}$, we have $0 \leq \sum_{i\in Z}(-f_i + f_i^* + c\big(o_i,\sigma(o_i)\big)$.*

Proof. Consider the following matching of $\mathcal{F} \cap Z$ to $S \cap Z$: we match i to $\sigma(o_i)$. The cost of the resulting new matching is $\sum_{i\notin Z} f_i + \sum_{i\in Z} c(i,\sigma(o_i))$ which should at least $\sum_i f_i$ since the latter is the min-cost way of matching $\mathcal{F}$ to S. So we get that $0 \leq \sum_{i\in Z}(-f_i + c(i,\sigma(o_i)) \leq \sum_{i\in Z}(-f_i + f_i^* + c(o_i,\sigma(o_i))$. ∎

LEMMA 3.3. *Let $s \in S_2$ and $o = \mathsf{center}(s)$, and consider* swap$(X := \{s\} \cup T(s), Y := \mathsf{cap}(s))$. *We have*

$$\begin{aligned}0 \leq \mathsf{MFL}\big((S\setminus X)\cup Y\big) - \mathsf{MFL}(S) &\leq \sum_{\substack{P\in\mathcal{P}_c(s)\\ i\in P}} 2f_i^* \\ + \sum_{j\in D^*(o)}\Big(\tfrac{t+1}{t}\cdot c_j^* - \tfrac{t-1}{t}\cdot c_j\Big) &+ \sum_{\substack{j\in D(\{s\}\cup T(s))\\ j\notin D^*(o)}} 2c_j^*.\end{aligned} \tag{3.2}$$

Proof. We can view this multi-location swap as doing swap$(\mathsf{start}(P), \mathsf{end}(P))$ for each $P \in \mathcal{P}_c(s)$ and swap(s,o) simultaneously. (Notice that no path $P \in \mathcal{P}_c(s)$ contains s, since $s = \sigma\big(\mathsf{end}(P)\big) \notin P$.) For each swap$(\mathsf{start}(P), \mathsf{end}(P))$ the movement-cost increase is bounded by $\mathsf{shift}(P) \leq \sum_{i\in P} 2f_i^*$. For swap$(s,o)$ we move the initial location of facility s to o, so the increase in movement cost is at most $c(s,o) = c(\sigma(o),o) \leq c(\sigma(j),o) \leq c_j + c_j^*$ for every $j \in D^*(o)$. So since $|D^*(o)| > t$, we have $c(s,o) \leq \sum_{j\in D^*(o)} \frac{c_j + c_j^*}{t}$. Thus, the increase in total movement cost is at most $\sum_{j\in D^*(o)} \frac{c_j+c_j^*}{t} + \sum_{P\in\mathcal{P}_c(s), i\in P} 2\cdot f_i^*$.

We upper bound the change in assignment cost by reassigning the clients in $D^*(o) \cup D(X)$ as follows. We reassign each $j \in D^*(o)$ to o. Each $j \in D(X) \setminus D^*(o)$ is assigned to $\sigma^*(j)$, if $\sigma^*(j) \in Y$, and otherwise to $s' = \sigma(\sigma^*(j))$. Note that $s' \notin X$: $s' \neq s$ since $\sigma^*(j) \notin \mathsf{cap}(s)$, and $s' \notin T(s)$ since $\bigcup_{s''\in T(s)} \mathsf{cap}(s'') = \emptyset$. The change in assignment cost for each such client j is at most $2c_j^*$ by Lemma 3.1. Thus the change in total assignment cost is at most $\sum_{j\in D^*(o)}(c_j^* - c_j) + \sum_{j\in D(X)\setminus D^*(o)} 2c_j^*$. Combining this with the bound on the movement-cost change proves the lemma. ∎

We now define a key ingredient of our analysis, called an *interval-swap* operation, that is used to bound the movement cost of the S_1- and S_0-locations and the assignment cost of the clients they serve. (We build upon this in Section 4 to give a tighter analysis proving a

3-approximation.) Let $S' = \{s'_1, \ldots, s'_r\} \subseteq S_0 \cup S_1$, $r \leq t^2$ be a subset of at most t^2 locations on a path in $\mathcal{P}$ or a cycle in $\mathcal{C}$, where s'_{q+1} is the next location in $(S_0 \cup S_1) \cap Z$ after s'_q. Let $O' = \{o'_1, \ldots, o'_r\} \subseteq O$ where $o'_{q-1} = \mathsf{center}(s'_q)$ for $q = 2, \ldots, r$ and o'_r is an arbitrary location that appears after s'_r (and before s'_1) on the corresponding path or cycle. Consider each s'_q. If $|\mathsf{cap}(s'_q)| > t$, choose a *random* path $P \in \mathcal{P}_c(s'_q)$ with probability $\frac{1}{|\mathcal{P}_c(s'_q)|}$, and set $X_q = \{\mathsf{start}(P)\}$ and $Y_q = \{o'_q\}$. If $|\mathsf{cap}(s'_q)| \leq t$, set $X_q = \{s'_q\} \cup T(s'_q)$, and $Y_q = \{o'_q\} \cup H(s'_q)$. Set $X = \bigcup_{q=1}^r X_q$ and $Y = \bigcup_{q=1}^r Y_q$. Note that $|X| = |Y| \leq t^3$ since for each $|X_q| = |Y_q| \leq t$ for every $q = 1, \ldots, r$. Notice that X is a random set, but $Y = O' \cup H(S' \cap S_3)$ is deterministic. To avoid cumbersome notation, we use $\mathrm{swap}(X, Y)$ to refer to the distribution of swap-moves that results by the random choices above, and call this the *interval swap corresponding to S' and O'*. We bound the expected change in cost due to this move below. Let $\mathbf{1}(s)$ be the indicator function that is 1 if $s \in S_3$ and 0 otherwise.

LEMMA 3.4. *Let $S' = \{s'_1, \cdots, s'_r\} \subseteq S_0 \cup S_1$, $r \leq t^2$ and O' be as given above. Let $o'_0 := \mathsf{center}(s'_1) = o_{\hat{\imath}}$, where $o'_0 = \mathsf{nil}$ and $D^*(o'_0) = \emptyset$ if $s'_1 \in S_0$. Consider the interval swap $\mathrm{swap}\big(X = \bigcup_{q=1}^r X_q, Y = \bigcup_{q=1}^r Y_q\big)$ corresponding to S' and O', as defined above. We have*

$$0 \leq \mathrm{E}\left[\mathsf{MFL}\big((S \setminus X) \cup Y\big) - \mathsf{MFL}(S)\right] \leq \sum_{q=1}^{r} \mathsf{shift}(s'_q, o'_q) + \sum_{P \in \mathcal{P}_c(S'), i \in P} 2f_i^* + \sum_{j \in D^*(O')} (c_j^* - c_j) + \sum_{j \in D(T(S' \setminus S_3))} \tfrac{2c_j^*}{t} + \sum_{j \in D(T(S' \cap S_3) \cup (S' \cap S_3))} 2c_j^* + \mathbf{1}(s'_1) \sum_{j \in D^*(o'_0)} (f_{\hat{\imath}}^* + f_{\hat{\imath}} + c_j^*). \tag{3.3}$$

Proof. Let Z be the path in $\mathcal{P}$ or cycle in $\mathcal{C}$ such that $S' \cup O' \subseteq Z$.

We first bound the increase in movement cost. The interval swap can be viewed as a collection of simultaneous $\mathrm{swap}(X_q, Y_q)$, $q = 1, \ldots, r$ moves. If $X_q = \{\mathsf{start}(P)\}$ for a random path $P \in \mathcal{P}_c(s'_q)$, the movement-cost increase can be broken into two parts. We do a shift along P, but move the last initial location on P to s'_q, and then do shift on Z from s'_q to o'_q. So the expected movement-cost change is at most

$$\frac{1}{|\mathcal{P}_c(s'_q)|} \sum_{P \in \mathcal{P}_c(s'_q)} \big(\mathsf{shift}(P) + c(\mathsf{end}(P), s'_q)\big) + \mathsf{shift}(s'_q, o'_q) \leq \frac{1}{|\mathcal{P}_c(s'_q)|} \sum_{P \in \mathcal{P}_c(s'_q), i \in P} 2f_i^* + \mathsf{shift}(s'_q, o'_q)$$

which is at most $\sum_{P \in \mathcal{P}_c(s'_q), i \in P} 2f_i^* + \mathsf{shift}(s'_q, o'_q)$. Similarly, if $|\mathsf{cap}(s'_q)| \leq t$, we can break the movement-cost increase into $\mathsf{shift}(P) \leq \sum_{i \in P} 2f_i^*$ for all $P \in \mathcal{P}_c(s'_q)$ and $\mathsf{shift}(s'_q, o'_q)$. Thus, the total increase in movement cost is at most

$$\sum_{q=1}^{r} \mathsf{shift}(s'_q, o'_q) + \sum_{P \in \mathcal{P}_c(S'), i \in P} 2f_i^*. \tag{3.4}$$

Next, we bound the change in assignment cost by reassigning clients in $\widehat{D} = D^*(O') \cup D(X)$ as follows. We assign each client $j \in D^*(O')$ to $\sigma^*(j)$. If $|\mathsf{cap}(s'_1)| > t$, then $s'_1 \notin X$. For every client $j \in \widehat{D} \setminus (D^*(O'))$, observe that either $\sigma^*(j) \in Y$ or $\sigma(\sigma^*(j)) \notin X$. To see this, let $o = \sigma^*(j)$ and $s = \sigma(o)$. If $o \notin Y$ then $s \notin S' \cap S_3$; also $s \notin T(S')$, and so $s \notin X$. So we assign j to $\sigma^*(j)$ if $\sigma^*(j) \in Y$ and to $\sigma(\sigma^*(j))$ otherwise; the change in assignment cost of j is at most $2c_j^*$ (Lemma 3.1).

Now suppose $|\mathsf{cap}(s'_1)| \leq t$, so $s'_1 \in X$. For each $j \in \widehat{D} \setminus (D^*(O') \cup D^*(o'_o))$, we again have $\sigma^*(j) \in Y$ or $\sigma(\sigma^*(j)) \notin X$, and we assign j to $\sigma^*(j)$ if $\sigma^*(j) \in Y$ and to $\sigma(\sigma^*(j))$ otherwise. We assign every $j \in \widehat{D} \cap D^*(o'_0)$ to $s_{\hat{\imath}}$ (recall that $o'_0 = o_{\hat{\imath}}$), and overestimate the resulting change in assignment cost by $\sum_{j \in D^*(o'_o)} (c_j^* + f_{\hat{\imath}}^* + f_{\hat{\imath}})$. Finally, note that we reassign a client $j \in D(T(S' \setminus S_3)) \setminus D^*(O')$ with probability at most $\frac{1}{t}$ (since $\sigma(j) \in X$ with probability at most $\frac{1}{t}$). So taking into account all cases, we can bound the change in total assignment cost by

$$\sum_{j \in D^*(O')} (c_j^* - c_j) + \sum_{j \in D(T(S' \cap S_3) \cup (S' \cap S_3))} 2c_j^* + \sum_{j \in D(T(S' \setminus S_3))} \tfrac{2c_j^*}{t} + \mathbf{1}(s'_1) \sum_{j \in D^*(o'_0)} (f_{\hat{\imath}}^* + f_{\hat{\imath}} + c_j^*). \tag{3.5}$$

In (3.5), we are double-counting clients in $D\big(T(S') \cup (S' \cup S_3)\big) \cap D^*(O')$. We are also overestimating the change in assignment cost of a client $j \in D(X) \cap D^*(o'_0)$ since we include both the $\mathbf{1}(s'_1)(c_j^* + f_{\hat{\imath}}^* + f_{\hat{\imath}})$ term, and the $2c_j^*$ or $\frac{2c_j^*}{t}$ terms. Adding (3.4) and (3.5) yields the lemma. ∎

Notice that Lemma 3.3 immediately translates to a bound on the assignment cost of the clients in $D^*(\mathsf{center}(s))$ for $s \in S_2$. In contrast, it is quite unclear how Lemma 3.4 may be useful, since the expression $\sum_{j \in D^*(o'_0)} (f_{\hat{\imath}}^* + f_{\hat{\imath}})$ in the RHS of (3.3) may be as large as $t(f_{\hat{\imath}}^* + f_{\hat{\imath}})$ (but no more since $|D^*(o'_0)| \leq t$ if $\mathbf{1}(s'_1) = 1$) and it is unclear how to cancel the contribution of $f_{\hat{\imath}}$ on the RHS. One of the novelties of our analysis is that we show how to *amortize* such expensive terms and make their contribution negligible by considering multiple interval swaps. We cover each path or cycle Z in t^2 different ways using intervals comprising

consecutive locations from $S_0 \cup S_1$. We then argue that averaging, over these t^2 covering ways, the inequalities obtained from the corresponding interval swaps yields (among other things) a good bound on the movement-cost of the $(S_0 \cup S_1)$-locations on Z and the assignment cost of the clients they serve.

LEMMA 3.5. *Let $Z \in \mathcal{P} \cup \mathcal{C}$, $S' = \{s'_1, \ldots, s'_r\} = S_1 \cap Z$, where s'_{q+1} is the next S_1-location on Z after s'_q, and $O' = \{\mathsf{center}(s'_1), \ldots, \mathsf{center}(s'_r)\}$. Let $o'_r = \mathsf{end}(Z)$ if $Z \in \mathcal{P}$ and $\mathsf{center}(s'_1)$ otherwise. For $r \geq t^2$,*

$$\begin{aligned} 0 \;\leq\; & \sum_{i \in Z} \Big(\tfrac{t+1}{t} \cdot f^*_i - \tfrac{t-1}{t} f_i \Big) + \sum_{P \in \mathcal{P}_c(S'), i \in P} 2 f^*_i \\ & + \sum_{j \in D^*(Z \cap O)} \Big(\tfrac{1}{t} \cdot c_j + \tfrac{t+1}{t^2} \cdot c^*_j \Big) \\ & + \sum_{j \in D^*(O' \cup \{o'_r\})} (c^*_j - c_j) + \sum_{j \in D(T(Z \cap S_3) \cup (Z \cap S_3))} 2 c^*_j \\ & + \sum_{j \in D(T(S' \setminus S_3))} \tfrac{2 c^*_j}{t}. \end{aligned} \tag{3.6}$$

Proof. We first define formally an interval of (at most) t^2 consecutive $(S_0 \cup S_1)$ locations along Z. As before, let $o'_{q-1} = \mathsf{center}(s'_q)$ for $q = 1, \ldots, r$. For a path Z, define $s'_q = \mathsf{start}(Z)$ for $q \leq 0$ and $s'_q = \mathsf{nil}$ for $q > r$. Also define $o'_q = o'_0$ for $q \leq 0$ and $o'_q = \mathsf{end}(Z)$ for $q \geq r$. If Z is a cycle, we let our indices wrap around and be mod r, i.e., $s'_q = s'_{q \bmod r}$, $o'_q = o'_{q \bmod r}$ for all q (so $o'_r = o'_0 = \mathsf{center}(s'_1)$).

For $1 - t^2 \leq h \leq r$, define $S'_h = \{s'_h, s'_{h+1}, \ldots, s'_{h+t^2-1}\}$ to be an interval of length at most t^2 on Z. Define $O'_h = \{o'_h, o'_{h+1}, \ldots, o'_{h+t^2-1}\}$. Note that we have $1 \leq |S'_h| = |O'_h| \leq t^2$ if Z is a path, and $|S'_h| = |O'_h| = t^2$ if Z is a cycle. Consider the collection of intervals, $\{S'_{-t^2+1}, S'_{-t^2+2}, \cdots, S'_r\}$. For each S'_h, O'_h, where $-t^2 + 1 \leq h \leq r$, we consider the interval swap (X_h, Y_h) corresponding to S'_h, O'_h. We add the inequalities $\frac{1}{t^2} \times$(3.3) for all such h. Since each $s' \in S' \cup \{s'_0\}$ participates in exactly t^2 such inequalities, and each $s'_h \in S'$ is the start of only the interval S'_h, we obtain the following.

$$\begin{aligned} 0 \;\leq\; & \sum_{q=0}^{r} \frac{1}{t^2} \cdot t^2 \cdot \mathsf{shift}(s'_q, o'_q) + \sum_{P \in \mathcal{P}_c(S'), i \in P} \frac{1}{t^2} \cdot t^2 \cdot 2 f^*_i \\ & + \sum_{j \in D^*(O' \cup \{o'_r\}))} \frac{1}{t^2} \cdot t^2 \cdot (c^*_j - c_j) \\ & + \sum_{j \in D(T(S' \setminus S_3))} \frac{1}{t^2} \cdot t^2 \cdot \frac{2 c^*_j}{t} \\ & + \sum_{j \in D(T(Z \cap S_3) \cup (Z \cap S_3))} \frac{1}{t^2} \cdot t^2 \cdot 2 c^*_j \\ & + \sum_{i : \sigma(o_i) \in Z} \mathbf{1}(\sigma(o_i)) \cdot \frac{1}{t^2} \cdot \sum_{j \in D^*(o_i)} (f^*_i + f_i + c^*_j). \end{aligned} \tag{3.7}$$

Notice that the S-locations other than s'_q on the $s'_q \rightsquigarrow o'_q$ sub-paths of Z lie in S_2, and for each i such that $\sigma(o_i) \in Z \cap S_2$, we have $c(o_i, \sigma(o_i)) \leq \frac{c_j + c^*_j}{t}$. Thus, using (3.1), we have

$$\begin{aligned} \sum_{q=0}^{r} \mathsf{shift}(s'_q, o'_q) &= \sum_{i \in Z} (f^*_i - f_i) + \sum_{i : \sigma(o_i) \in Z \cap S_2} c(o_i, \sigma(o_i)) \\ &\leq \sum_{i \in Z} (f^*_i - f_i) + \sum_{j \in D^*(Z \cap O)} \frac{c_j + c^*_j}{t}. \end{aligned} \tag{3.8}$$

Since $\mathbf{1}(\sigma(o)) = 1$ means that $\sigma(o) \in S_3$, and so $|D^*(o)| \leq t$, we have $\sum_{i : \sigma(o_i) \in Z \cap S_3} \sum_{j \in D^*(o_i)} \frac{f^*_i + f_i + c^*_j}{t^2}$ is at most

$$\sum_{i \in Z} \Big(\tfrac{f^*_i + f_i}{t} + \tfrac{\sum_{j \in D^*(o_i)} c^*_j}{t^2} \Big) \leq \sum_{i \in Z} \tfrac{f^*_i + f_i}{t} + \sum_{j \in D^*(Z \cap O)} \tfrac{c^*_j}{t^2}. \tag{3.9}$$

Incorporating (3.8) and (3.9) in (3.7), and simplifying yields the desired inequality. ∎

For a path or cycle Z where $|S_1 \cap Z| < t^2$, we obtain an inequality similar to (3.6). Since we can now cover Z with a single interval, we never have a client j such that none of $\sigma(j)$, $\sigma^*(j)$, $\sigma(\sigma^*(j))$ are in our new set of final locations. So the resulting inequality does not have any $\frac{f^*_i + f_i}{t} + \frac{c^*_j}{t^2}$ terms.

LEMMA 3.6. *Let $Z \in \mathcal{P} \cup \mathcal{C}$, $S' = \{s'_1, \ldots, s'_r\} = S_1 \cap Z$, where s'_{q+1} is the next S_1-location on Z after s'_q, and $O' = \{\mathsf{center}(s'_1), \ldots, \mathsf{center}(s'_r)\}$. Let $o'_r = \mathsf{end}(Z)$ if*

$Z \in \mathcal{P}$ *and* $\mathsf{center}(s'_1)$ *otherwise. For* $r' < t^2$,

$$0 \;\le\; \sum_{i\in Z}(f^*_i - f_i) + \sum_{P\in\mathcal{P}_c(S'),i\in P} 2f^*_i + \sum_{j\in D^*(Z\cap O)} \frac{c_j + c^*_j}{t} + \sum_{j\in D^*(O'\cup\{o'_r\})} (c^*_j - c_j) + \sum_{j\in D(T(Z\cap S_3)\cup(Z\cap S_3))} 2c^*_j + \sum_{j\in D(T(S'\setminus S_3))} \tfrac{2c^*_j}{t}. \tag{3.10}$$

Proof. The proof is similar to that of Lemma 3.5, except that since we can cover Z with a single interval, we only need to consider a single (multi-location) swap. We consider two cases for clarity.

1. ***Z* is a path, or $r > 0$.** As before, let $o'_{q-1} = \mathsf{center}(s'_q)$ for $q = 1,\ldots,r$. If Z is a path, define $s'_0 = \mathsf{start}(Z)$. If Z is a cycle, we again set $s'_q = s'_{q \bmod r}$, $o'_q = o'_{q \bmod r}$ for all q. Consider the interval swap (X,Y) corresponding to $S'\cup\{s'_0\}, O'\cup\{o'_r\}$. The inequality generated by this is similar to (3.3) except that we do not have any $(f^*_i + f_i + c^*_j)$ terms since for client $j \in D(X)\cup D^*(Y)$, we always have that either $\sigma^*(j) \in Y$ or $\sigma(\sigma^*(j)) \notin X$. Thus, (3.3) translates to the following.

$$0 \le \sum_{q=0}^{r} \mathsf{shift}(s'_q, o'_q) + \sum_{P\in\mathcal{P}_c(S'),i\in P} 2f^*_i + \sum_{j\in D^*(O'\cup\{o'_r\})} (c^*_j - c_j) + \sum_{j\in D(T(Z\cap S_3)\cup(Z\cap S_3))} 2c^*_j + \sum_{j\in D(T(S'\setminus S_3))} \tfrac{2c^*_j}{t}.$$

Substituting $\sum_{q=0}^{r} \mathsf{shift}(s'_q, o'_q) \le \sum_{i\in Z}(f^*_i - f_i) + \sum_{j\in D^*(Z\cap O)} \frac{c_j + c^*_j}{t}$ as in (3.8) yields the stated inequality.

2. ***Z* is a cycle with $r = 0$.** Here, Lemma 3.2 yields $0 \le \sum_{i\in Z}(f^*_i - f_i) + \sum_{j\in D^*(Z\cap O)} \frac{c_j + c^*_j}{t}$ (which is the special case of the earlier inequality with $s'_0 = \mathsf{nil} = o'_r$, $S' = O' = Z\cap S_3 = \emptyset$). ■

Proof of Theorem 3.1. We consider the following set of swaps.

- A1 For every $s \in S_2$, the move $\mathrm{swap}\big(\{s\}\cup T(s), \mathsf{cap}(s)\big)$.
- A2 For every path or cycle Z with $|Z\cap S_1| \ge t^2$, the $\frac{1}{t^2}$-weighted interval swaps as defined in Lemma 3.5.
- A3 For every path or cycle Z with $|Z\cap S_1| < t^2$, the interval swap defined in Lemma 3.6.

Notice that every location $o \in O$ is swapped in to an extent of at least 1 and at most 2. (By "extent" we mean that the total weight of the inequalities involving o.) To see this, suppose first $o = \mathsf{end}(Z)$ for some path Z, then o is involved to an extent of 1 in the interval swaps for Z in A2 or A3. In this case, we say that the interval swap for Z is *responsible* for o. Additionally, if $s = \sigma(o) \in S_2$, then o belongs to the multi-swap for s in A1, else if $s \in S_3$ then o is part of the interval swap for the path/cycle containing s in A2 or A3. Now suppose $o = \mathsf{center}(s)$. If $s \in S_2$, then o is included in the multi-swap for s in A1. We say that this multi-swap is responsible for o. If $s \in S_1$, then o is included in the interval swap for the path/cycle containing s in A2 or A3; we say that this interval swap is responsible for o.

Consider the compound inequality obtained by summing (3.2), (3.6), and (3.10) corresponding to the moves considered in A1, A2, and A3 respectively. The LHS of this inequality is 0. We now need to do some bookkeeping to bound the coefficients of the f^*_i, f_i, c^*_j, c_j terms on the RHS. We ignore $o(1)$ coefficients like $\frac{1}{t}$, $\frac{1}{t^2}$ in this bookkeeping, since for a given $\{f^*_i, f_i, c^*_j, c_j\}$ term, such coefficients appear in only a constant number of inequalities, so they have $o(1)$ effect overall. Let F and C denote respectively the movement- and assignment- cost of the local optimum.

Contribution from the c^*_j and c_j terms. First, observe that for each $o \in O$, we have labeled exactly one move involving o as being responsible for it. Consider a client $j \in D(s)\cap D^*(o)$. Observe that c^*_j or c_j terms appear (with a $\Theta(1)$-coefficient) in an inequality generated by a move if (i) j is reassigned because the move is responsible for o; or (ii) s is swapped out (to an extent of 1) by the move (so this excludes the case where $s \in T(s')$, $s' \in S_1\setminus S_3$ and the move is the interval swap for the path containing s'). If (i) applies, then the inequality generates the term $(c^*_j - c_j)$. If (ii) applies then the term $2c^*_j$ appears in the inequality. Finally, note that there are at most two inequalities for which (ii) applies:

- If $s = \mathsf{start}(Z) \in S_0$, then (ii) applies for the interval-swap move for Z. If $s' = \sigma(\mathsf{end}(Z)) \in S_2\cup S_3$, then (ii) again applies, for the multi-swap move for s' if $s' \in S_2$, or for the interval swap for the path containing s' if $s' \in S_3$.
- If $s \in S_1\cap S_3$, then (ii) applies for the interval swap for the path containing s.
- If $s \in S_2$, then (ii) applies for the multi-swap move for s.

So overall, we get a $5c^*_j - c_j$ contribution to the RHS, the bottleneck being the two inequalities for which (ii) applies when $s \in \mathsf{start}(Z)$, $\sigma(\mathsf{end}(Z)) \in S_2\cup S_3$.

Contribution from the f_i^* and f_i terms. For every $i \in \mathcal{F}$, the expression $(f_i^* - f_i)$ is counted once in the RHS of the inequality (3.6) or (3.10) for the unique path or cycle Z containing i. The total contribution of all these terms is therefore, $F^* - F$. The remaining contribution comes from expressions of the form $\sum_{P \in \mathcal{P}_c(s), i \in P} 2f_i^*$ on the RHS of (3.2), (3.6), and (3.10). The paths P involved in these expressions come from $\mathcal{P}_c(S_2) \cup \big(\bigcup_{Z \in \mathcal{P} \cup \mathcal{C}} \mathcal{P}_c(Z \cap S_3)\big) \subseteq \mathcal{P}$. Therefore, the total contribution of these terms is at most $2F^*$.

Thus, we obtain the compound inequality

$$0 \leq \big(5 + o(1)\big)C^* + \big(3 + o(1)\big)F^* - \big(1 - o(1)\big)(F + C)$$

where the $o(1)$ terms are $O\big(\frac{1}{t}\big) = O\big(p^{-1/3}\big)$. This shows that $F + C \leq \big(3 + o(1)\big)F^* + \big(5 + o(1)\big)C^*$. ■

4 Improved analysis leading to a 3-approximation

In this section, we improve the analysis from Section 3. Specifically, we prove the following theorem.

THEOREM 4.1. *The cost of a local optimum solution using p swaps is at most $3 + O\left(\sqrt{\frac{\log\log p}{\log p}}\right)$ times the optimum solution cost.*

To gain some intuition behind this tighter analysis, note that the *only* reason we lost a factor of 5 in the previous analysis was because there could be locations $s = \mathsf{start}(Z) \in S_0$ that are swapped out to an extent of 2; hence, there could be clients $j \in D(s)$ for which we "pay" $2c_j^*$ each time s is swapped out, and also pay an additional $c_j^* - c_j$ term when $\sigma^*(j)$ is swapped in. To improve the analysis, we will consider a set of test swaps that swap out each location in S to an extent of $1 + o(1)$.

The aforementioned bad case happens only when $s' = \sigma(\mathsf{end}(Z)) \in S_2 \cup S_3$, because when we close (i.e., swap out) s' as part of an interval swap or a multi-swap, we open (i.e., swap in) all the locations in $H(s')$, and we achieve this via path swaps (i.e., shift moves) along paths in $\mathcal{P}_c(s')$ that swap out locations in $T(s')$ (for a second time). The main idea behind our refined analysis is to not perform such path swaps, but instead to "recursively" start an interval swap on each path in $\mathcal{P}_c(s')$. Of course, we cannot carry out this recursion to arbitrary depth so we terminate the recursion at a depth of t^2. So, whereas an interval swap included at most t^2 S_1-locations on the main path or cycle Z, we now consider a "subtree" swap obtained by aggregating interval swaps on the paths in $\bigcup \mathcal{P}_c(Z \cap S_3)$. A subtree swap can be viewed as a bounded-depth recursion tree where each leaf to root path encounters at most t^2 locations in S_1. Because we no longer initiate path swaps for S_3-locations, a leaf location $s'' \in S_3$ in this recursion tree will not have any locations in $\mathsf{cap}(s'')$ opened. But we will slightly redefine the S_1, S_2, S_3 sets to ensure that $|D^*(\mathsf{cap}(s''))| \leq t$, and *average* over different sets of subtree swaps (like we did with interval swaps in Section 3) to ensure that s'' is a leaf location with probability at most $\frac{1}{t^2}$; this ensures that we incur, to an extent of at most $\frac{1}{t}$, the cost $f_i^* + f_i + c(o_i, s'')$, where $o_i = \mathsf{center}(s'')$, for moving j with $\sigma(\sigma^*(j)) = s''$ from s'' to s_i.

Notation. Let t be an integer such that $p \geq t^2 t^{t^2} + 1$. We prove that the local-search algorithm has approximation ratio $3 + O(t^{-1})$. We redefine S_0, S_1, S_2, S_3 as: $S_0 = \{s \in S : |\mathsf{cap}(s)| = 0\}$, $S_1 = \{s \in S \setminus S_0 : |D^*(\mathsf{cap}(s))| \leq t \text{ or } |\mathsf{cap}(s)| > t\}$, $S_2 = \{s \in S : |D^*(\mathsf{cap}(s))| > t,\ |\mathsf{cap}(s)| \leq t\}$, and $S_3 = S_0 \cup \{s \in S_1 : |\mathsf{cap}(s)| \leq t\}$. We redefine $\mathsf{center}(s)$ to be the location in $\mathsf{cap}(s)$ closest to s.

CLAIM 4.1. *Let $s \in S_2$ and $o = \mathsf{center}(s)$. Then $c(s, o) \leq \frac{1}{t} \sum_{j \in D^*(\mathsf{cap}(s))} (c_j + c_j^*)$.*

Proof. We have $c(s, o) \leq c(s, o')$ for any $o' \in \mathsf{cap}(s)$, and $c(s, o') \leq c_j + c_j^*$ for any $j \in D^*(o')$. Therefore, $c(s, o) \leq c_j + c_j^*$ for any $j \in D^*(\mathsf{cap}(s))$, and the claim follows since $|D^*(\mathsf{cap}(s))| > t$ as $s \in S_2$.

It will be more convenient to work with the digraph $H = (\mathcal{F}, E)$ obtained from $\widehat{G}$ by contracting each triple $\{s_i, i, o_i\}$ of nodes associated with a facility i into a single node that we also denote by i. Thus, (i, i') is an arc in E if $\sigma(o_i) = s_{i'}$ (it may be that $i = i'$). Note that H may have self loops, and each node in H has outdegree exactly 1 so each component of H looks like a tree with all edges oriented toward the root, except the root is in fact a directed cycle (possibly a self-loop). Figure 1 illustrates this graph and some of the subgraphs and structures discussed below. For brevity, we say that an edge (i, i') in H is a *center* edge if $o_i = \mathsf{center}(s_{i'})$. In the arc set $E' = \{(i, i') \in E : o_i = \mathsf{center}(s_{i'})\}$, each node has indegree and outdegree at most 1, so E' consists of a collection node-disjoint paths $\mathcal{P}$ and cycles $\mathcal{C}$. For a facility $i \in \mathcal{P}$, let $\mathcal{P}(i)$ denote the unique path in $\mathcal{P}$ containing i. Let $\mathsf{start}(i)$ and $\mathsf{end}(i)$ denote the start and end facilities of $\mathcal{P}(i)$ respectively. Define $H^* = \big(\mathcal{F}, E' \cup \{(i, i') : s_{i'} \in S_3, \sigma(o_i) = s_{i'}\}\big)$. Call a node i of H^* a *root* if i has no outgoing arc or i lies on a directed cycle in H^*.

We consider an integer $1 \leq l \leq t^2$ and describe a set of swaps for l. The inequalities for the swaps for different l will be averaged in the final analysis. We obtain H_l^* by deleting the following (i, i') edges from H^*: i is not on a cycle, $s_{i'} \in S_1$, and the number

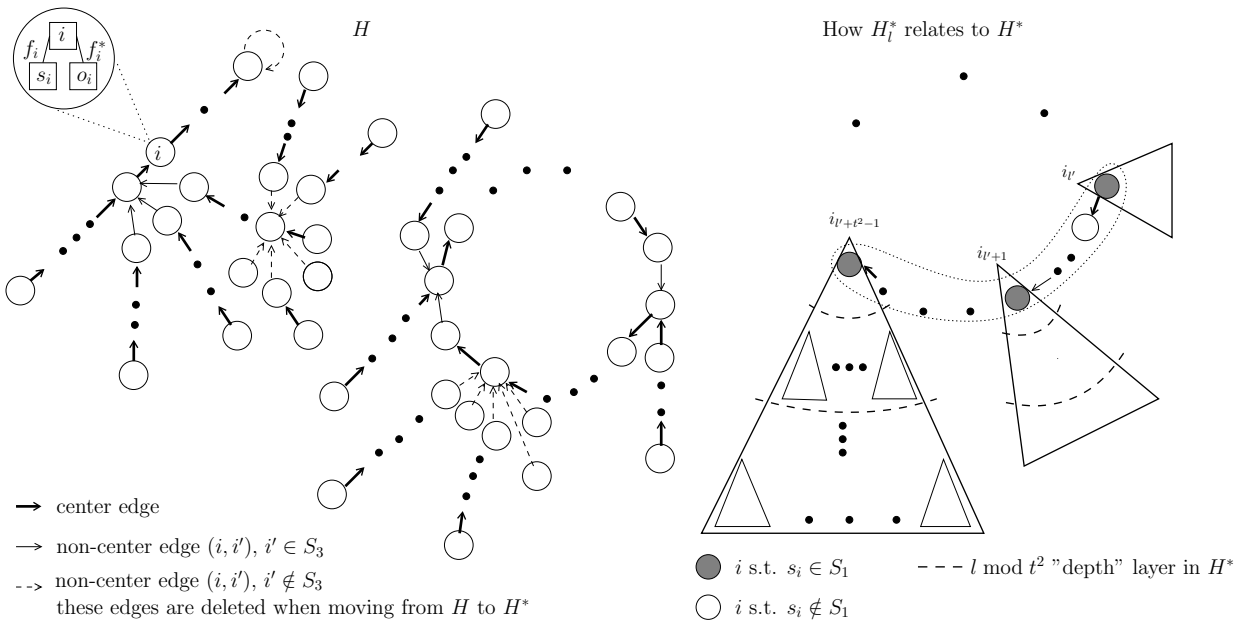


Figure 1: The digraph H and related structures

of facilities i'' with $s_{i''} \in S_1$ on the path between i' and the root of its component in H^* (both included) is $l \bmod t^2$. A *subtree* of H_l^* is an acyclic component of H_l^*, or a component that results by deleting the edges of the cycle contained in a component of H_l^*

For a facility i, define $\mathsf{cand}(i) = \{i' : o_{i'} \in \mathsf{cap}(s_i) \setminus \{\mathsf{center}(s_i)\},\ \nexists i \rightsquigarrow i' \text{ path in } H^*\}$. Note that $|\mathsf{cand}(i)| \geq |\mathsf{cap}(s_i)| - 2$. The reason we define $\mathsf{cand}(i)$ is that we will sometimes perform a shift along some path $Z \in \mathcal{P}_c(s_i)$ to reassign the facilities on Z but we will not want this to interfere with the operations in the subtree of H_l^* containing i. For a facility i with $s_i \notin S_2$, let $\mathsf{next}(i)$ be the facility obtained as follows. Follow the unique walk from i in H using only center edges until the walk reaches a node i' with either no outgoing center edge, or the unique (i', i'') center edge satisfies $s_{i''} \in S_1$; we set $\mathsf{next}(i) = i'$.

Claim 4.2. *The number of facilities i with $s_i \in S_0 \cup S_1$ in any subtree of H_l^* is at most t^{t^2}.*

Proof. The facilities i in such a subtree that are in S_2 have indegree and outdegree at most 1. Shortcutting past these facilities yields a tree with depth at most t^2 and branching factor at most t. ∎

The test swaps For a subtree T of H_l^*, we describe a set of nodes X_T to be swapped out and a set of nodes Y_T to be swapped in with $|X_T| = |Y_T| \leq t^{t^2}$. We do not actually perform these swaps yet to generate the inequalities since we will have to combine some of these swaps for various components.

For each $i \in T$ with $s_i \in S_0 \cup S_1$, we add the following location in S to X_T: if $s_i \in S_3$ we add s_i to X_T; otherwise (so $s_i \in S_1 \setminus S_3$), we choose any single $i' \in \mathsf{cand}(i)$ uniformly at random and add $s_{\mathsf{start}(i')}$ to X_T. We also add $o_{\mathsf{next}(i)}$ to Y_T.

When we say perform swap(X_T, Y_T), we specifically mean the following reassignment of facilities. For $s_i \in X_T$ with $s_i \in S_0 \cup S_3$, we perform $\mathsf{shift}(s_i, o_{\mathsf{next}(i)})$. For $s_i \in X_T$ with $s_i \in S_1 \setminus S_3$, say i' is the facility in $\mathsf{cand}(i)$ for which $s_{\mathsf{start}(i')}$ was added to X_T. Then we perform $\mathsf{shift}(s_{\mathsf{start}(i')}, o_{i'})$, move facility i' from $o_{i'}$ to s_i, and finally perform $\mathsf{shift}(s_i, o_{\mathsf{next}(i)})$. As always, each client is then assigned to its nearest final location. Lemma 4.1 implies that these shift operations charge different portions of the local and global optimum.

Lemma 4.1. *For a subtree T, all of the shift operations described for* swap(X_T, Y_T) *have their associated paths being vertex disjoint.*

Proof. For any subtree T, the paths between s_i and $o_{\mathsf{next}(i)}$ for the facilities $i \in T$ with $s_i \in S_0 \cup S_1$ are vertex-disjoint by definition of $\mathsf{next}(i)$. Also, for any two distinct $i_1, i_2 \in T$, and any $i \in \mathsf{cand}(i_1)$, $i' \in \mathsf{cand}(i_2)$, we have $\mathsf{start}(i) \neq \mathsf{start}(i')$, and so their associated paths $\mathcal{P}(i)$ and $\mathcal{P}(i')$ are also vertex-disjoint.

Finally, consider any $i \in T$ with $s_i \in S_0 \cup S_1$, and $i'' \in T$ (i'' could be i) with $s_{i''} \in S_1 \setminus S_3$. Consider the paths involved in swap$(s_i, o_{\mathsf{next}(i)})$ and swap$(s_{\mathsf{start}(i')}, o_{i'})$, where $i' \in \mathsf{cand}(i'')$. Note that both of these paths consist of only center edges. Therefore,

since each facility has at most one incoming and one outgoing center edge, and $i' = \mathsf{end}(i')$, if these paths are not vertex-disjoint, then it must be that the path involved in $\text{swap}(s_i, o_{\mathsf{next}(i)})$ is a subpath of the path involved in $\text{swap}(s_{\mathsf{start}(i')}, o_{i'})$. This means that i and i', and hence, i, i', i'', are all in the same component of H^*. Also, the edge (i', i'') is not in H^* so i' is the root of its component in H^*. But then there is a path from i'' to i', which contradicts that $i' \in \mathsf{cand}(i'')$. ∎

We need to coordinate the swaps for various subtrees of H_l^*. Consider a component Z in H^*. Let $C = \emptyset$ if Z is rooted at a node, otherwise let C be its cycle of root nodes. Let $i_1, \ldots, i_k$ be the facilities on C with $s_i \in S_1$, indexed by order of appearance on C starting from an arbitrary facility i_1 on C ($k = 0$ if $C = \emptyset$). We consider four kinds of swaps.

Type 1. If $1 \leq k \leq t^2$, simultaneously do $\text{swap}(X_T, Y_T)$ for all subtrees T rooted at some $i \in C$ with $s_i \in S_1$.

Type 2. Otherwise, if $k > t^2$, define $I_{l'} = \{i_{l'}, i_{l'+1}, \ldots, i_{l'+t^2-1}\}$ for all $l' = 1, \ldots, k$ (where the indices are mod k). Simultaneously perform $\text{swap}(X_T, Y_T)$ for each subtree T rooted at a facility in $I_{l'}$. Reasoning similarly as in Lemma 4.1 and noting that different subtrees involved in a single type-1 or type-2 swap are all disjoint, we can see that all shift paths involved in a single type-1 or type-2 swap are vertex-disjoint.

Type 3. For each i with $s_i \in S_2$, simultaneously perform $\text{swap}(X_T, Y_T)$ for all subtrees T rooted at some i' with $o_{i'} \in \mathsf{cap}(s_i) \setminus \{\mathsf{center}(s_i)\}$. At the same time, we also swap out s_i and swap in $o_{i''} = \mathsf{center}(s_i)$ for a total of at most $t^{t^2+1} + 1 \leq p$ swaps. It may be that some (at most one) shift path in this swap includes s_i, but then we just move i'' to $o_{i''}$ instead of s_i, and then move s_i according to the shift operation.

Type 4. Finally, for every other subtree T of H_l^* that was not swapped in the previous cases, perform $\text{swap}(X_T, Y_T)$ on its own.

Analysis. We first bound the net client-assignment cost increase for any single one of these test swaps. So, fix one such swap, let $\{T_r\}_{r=1}^k, k \leq t^2$ be the set of subtrees involved in the swap, and let B denote the set of facilities i such that $o_i = \mathsf{center}(\sigma(o_i))$ and $\sigma(o_i)$ is closed during the swap while o_i is not opened. So B consists of facilities with a center edge to some leaf of some subtree T_r or, if the swap is of type 2, to the start of an interval $I_{l'}$. For this swap, let $C_1 = \{j \in \mathcal{D} : \sigma^*(j) \text{ is opened}\}$, $C_2 = D^*(\{o_i : i \in B\})$, and $C_3 = \{j : \sigma(j) = s_i \in S_0 \text{ and } \mathsf{end}(i) \in \mathsf{cand}(i') \text{ for some } i' \in T_r\}$.

LEMMA 4.2. *The expected change in client-assignment cost for a test swap is at most* $\sum_{j \in C_1}(c_j^* - c_j) + \sum_{j \in C_2} 2c_j^* + \frac{1}{t-1}\sum_{j \in C_3} 2c_j^* + 2t\sum_{i \in B}(f_i^* + f_i)$

Proof. After the swap, we move every $j \in C_1$ from $\sigma(j)$ to $\sigma^*(j)$ for a cost change of $c_j^* - c_j$. Every client in $j \in C_2 \cup C_3$ for which $\sigma(j)$ is closed is moved initially to $\sigma(\sigma^*(j))$ for a cost increase of at most $2c_j^*$. Suppose $\sigma^*(j) = o_i$.

Suppose i is such that $\sigma(o_i) = \sigma(\sigma^*(j))$ and $o_i = \mathsf{center}(\sigma(o_i))$. It may be that $\sigma(o_i)$ is still not open which means that $i \in B$. Note that either s_i or o_i is opened after the shift and we move every client that was moved to $\sigma(o_i)$ to s_i or o_i (whichever is open). This extra distance moved is at most $f_i^* + f_i + c(o_i, \sigma(o_i)) \leq 2f_i^* + 2f_i$. Note that $i \in B$ implies that $\sigma(o_i) \in S_3$, otherwise $\sigma(o_i)$ would not have been closed down in the swap. So $|D^*(\mathsf{cap}(\sigma(o_i)))| \leq t$, by definition of S_3, and at most t clients will be moved to either s_i or o_i in this manner.

Finally, we note that while $j \in C_3$ may have $\sigma(j)$ being closed, this only happens with probability at most $\frac{1}{t-1}$. ∎

Now, we consider the following weightings of the swaps. First, for a particular $1 \leq l \leq t^2$ we perform all type 1, 3, and 4 swaps. For a component of H_l^* containing a cycle C, we perform all type 2 swaps for the various intervals $I_{l'}$ for C and weight the client and facility cost change by $\frac{1}{t^2}$. Finally, these weighted bounds on the client and facility cost change are averaged over all $1 \leq l \leq t^2$.

LEMMA 4.3. *The expected total client-assignment cost change, weighted in the described manner, is at most* $\sum_j 3c_j^* - c_j + O\left(\frac{1}{t}\right)\left(\sum_i (f_i^* + f_i) + \sum_j c_j^*\right)$.

Proof. For a fixed l, every client j is in C_1 as in Lemma 4.2 to the extent of 1; either once in a type 1, 3, or 4 swap or exactly t^2 times among the type 2 swaps, each of which was counted with weight $\frac{1}{t^2}$. Similarly, every client j is in C_2 to the extent of at most 1 and is in C_3 to the extent of at most 1 over all swaps for this fixed l. Finally, we note each facility i on a cycle in H^* lies in the set B for at most one offset $1 \leq l' \leq k$ for that cycle, so its contribution $2t(f_i^* + f_i)$ to the bound is only counted with weight $\frac{1}{t^2}$ for this fixed l.

Lastly, every facility i not on a cycle in H^* lies in B for at most one index $l, 1 \leq l \leq t^2$ and, then, in only one swap for that particular l. Since we average the bound over indices l between 1 and t^2 then the contribution $2t(f_i^* + f_i)$ of such i is counted with weight only $\frac{1}{t^2}$. ∎

Next we bound the expected facility movement cost change. Let F' be the set of facilities i for which i does

not lie on a cycle in H^* consisting of only facilities i' with $s_{i'} \in S_2$.

LEMMA 4.4. *The expected change in movement cost is at most* $\sum_{i \in F'} (f_i^* - f_i) + O\left(\frac{1}{t}\right)\left(\sum_i f_i^* + \sum_j (c_j^* + c_j)\right)$.

Proof. We consider two cases for a facility i. First, suppose $s_i \in S_0 \cup S_1$. Then when s_i is swapped out in a subtree during the shift from s_i to $o_{\mathsf{next}(i)}$, i is moved to either o_i or to $\sigma(o_i)$. The latter can only occur if $\sigma(o_i) \in S_2$. The total movement change is at most $f_i^* - f_i$ if i is moved to o_i and is at most $f_i^* - f_i + c(o_i, \sigma(o_i))$ if i is moved to $\sigma(o_i)$. Since $\sigma(o_i) \in S_2$ and $o_i = \mathsf{center}(\sigma(o_i))$, by Claim 4.1 we have that $c(o_i, \sigma(o_i)) \leq \frac{1}{t} \sum_{j \in D^*(\mathsf{cap}(\sigma(o_i)))} (c_j^* + c_j)$.

The only other time i is moved is when $s_i \in S_0$ and $\mathsf{end}(i)$ is randomly chosen from $\mathsf{cand}(i')$ for some facility i'. But this happens with probability at most $\frac{1}{t-1}$. In this case, i is shifted from s_i to either o_i or $\sigma(o_i)$. We do not necessarily have $\sigma(o_i) \in S_2$ in this case, but we can use the bound $c(o_i, \sigma(o_i)) \leq f_i^* + f_i$ to bound the expected movement-cost change for i in this case to be at most $\frac{2f_i^*}{t-1}$. Overall, the expected movement cost increase for i is at most $(1 + \frac{2}{t-1}) f_i^* - f_i + \frac{1}{t-1} \sum_{j \in D^*(\mathsf{cap}(\sigma(o_i)))} (c_j^* + c_j)$.

Next, we consider the case $s_i \in S_2$. Let $\mathsf{center}(s_i) = o_{i'}$. When the swap consisting of i and all components C rooted at $\mathsf{cap}(s_i) \setminus \{o_{i'}\}$ is performed, i is moved from s_i to $o_{i'}$ unless i lies on a shift path during that swap, in which case it is moved like in the shift. Since $s_i \in S_2$, we have $c(s_i, o_{i'}) \leq \frac{1}{t} \sum_{j \in D^*(\mathsf{cap}(s_i))} (c_j^* + c_j)$. Unless i lies on a cycle with no S_1-locations, that is, $i \notin F'$, i lies between i'' and $\mathsf{next}(i'')$ for exactly one i'', and $\mathsf{shift}(s_{i''}, o_{\mathsf{next}(i'')})$ is performed to the extent of 1; this holds even if s_i lies on a shift path during the corresponding type 3 swap involving i. All other times when i is moved, it is due to the same reasons as in the previous case, so the bound on the total change in movement cost for facility i is

$$\left(1 + \frac{2}{t-1}\right) f_i^* - f_i + \frac{1}{t} \sum_{j \in D^*(\mathsf{cap}(s_i))} (c_j^* + c_j) + \frac{1}{t} \sum_{j \in D^*(\mathsf{cap}(\sigma(o_i)))} (c_j^* + c_j).$$

Adding up the appropriate expression for each facility accounts for the expected change in total movement cost. ∎

Proof of Theorem 4.1. By local optimality, the change in total cost for every test swap (counting every random choice) is nonnegative. By averaging over the various swaps, the expected change in total cost is nonnegative, so the sum of the expressions in Lemmas 4.3 and 4.4 is nonnegative. To generate an inequality involving a $-f_i$ term for facilities $i \notin F'$, we sum the bound given by Lemma 3.2 here over all cycles of H^* involving only facilities i with $s_i \in S_2$. This yields $0 \leq \sum_{i \notin F'} \left(-f_i + f_i^* + c(o_i, \sigma(o_i))\right)$, and we can bound $c(o_i, \sigma(o_i))$ by $\frac{1}{t} \sum_{j \in D^*(\mathsf{cap}(\sigma(o_i)))} (c_j^* + c_j)$. Adding this to the inequality that the expected change in total cost is nonnegative gives $\left(1 - O\left(\frac{1}{t}\right)\right)(C + F) \leq \left(3 + O\left(\frac{1}{t}\right)\right) C^* + \left(1 + O\left(\frac{1}{t}\right)\right) F^*$. ∎

5 Extension to the weighted case

The analysis in Section 4 (as also the proof of the 5 approximation) extends easily to the weighted generalization, wherein each facility i has a weight $w_i \geq 0$ and the cost of moving facility i to a location v is now given by $w_i c(i, s)$, to yield the same $\big(3 + o(1)\big)$-approximation guarantee. More specifically, we show that Theorem 4.1 also holds in this weighted setting. With the exception of one small difference in the analysis, this requires only minor changes in the arguments. We discuss these briefly in this section.

Unless otherwise stated, the same notation from Section 4 is used in this section. We emphasize that f_i^* and f_i now refer to the weighted movement of facility i in the global or local optimum, respectively. So, $f_i^* = w_i \cdot c(i, o_i)$ and $f_i = w_i \cdot c(i, s_i)$.

One difference in notation is that the definition of S_1 is slightly revised to this weighted setting: $s_i \in S_1$ if $|\mathsf{cap}(s_i)| > t$ or $0 < |\mathsf{cap}(s_i)| \leq t$ and $|D^*(\mathsf{cap}(s_i))| \leq \max\{w_i, w_{i'}\} \cdot t$ where i' is such that $o_{i'} = \mathsf{center}(s_i)$ (equivalently, (i', i) is a center edge in H). If all facility weights are 1, then this definition of S_1 agrees with the definition in Section 4. Similarly, we say $s_i \in S_2$ if $|\mathsf{cap}(s_i)| \leq t$ and $|D^*(\mathsf{cap}(s_i))| > \max\{w_i, w_{i'}\} \cdot t$. Under these definitions, similar to Claim 4.1, we now have that $w_i \cdot c(s_i, o_{i'}) \leq \frac{1}{t} \sum_{j \in D^*(\mathsf{cap}(s_i))} (c_j + c_j^*)$ (since $c(o_i, \sigma(o_i)) \leq c_j^* + c_j$ for any $j \in D^*(\mathsf{cap}(s_i))$ as before, and $|D^*(\mathsf{cap}(s_i))| > w_i t$).

We consider the same set of test swaps and the same averaging of the inequalities generated by these swaps. When a test swap is performed, we consider the same shifting and reassigning of facilities to generate the inequalities. In most cases, we also move the clients in the same way as before with the exception that if a client j has all of $\sigma(j), \sigma^*(j)$ and $\sigma(\sigma^*(j))$ being closed, then we do not necessarily send it to o_i or s_i where i is such that $o_i = \sigma^*(j)$. This will be discussed in the following lemma.

As in the discussion before Lemma 4.2, we consider a swap involving subtrees $\{T_r\}_{r=1}^k$. Let B be as before, and let B' be the set of facilities i such that i is a leaf

in some T_r or, if the swap is a type-2 swap, that i is the first facility in $I_{l'}$. Note that $i \in B$ if and only if the unique (i, i') arc in H^* is a center arc with $i' \in B'$. We let C_1, C_2, and C_3 also be defined as in Section 4.

LEMMA 5.1. *The expected net change in the client assignment cost for a test swap is at most* $\sum_{j \in C_1}(c_j^* - c_j) + \sum_{j \in C_2} 2c_j^* + \frac{1}{t-1}\sum_{j \in C_3} 2c_j^* + 4t\sum_{i \in B \cup B'}(f_i^* + f_i)$.

Proof. Consider one particular swap. As in the proof of Lemma 4.2, we move $j \in C_1$ to $\sigma^*(j)$ and $j \in C_2 \cup C_3$ to $\sigma(\sigma^*(j))$ and bound their assignment cost change in the same way. As before, it may be that for some of these clients $j \in C_2 \cup C_3$ we have that $\sigma(\sigma^*(j))$ was closed in the swap. For such clients, we do the following slight variant of the reassignment that was done in the proof of Lemma 4.2.

Say (i, i') is the center edge such that $\sigma(\sigma^*(j)) = s_{i'}$ for a client $j \in C_2 \cup C_3$ with $\sigma(\sigma^*(j))$ not being open. If $w_i \geq w_{i'}$, then we send j to either s_i or o_i. As in the proof of Lemma 4.2, one of these must be open and the total cost of moving j from $s_{i'}$ to either s_i or o_i is at most $2c(i, s_i) + 2c(i, o_i)$. Otherwise, if $w_{i'} > w_i$ then we send j to either $o_{i'}$ or $\sigma(o_{i'})$ (one of them must be open). The distance from $s_{i'}$ to either $o_{i'}$ or $\sigma(o_{i'})$ is bounded by $2c(i', s_{i'}) + 2c(i', o_{i'})$.

We conclude by noting that each facility $\hat{i} \in B$ has at most $w_{\hat{i}} \cdot t$ clients sent to either $s_{\hat{i}}$ or $o_{\hat{i}}$ from $\sigma(o_{\hat{i}})$ in the manner just described, since $\sigma(o_{\hat{i}}) \in S_3$. Similarly, each $\hat{i} \in B'$ has at most $w_{\hat{i}} \cdot t$ clients sent to either $o_{\hat{i}}$ or $\sigma(o_{\hat{i}})$ from $s_{\hat{i}}$ in the manner described above, since $s_{\hat{i}} \in S_3$. So, the total client movement charged to $i \in B \cup B'$ this way is at most $4tw_i(c(i, s_i) + c(i, o_i)) = 4tf_i^* + 4tf_i$. ∎

Using the same weighting of the swaps as in Section 4 we get the following bound on the contribution of the client movement cost changes over these swaps. The proof is nearly identical, except we notice that a facility i' lies in the B'-set for various swaps to an extent of at most $\frac{1}{t^2}$ (under this weighting), since the facility i such that (i, i') is a center edge lies in some B-set to an extent of at most $\frac{1}{t^2}$.

LEMMA 5.2. *The expected total client assignment cost change, weighted in the described manner, is at most* $\sum_j 3c_j^* - c_j + O\left(\frac{1}{t}\right)\left(\sum_i (f_i^* + f_i) + \sum_j c_j^*\right)$.

The contribution of the facility movement costs is bounded in essentially the same way as in Lemma 4.4. We just provide the details on how to account for the weights of the facilities. As before, let F' be the set of facilities i that do not lie on a cycle in H^* consisting solely of facilities i' with $s_{i'} \in S_2$.

LEMMA 5.3. *The expected change in movement cost is at most* $\sum_{i \in F'}(f_i^* - f_i) + O\left(\frac{1}{t}\right)\left(\sum_i f_i^* + \sum_j (c_j^* + c_j)\right)$.

Proof. When $\mathsf{shift}(s, o)$ is performed, we move facilities i from s_i to $\mathsf{center}(o_i)$. If this shift was performed during a path swap, then the assignment cost change for a facility i moved in this shift is at most $w_i c(i, o_i) + w_i c(o_i, \sigma(o_i)) - w_i c(i, s_i) \leq 2w_i c(i, o_i) = 2f_i^*$ so the same bound used before applies.

If such a shift was performed along a path in a subtree, then we did not want to bound $c(o_i, \sigma(o_i))$ by $c(i, s_i) + c(i, o_i)$ because we do not want to cancel the contribution of $-c(i, s_i)$ to the bound. However, this only happened when $\sigma(o_i) \in S_2$ so we can the fact that $|D^*(\mathsf{cap}(\sigma(o_i)))|$ is large and that $c(o_i, \sigma(o_i)) \leq c_j^* + c_j$ for any $j \in D^*(\mathsf{cap}(\sigma(o_i)))$. In our setting, as noted earlier, the movement cost $w_i \cdot c(o_i, \sigma(o_i))$ can be bounded by $\frac{1}{t}\sum_{j \in D^*(\mathsf{cap}(\sigma(o_i)))}(c_j^* + c_j)$, which is the same upper bound we used in the unweighted case.

Finally, the only other time we moved a facility was from some $s_i \in S_2$ to $\mathsf{center}(s_i)$. The cost of this move is now $w_i \cdot c(s_i, \mathsf{center}(s_i))$ which can also be bounded by $\frac{1}{t}\sum_{j \in D^*(\mathsf{cap}(s_i))}(c_j^* + c_j)$ using the same argument as in the previous paragraph. So, all bounds for the unweighted facility movement cost increase averaged over the swaps also hold in the weighted case. ∎

Finally, we remark that the same bound for the facility movement cost for facilities on a cycle with only S_2 facilities holds for the weighted case, again using arguments like in the proof of Lemma 5.3 to bound $w_i \cdot c(o_i, \sigma(o_i))$. Thus, the proof of Theorem 4.1 is adapted to prove the following result for the weighted case.

THEOREM 5.1. *The cost of a local optimum solution using p swaps is at most* $3 + O\left(\sqrt{\frac{\log\log p}{\log p}}\right)$ *times the optimum solution cost in weighted instances of mobile facility location.*

References

[1] A. Aggarwal, L. Anand, M. Bansal, N. Garg, N. Gupta, S. Gupta, and S. Jain. A 3-approximation for facility location with uniform capacities. In *Proceedings of the 14th IPCO*, pages 149–162, 2010.

[2] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local search heuristics for k-median and facility location problems. *SIAM Journal on Computing*, 33(3):544–562, 2004.

[3] I. Baev, R. Rajaraman, C. Swamy. Approximation algorithms for data placement problems. *SIAM Journal on Computing*, 37(5): 1499–1516, 2008.

[4] M. Bansal, N. Garg, N. Gupta. A 5-approximation for capacitated facility location. In *Proceedings of the 20th ESA*, pages 133–144, 2012.

[5] D. Chakrabarty and C. Swamy. Improved approximation algorithms for matroid median problems and applications. *Manuscript*, 2012.

[6] M. Charikar and S. Guha. Improved combinatorial algorithms for facility location problems. *SIAM Journal on Computing*, 34(4):803–824, 2005.

[7] M. Charikar, S. Guha, É. Tardos, and D. B. Shmoys. A constant-factor approximation algorithm for the k-median problem. *Journal of Computer and System Sciences*, 65(1):129–149, 2002.

[8] M. Charikar and S. Li. A dependent LP-rounding approach for the k-median problem. In *Proceedings of the 39th ICALP*, pages 194–205, 2012.

[9] F. Chudak and D. Williamson. Improved approximation algorithms for capacitated facility location problems. *Mathematical Programming*, 102(2):207–222, 2005.

[10] E. Demaine, M. Hajiaghayi, H. Mahini, A. Sayedi-Roshkhar, S. Oveis Gharan, and M. Zadimoghaddam. Minimizing movement. *ACM Transactions on Algorithms*, 5(3):2009.

[11] N. Devanur, N. Garg, R. Khandekar, V. Pandit, A. Saberi, and V. Vazirani. Price of anarchy, locality gap, and a network service provider game. In *Proceedings of 1st WINE*, pages 1046-1055, 2005.

[12] Z. Friggstad and M. Salavatipour. Minimizing movement in mobile facility location problems. *ACM Transactions on Algorithms*, 7(3), 2011.

[13] I. Gϕrtz and V. Nagarajan. Locating depots for capacitated vehicle routing. In *Proceedings of the 14th APPROX*, pages 230–241, 2011.

[14] A. Gupta and K. Tangwongsan. Simpler analyses of local search algorithms for facility location. *CS arXiv*, 2008.

[15] M. Hajiaghayi, R. Khandekar, and G. Kortsarz. Local search algorithms for the red-blue median problem. *Algorithmica*, 63(4):795–814, 2012.

[16] K. Jain and V. V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation. *Journal of the ACM*, 48(2):274–296, 2001.

[17] K. Jain, M. Mahdian, E. Markakis, A. Saberi, and V. Vazirani. Greedy facility location algorithms analyzed using dual-fitting with factor-revealing LP. *Journal of the ACM* 50(6):795–824, 2003.

[18] B. Korte and J. Vygen. Facility Location. In *Combinatorial Optimization: Theory and Algorithms*, chapter 22, pages 563–598, Springer-Verlag, 2008.

[19] M. R. Korupolu, C. G. Plaxton, and R. Rajaraman. Analysis of a local search heuristic for facility location problems. *Journal of Algorithms*, 37(1):146–188, 2000.

[20] R. Krishnaswamy, A. Kumar, V. Nagarajan, Y. Sabharwal, and B. Saha. The matroid median problem. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms*, 2011.

[21] S. Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. In *Proceedings of the 38th International Colloquium on Automata Languages and Programming*, pages 77–88, 2011.

[22] M. Mahdian and M. Pál. Universal facility location. In *Proceedings of 11th ESA*, pages 409–421, 2003.

[23] P. Mirchandani and R. Francis, editors. *Discrete Location Theory*. John Wiley and Sons, Inc., New York, 1990.

[24] M. Pál, É. Tardos, and T. Wexler. Facility location with nonuniform hard capacities. In *Proceedings of the 42nd FOCS*, pages 329–338, 2001.

[25] D. B. Shmoys. The design and analysis of approximation algorithms: facility location as a case study. In S. Hosten, J. Lee, and R. Thomas, editors. *Trends in Optimization, AMS Proceedings of Symposia in Applied Mathematics 61*, pages 85–97, 2004.

[26] D. B. Shmoys, É. Tardos, and K. I. Aardal. Approximation algorithms for facility location problems. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 265–274, 1997.

[27] Z. Svitkina and É. Tardos. Facility location with hierarchical facility costs. *ACM Transactions on Algorithms*, 6(2), 2010.

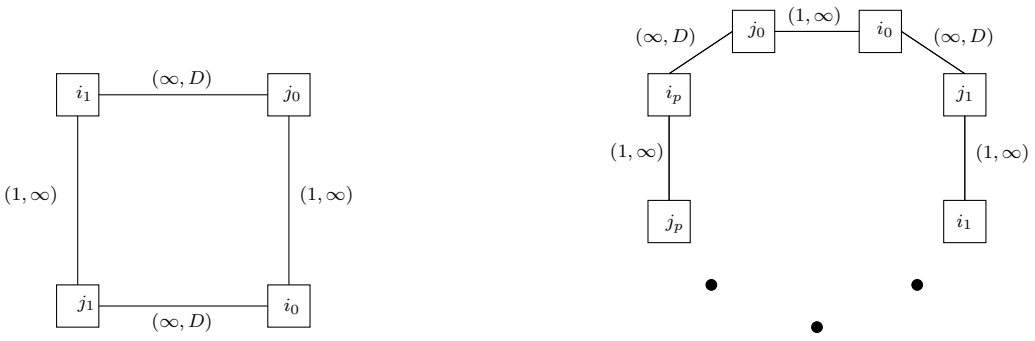
[28] J. Zhang, B. Chen, and Y. Ye. A multi-exchange local search algorithm for the capacitated facility location problem. *Mathematics of Operations Research*, 30:389–403, 2005.

A Bad locality gap with arbitrary facility-movement costs

In this section, we present an example that shows that if the facility-movement costs and the client-assignment costs come from different (unrelated) metrics then the p-swap local-search algorithm has an unbounded locality gap; that is, the cost of a local optimum may be arbitrarily large compared to optimal cost.

We first show a simple example for a single swap case, which we will later generalize for p swaps. Suppose we have two clients j_0, j_1 and two facilities i_0, i_1. Some distances between these clients and facilities are shown in the Fig. 2(a); all other distances are obtained by taking the metric completion. Note that in this example, in order to have a bounded movement cost for facilities, the only option is to have one of i_0, j_1 as a final location of facility i_0 and one of i_1, j_0 as a final location of facility i_1.

As can be seen from the figure, the solution $O = \{i_0, i_1\}$ has total cost 2 (the movement cost is 0 and the client-assignment cost is 2). Now consider the solution $S = \{j_0, j_1\}$ which has a total cost of $2D$ (the movement cost is $2D$ and the client-assignment cost is 0). This is a local optimum since by symmetry if we swap out j_0, then we have to swap in i_1 to have a bounded movement cost, which leads j_0 having assignment cost of ∞. So there is



(a) (b)

Figure 2: Examples showing large locality gap for the cases where local search allows (a) single swaps (b) at most p simultaneous swaps. The label (a, b) of an edge gives client-assignment cost a and the movement cost b of a facility along that edge.

no improving move for solution S, and the locality gap is D.

Now consider the following example (Fig. 2(b)) for local-search with p simultaneous swaps. Suppose we have facility set $\{i_0, i_1, \ldots, i_p\}$ and client set $\{j_0, j_1, \ldots, j_p\}$. The global optimum $O = \{i_0, i_1, \cdots, i_p\}$ has total cost $p + 1$ (facility movement cost is 0 and client-assignment cost is $(p + 1) \cdot 1$) while $S = \{j_0, j_1, \ldots, j_p\}$ is a local optimum whose total cost is $(p + 1) \cdot D$ (facility movement cost is $(p + 1) \cdot D$ and client-assignment cost is 0). Consider any move $swap(X, Y)$. Note that $j_k \in X$ iff $i_{k-1} \in Y$ (where indices are $\bmod(p + 1)$) to ensure bounded movement cost. Let k be such that $j_k \in X$ and $j_{k+1} \notin X$. Then, j_k has an assignment cost of ∞ in the solution $(S \setminus X) \cup Y$. Hence, S is a local optimum.