Continuous Facility Location Algorithms for k-Means and k-Median

Nathan Cordner

October 25, 2019

Abstract

k-means and k-median clustering are classic NP-hard problems that have many applications in computer science. Both problems can be expressed as instances of an uncapacitated facility location (UFL) problem, which seeks to open a minimum cost number of *facilities* to service a set of *clients*. In 2001, Jain and Vazirani introduced an algorithm for UFL that guarantees a 3-approximation for k-median and a 9-approximation for k-means—provided that the algorithm is able to open exactly k facilities. In this paper, we will summarize the work of Archer et al. and Ahmadian et al. to create a *continuous* Jain and Vazirani algorithm that can find solutions for all possible values of k.

1 Introduction

A k-median instance is defined with an integer $k \geq 0$, a set F of potential *facilities*, a set C of *clients*, and a metric d. The objective is to choose a subset of k facilities to *open*, that minimizes the total distance of each client to its nearest open facility. A typical k-means instance can be defined similarly, but its objective is to minimize squared distances from clients to their nearest open facilities. Also, the facility space is often infinite (e.g. a high-dimensional Euclidean space). Under standard discretization techniques [5], we can choose a discrete set of facilities for k-means and only incur an arbitrarily small loss in the final approximation guarantee.

Related to the k-means and k-median problems is the *uncapacitated facility location* (UFL) problem. This problem swaps the hard constraint of opening k facilities for a nonnegative (possibly unique) cost for opening any given facility. The objective is similar in wanting to minimize distances from clients to their nearest open facility, but it also seeks to minimize the total cost of opening facilities.

In 2001, Jain and Vazirani [7] presented a primal-dual approximation algorithm for the UFL problem. They also related the UFL problem to k-means and k-median using a *Lagrangean relaxation*. By allowing each facility to have uniform cost $\lambda \geq 0$, the Jain and Vazirani algorithm will open more facilities when λ is low and fewer facilities when λ is high. Because the algorithm further satisfies a *Lagrange-multiplier preserving* (LMP) property, if it happens to open exactly k facilities then the same solution will also be a 3-approximation for the k-median problem and a 9-approximation for the k-means problem [7; 12].

However, their algorithm does not always allow for opening k facilities for each value of k. Jain and Vazirani also developed a rounding method. Using two UFL solutions (one with fewer than k facilities, and one with more than k), we can combine them into a solution that has exactly k facilities. This rounding process loses a factor of 2 in the approximation guarantee for k-median, giving a 6-approximation algorithm in general. The loss increases to 6 for k-means, creating a 54-approximation algorithm. A $(9+\epsilon)$-approximation algorithm for k-means was given in 2004 by Kanungo et al. [9] for the Euclidean metric. A 16-approximation algorithm for k-means in general metrics was given by Gupta and Tangwongsan [6] in 2008. In 2002, Jain et al. [8] developed a greedy 2-approximation algorithm for the UFL that also satisfies the LMP property for k-median. Using the same rounding technique as before immediately gave a 4-approximation algorithm for k-median.

In 2003, Archer et al. [2] demonstrated a way to adapt the Jain and Vazirani algorithm to satisfy a *continuity* property by opening every possible number of facilities while still maintaining the approximation guarantee. Their algorithm uses a solver for the NP-complete maximum independent set problem, and so runs in exponential time, but they were able to prove an upper bound of 3 for the integrality gap of the k-median problem and 9 for the k-means problem. They asked whether it were possible to find a polynomial time version of the Jain and Vazirani algorithm.

By 2012, Li and Svensson [11] reduced the approximation error in the Jain and Vazirani rounding method down to $(1+\sqrt{3})/2$, leading to an overall $1+\sqrt{3} \approx 2.732$-approximation algorithm for k-median via the Jain et al. algorithm. This rounding was further improved by Byrka et al. [4] in 2014 for an overall 2.675-approximation algorithm. Since both algorithms solve modified instances of their given input, these approximation improvements do not improve the integrality gap in general for k-median.

In 2017, Ahmadian et al. [1] demonstrated a polynomial time, continuous version of the Jain and Vazirani algorithm answering a question posed by Archer et al. They also introduced a variant of the algorithm that guarantees a 2.633-approximation algorithm for k-median and a 6.357-approximation for k-means in the special case where the metric is Euclidean.

As for lower bounds, Jain et al. [8] established that it is hard to approximate the k-median problem within a factor of $1+2/e \approx 1.736$. It can also be shown that the natural linear programming relaxation of the k-median problem has an integrality gap of at least 2 [2; 11]. Lee et al. [10] proved that it is hard to approximate the k-means problem in the Euclidean metric within a factor of 1.0013. For general metrics, the bound is 3.94 [8; 1]. This paper will focus on the work of Archer et al. and Ahmadian et al. in creating a polynomial time version of the Jain and Vazirani algorithm.

2 Background

2.1 UFL and k-Median

The UFL problem was first formulated as an integer program by Balinski [3]. We will present the LP relaxation and its dual here. We define c_{ij} to be distance between facility i and client j. We let f_i be

the cost of opening facility i, x_{ij} be an indicator for whether client j connects to facility i, and y_i be an indicator for whether facility i is open. We state the UFL as:

Primal			**Dual**		
minimize	$\sum_{i\in F, j\in C} c_{ij}x_{ij} + \sum_{i\in F} f_i y_i$		maximize	$\sum_{j\in C} \alpha_j$	
subject to	$\sum_{i\in F} x_{ij} \geq 1,$	$\forall j \in C,$	subject to	$\alpha_j - \beta_{ij} \leq c_{ij},$	$\forall i \in F, j \in C,$
	$y_i - x_{ij} \geq 0,$	$\forall i \in F, j \in C,$		$\sum_{j\in C} \beta_{ij} \leq f_i,$	$\forall i \in F,$
	$x_{ij} \geq 0,$	$\forall i \in F, j \in C,$		$\alpha_j \geq 0,$	$\forall j \in C,$
	$y_i \geq 0,$	$\forall i \in F.$		$\beta_{ij} \geq 0,$	$\forall i \in F, j \in C.$

In the primal, the constraints $\sum_{i\in F} x_{ij} \geq 1$ ensure that each client is assigned to some facility, and the constraints $y_i - x_{ij} \geq 0$ ensure that clients can only be assigned to facilities that appear in the solution. We interpret the dual variables α_j as the overall value derived from client j, and β_{ij} as the amount that client j contributes towards the opening cost of facility i. A client's value is no more than the sum of its distance and contribution to any given facility i (i.e. $\alpha_j \leq c_{ij} + \beta_{ij}$), and contributions from all clients for a particular facility i cannot exceed the opening cost ($\sum_{j\in C} \beta_{ij} \leq f_i$).

The statement of the k-median problem is very similar to the UFL problem. We reuse many of the same variables. We drop the facility opening costs from the primal objective, but introduce a hard constraint of choosing at most k facilities in the primal constraints. We state the k-median primal and dual programs as:

Primal			**Dual**		
minimize	$\sum_{i\in F, j\in C} c_{ij}x_{ij}$		maximize	$\sum_{j\in C} \alpha_j - \lambda k$	
subject to	$\sum_{i\in F} x_{ij} \geq 1,$	$\forall j \in C,$	subject to	$\alpha_j - \beta_{ij} \leq c_{ij},$	$\forall i \in F, j \in C,$
	$y_i - x_{ij} \geq 0,$	$\forall i \in F, j \in C,$		$\sum_{j\in C} \beta_{ij} \leq \lambda,$	$\forall i \in F,$
	$\sum_{i\in F} -y_i \geq -k,$			$\alpha_j \geq 0,$	$\forall j \in C,$
	$x_{ij} \geq 0,$	$\forall i \in F, j \in C,$		$\beta_{ij} \geq 0,$	$\forall i \in F, j \in C,$
	$y_i \geq 0,$	$\forall i \in F.$		$\lambda \geq 0.$	

Now suppose that in the initial instance of the UFL problem we set each facility cost $f_i = \lambda \geq 0$. We can consider a new objective for the k-median problem by replacing the term $\sum_{i\in F} f_i y_i$ with the *penalty term* $\sum_{i\in F} \lambda y_i - \lambda k$:

$$\text{minimize} \sum_{i\in F, j\in C} c_{ij}x_{ij} + \lambda \sum_{i\in F} y_i - \lambda k.$$

We can then remove the hard constraint of opening at most k facilities from the primal, as well as the constraint $\lambda \geq 0$ from the dual. This yields the the *Lagrangean relaxation* of the k-median problem. The constraints of this primal and dual relaxation exactly match the primal and dual constraints for the UFL problem. This relaxation also favors solutions that do not open more than k facilities.

2.2 UFL and k-means

The general k-means problem is defined over a subset of points (clients) C of a metric space (X, d). We want to find k *centers* within X that minimize the total squared distance between each point of C to its nearest center.

Applying a discretization technique, such as the one proposed by Feldman et al. [5], we can choose a finite set $F \subseteq X$ of potential centers (facilities) and only incur an arbitrarily small loss in the approximation guarantee. From now on we will refer to the k-means problem by its discrete version.

Now the only difference between k-median and k-means is the use of squared distances in k-means. Thus k-means can be modeled with the same primal and dual LPs (substituting the values for c_{ij}) that we used for UFL and k-median.

2.3 The Jain and Vazirani Algorithm for UFL

The Jain and Vazirani (JV) algorithm builds a feasible solution for the UFL problem by maintaining a feasible solution to the dual LP. The algorithm continues to increase the values of the dual variables

until each of its constraints go tight. The algorithm proceeds in two stages. The initial dual feasible solution begins with each α_j and β_{ij} set to 0.

Stage 1: each client is initially *unconnected*, and each facility is not *temporarily open*. We start the *time* of the algorithm at $t = 0$. As the time increases, each dual variable α_j also increases at the same rate until one of the following events occurs (breaking ties arbitrarily).

- An *edge goes tight* when $\alpha_j = c_{ij}$ for some facility i and client j.
  - If facility i is not temporarily open, then we start to increase β_{ij} at the same rate as the other dual variables. We say that client j is *contributing* to facility i.
  - If facility i is temporarily open, then we declare client j to be connected. We stop increasing α_j and each β_{hj} for all facilities $h \in F$. We say that facility i is the *connecting witness* for client j.
- Facility i is *paid for* when $\sum_{j \in C} \beta_{ij} = f_i$. We declare facility i to be temporarily open. Each unconnected client j that was contributing to facility i is now declared to be connected (with facility i as the connecting witness), and the corresponding dual variables for each of these clients now stops increasing.

Stage 1 ends when no unconnected clients remain.

Stage 2: construct a graph G with vertices given by the temporarily opened facilities from Stage 1. We allow an edge between facilities $i \neq i'$ if there is some client j that made positive contributions to both (i.e. $\beta_{ij} > 0$ and $\beta_{i'j} > 0$). We return any maximal independent set of facilities for graph G.

2.4 The LMP property

The JV algorithm provides a 3-approximation solution to the UFL problem. Given some UFL instance and solution, weak duality guarantees that

$$\sum_{j \in C} \alpha_j \leq \sum_{i \in F, j \in C} c_{ij} x_{ij} + \sum_{i \in F} f_i y_i.$$

A pair of primal and dual solutions are optimal if this equation becomes an equality. The primal and dual solutions generated by the JV algorithm satisfy

$$\sum_{i \in F, j \in C} c_{ij} x_{ij} + \sum_{i \in F} f_i y_i \leq 3 \sum_{j \in C} \alpha_j,$$

which shows that the primal solution is no worse than 3 times the optimal solution. Jain and Vazirani demonstrated the slightly stronger result

$$\sum_{i \in F, j \in C} c_{ij} x_{ij} + 3 \sum_{i \in F} f_i y_i \leq 3 \sum_{j \in C} \alpha_j.$$

By replacing each f_i by $\lambda \geq 0$, we see that

$$\sum_{i \in F, j \in C} c_{ij} x_{ij} \leq 3 \left(\sum_{j \in C} \alpha_j - \lambda \sum_{i \in F} y_i \right).$$

If $\sum_{i \in F} y_i$ happens to equal k, then the inner part of the right-hand side of the last equation becomes $\sum_{j \in C} \alpha_j - \lambda k$. The left-hand side is now the primal objective of the k-median problem, and the right-hand side resembles the dual objective of the k-median problem. Thus a solution to the UFL problem using the JV algorithm that opens k facilities is immediately a 3-approximate solution for the k-median problem. For k-means, we obtain the weaker guarantee of a 9-approximate solution.

We note that the JV algorithm satisfies the following *Lagrange-multiplier preserving* (LMP) property when then facility opening costs are set to a uniform value λ.

Definition 2.1. A ρ-approximation algorithm is LMP for the UFL problem with opening costs $\lambda \geq 0$ if it returns a primal and dual solution that satisfies

$$\sum_{i \in F, j \in C} c_{ij} x_{ij} \leq \rho \left(\sum_{j \in C} \alpha_j - \lambda \sum_{i \in F} y_i \right).$$

Though better approximation algorithms exist for the UFL without the LMP property, solutions from these algorithms that open k-facilities do not guarantee good approximation results for k-means or k-median. The focus of the rest of this paper is to create an approximation algorithm for UFL with the LMP property that can open k facilities for all possible values of k.

3 First Attempt for a Continuous JV Algorithm

In this section we summarize the work of Archer et al. The main result of this work is to create a *continuous* version of the JV algorithm, in order to use it as an immediate solver for the k-median and k-means problems.

3.1 Discontinuity in the JV Algorithm

We state the desired *continuity* property as

Definition 3.1. Given an instance of the UFL problem with uniform facility cost $\lambda \geq 0$. An algorithm that solves the UFL problem is *continuous* if, as λ increases, the total number of facilities opened by the algorithm never jumps by more than 1 at a time.

We would like for a UFL solver to be able to create solutions that open any number of facilities for each value $k = 1, \ldots, m$. If the UFL algorithm satisfies the LMP property, then we immediately obtain solutions to the k-means and k-median problems for each value of k. For the rest of this section, we will focus on how this work relates to the k-median problem.

We know that when $\lambda = 0$, the JV algorithm opens every facility (since there is no cost to open them). As λ increases the number of facilities opened by the JV algorithm diminishes until just one facility remains open. However, the following example shows that the JV algorithm is generally not continuous in its current form.

Example 3.2. Consider a star graph with h arms, each of length 1. At the end of each arm is a client and a facility (at the same location). In the center there is one additional facility. When $\lambda < 1 + \frac{1}{h-1}$, each client pays for its own facility by time λ. The solution opens h facilities, one for each client, and the center facility remains closed. When $\lambda > 1 + \frac{1}{h-1}$, the center facility opens before time λ and each client connects to it. Therefore the solution opens just one facility in this case. The number of opened facilities thus jumps from h down to 1 at the value $\lambda = 1 + \frac{1}{h-1}$.

3.2 Modifying the JV Algorithm

We propose two main ideas to remedy the discontinuity problem:

1. Perturb the data slightly so that no pair of distinct distances c_{ij} and $c_{i'j'}$ (for clients j, j' and facilities i, i') are exactly equal.

2. In Stage 2 of the JV algorithm, always choose a *maximum* (rather than maximal) independent set of facilities.

We pause to note that the modification to the JV algorithm removes the guarantee of a polynomial run time, since it requires a solver for the maximum independent set problem. However, showing that this new algorithm does satisfy the continuity property will still show that the integrality gap for the k-median is at most 3.

We define an *event* of the JV algorithm as either when an edge goes tight, or when a facility gets paid for during Stage 1. We say that an instance of the UFL problem is *degenerate* if there is some point in time where three or more events coincide, or if there are two or more points in time where at least two events coincide. An instance of the k-median problem is said to be degenerate if there is some $\lambda > 0$ that yields a degenerate UFL instance. We note that the since the set of degenerate k-median instances has Lebesgue measure zero, we might as well just focus on the non-degenerate instances.

For a non-degenerate UFL instance, we further define the *trace* of the JV algorithm to be the timeline (according to the values of t) of events encountered in Stage 1. We define λ_0 to be a *critical value* if when $\lambda = \lambda_0$ there is some point in the trace where two events coincide. For the graph G of Stage 2 of the JV algorithm, we let $I(G)$ denote the size of the maximum independent set in G. For a k-median instance, we let $G(\lambda)$ denote the Stage 2 graph for the given value of λ. We state our main result as

Theorem 3.3. *Given a non-degenerate k-median instance. As λ passes through a critical value, $I(G(\lambda))$ changes by at most 1.*

This theorem is directly implied by the following more technical result.

Theorem 3.4. *When λ passes through a critical value, the graph $G(\lambda)$ can change only in one of the following ways:*

1. *a single existing facility is deleted (with its incident edges),*
2. *a single new facility is added, along with edges to one or more cliques of existing facilities,*
3. *a single existing facility gains edges to one clique of facilities, or loses edges to one clique.*

To prove this theorem, we define a new graph $H(\lambda)$ that has one node per client, one node for temporarily opened facility, and an edge between every client j and facility i where $\beta_{ij} > 0$. Thus facilities in $G(\lambda)$ are connected if and only if there is a distance two path between them in $H(\lambda)$. It suffices to show that at a critical value λ, $H(\lambda)$ changes by either adding/deleting one facility (with its incident edges), or by adding/deleting a single client-facility edge.

We now examine the traces of UFL instances. As λ varies, the times of tight edge events remain fixed. The possible differences between traces lie in the timing of the facility opening events. The proof breaks down into cases. The first three cases deal with two consecutive facility events i and i' with no edge events in between.

1. If events i and i' trade places, then $H(\lambda)$ does not change.
2. If events i and i' trade places, but then i disappears from the new trace, we remove i and its incident edges from $H(\lambda)$.
3. If events i and i' trade places, but i happens after at least one edge event c_{ij} has occurred, then we add edge (i, j) to $H(\lambda)$.

The last three cases deal with an edge event $c_{i'j}$ and a facility event i.

4. If events $c_{i'j}$ and i swap to i and then $c_{i'j}$, then one of two things could happen. If $i = i'$, then we remove edge (i, j) from $H(\lambda)$. If $i \neq i'$, then $H(\lambda)$ remains the same.
5. If events i and $c_{i'j}$ swap to become just $c_{i'j}$, then we again remove i and its incident edges from $H(\lambda)$.
6. If events i and $c_{i'j}$ swap (with $i \neq i'$), and i happens later in the trace after at least one other edge event $c_{ij'}$, then $H(\lambda)$ gets one new edge (i, j').

This finishes the proof outline of the theorems. By creating a continuous version of the JV algorithm, we establish an upper bound of 3 for the integrality gap of the natural LP relaxation for the k-median problem. We again note that if the JV algorithm can be made continuous in polynomial time, then we would have a 3-approximation algorithm for the k-median problem.

4 Better Rounding for the JV Algorithm

In this section we summarize the work of Ahmadian et al. related to the k-means and k-median problems.

4.1 The JV(δ) Algorithm

We begin with a few definitions. For a client j, define the set of *neighbors* of j as $N(j) = \{i \in F \mid \alpha_j - c_{ij} > 0\}$. Similarly, we define the set of neighbors of a facility i as $N(i) = \{j \in C \mid \alpha_j - c_{ij} > 0\}$. For a temporarily opened facility in Stage 1 of the JV algorithm, let $t_i = \max_{j \in N(i)} \alpha_j$, and let $t_i = 0$ if $N(i)$ is empty. If $N(i)$ is not empty, then t_i denotes the time that facility i was temporarily opened during Stage 1.

We now introduce a variant of the JV algorithm, called the JV(δ) algorithm. We run Stage 1 of the JV algorithm the same as before, but alter the way we construct the graph G in Stage 2. For JV(δ), we admit an edge between two facilities $i \neq i'$ in G if some client j made positive contributions to both and if the distance between i and i' is less than or equal to $\delta \min\{t_i, t_{i'}\}$.

We first note that if $\delta = \infty$, then the JV(δ) algorithm is exactly the same as the original JV algorithm. In general metrics, this is the best analysis we have so far. But we can do better when considering the Euclidean metric. We note the following results about the LMP property of the JV(δ) algorithm:

Theorem 4.1. *There are choices of $\delta > 0$ where the JV(δ) algorithm satisfies the LMP property with*

1. *$\rho \approx 6.3574$ for k-means in the Euclidean metric, and*
2. *$\rho \approx 2.633$ for k-median in the Euclidean metric.*

Proof. We will give the details of the proof for k-median in the Euclidean metric. Let IS denote the maximal independent set of open facilities chosen by the JV(δ) algorithm, and let $d(j, \text{IS})$ denote the distance of client j to its nearest facility in IS. We will choose a value of δ so that the JV(δ) algorithm returns a solution that satisfies

$$\sum_{j \in C} d(j, \text{IS}) \leq \rho \left(\sum_{j \in C} \alpha_j - \lambda |\text{IS}| \right).$$

To show this, we will prove for each client j that

$$\frac{d(j, \text{IS})}{\rho} \leq \alpha_j - \sum_{i \in N(j) \cap \text{IS}} (\alpha_j - c_{ij}) = \alpha_j - \sum_{i \in \text{IS}} [\alpha_j - c_{ij}]^+, \tag{1}$$

where $[a]^+$ denotes $\max\{a, 0\}$. Note that $\alpha_j - c_{ij} \leq \beta_{ij}$, and for a temporarily opened facility i we have $\sum_{j \in C} \beta_{ij} = \lambda$. Since each facility i in IS was temporarily opened (and thus fully paid for), then $\sum_{j \in C} [\alpha_j - c_{ij}]^+ = \lambda$. Summing Equation 1 over all clients gives the desired result.

For a fixed client j, let $S := N(j) \cap \text{IS}$, and let $s = |S|$. We want to find a bound on the value of s. Using a centroid property of squared distances in Euclidean space, we have

$$\sum_{i \in S} d(i, j)^2 \geq \frac{1}{2s} \sum_{i \in S} \sum_{i' \in S} d(i, i')^2 = \frac{1}{2s} \sum_{i \in S} \sum_{i' \neq i \in S} d(i, i')^2.$$

Now each pair of facilities i, i' are in IS, and so $d(i, i') > \delta \min\{t_i, t_{i'}\}$. Also, $\min\{t_i, t_{i'}\} \geq \alpha_j$, since both $i, i' \in N(j)$. So

$$\frac{1}{2s} \sum_{i \in S} \sum_{i' \neq i \in S} d(i, i')^2 > \frac{1}{2s} \sum_{i \in S} \sum_{i' \neq i \in S} (\delta \alpha_j)^2 = \frac{1}{2s} s(s-1) \delta^2 \alpha_j^2 = \frac{1}{2}(s-1) \delta^2 \alpha_j^2.$$

Now $s\alpha_j^2 \geq \sum_{i \in S} d(i, j)^2$, so we end up with $s > (1/2)(s-1)\delta^2$ and thus $s < \delta^2/(\delta^2 - 2)$. Our choice of δ will bound the value of s. We will consider three cases: $s = 0$, $s = 1$, and $s > 1$.

Recall that for a client j and its witness facility i, we have $\alpha_j \geq t_i$. And for any $j' \in N(i)$, we have $t_i \geq \alpha_{j'}$. For $s = 0$, let i_1 be the witness of j. Thus $\alpha_j \geq t_{i_1}$, and $\alpha_j \geq d(j, i_1)$. Now either $i_1 \in \text{IS}$, in which case $d(j, \text{IS}) \leq d(j, i_1)$, or there is some $i_2 \in \text{IS}$ such that edge (i_1, i_2) is in the graph G from Stage 2 of the JV(δ) algorithm. In this case,

$$d(j, \text{IS}) \leq d(j, i_1) + d(i_1, i_2) \leq d(j, i_1) + \delta t_{i_1} \leq (1 + \delta) \alpha_j.$$

Thus we must have $\rho \geq 1 + \delta$.

For $s = 1$, let $S = \{i^*\}$. Then for $\rho \geq 1$,

$$\frac{d(j, \text{IS})}{\rho} \leq d(j, \text{IS}) \leq d(j, i^*) = \alpha_j - (\alpha_j - d(j, i^*)) = \alpha_j - \sum_{i \in N(j) \cap \text{IS}} (\alpha_j - d(j, i^*)).$$

Now consider $s > 1$. We have

$$\begin{aligned} s\alpha_j &= \sum_{i \in S} d(i, j) + \sum_{i \in S} (\alpha_j - d(i, j)) \\ &\geq \frac{1}{s-1} \sum_{\{i, i'\} \subseteq S} d(i, i') + \sum_{i \in S} (\alpha_j - d(i, j)) \\ &\geq \frac{1}{s-1} \binom{s}{2} \delta \alpha_j + \sum_{i \in S} (\alpha_j - d(i, j)) \end{aligned}$$

since $d(i,i') > \delta \min\{t_i, t_{i'}\} \geq \delta\alpha_j$ for $i, i' \in S$ (thus i, i' are not adjacent in the graph G). Rearranging the inequality yields $\left((s-1) - \frac{1}{s-1}\binom{s}{2}\delta\right)\alpha_j \geq -\alpha_j + \sum_{i\in S}(\alpha_j - d(i,j))$. Using $\alpha_j \geq d(j, \text{IS})$, and multiplying both sides by -1 yields

$$\left(\frac{1}{s-1}\binom{s}{2}\delta - (s-1)\right) d(j, \text{IS}) \leq \alpha_j - \sum_{i\in S}(\alpha_j - d(i,j)).$$

We can quickly generate bounds on ρ for different values of s. For example, $s = 2$ yields $\rho \geq 1/(\delta - 1)$, $s = 3$ yields $\rho \geq 1/((3/2)\delta - 2)$, and $s = 4$ yields $\rho \geq 1/(3\delta - 3)$. Recall that $s < \delta^2/(\delta^2 - 2)$. We want to choose a value of δ that minimizes the value of ρ. We achieve an optimal value by picking $\delta = \sqrt{8/3}$. Then $s < 4$, and $\rho = 1 + \sqrt{8/3} \approx 2.633$ is the max of $\delta + 1, 1/(\delta - 1)$, and $1/((3/2)\delta - 2)$. □

As with Archer et al., before we continue we need to scale the distances of the input instance. These adjustments will help with the later analysis, and satisfy the following guarantee:

Lemma 4.2. *We can scale the distances between each client j and facility i to satisfy $1 \leq d(i,j) \leq n^6$, where $n = |C|$, and lose only a factor of $1 + 100/n^2$ in the approximation guarantee.*

4.2 A Quasi-Polynomial Time Algorithm

We will consider some properties of the dual solutions constructed by the JV(δ) algorithm. We seek to create a $(\rho + O(\epsilon))$-approximation algorithm for k-means/k-median. Let $\alpha = \{\alpha_j\}_{j\in C}$. We have two important definitions:

1. We say that α is *good* if for every $j \in C$ there is a temporarily opened facility i such that $(1 + \sqrt{\delta} + \epsilon)\sqrt{\alpha_j} \geq d(j,i) + \sqrt{\delta t_i}$.
2. Two solutions α, α' are *close* if $|\alpha'_j - \alpha_j| \leq 1/n^2$ for all $j \in C$.

Our goal now is to generate a list of parameter values $\lambda = 0, \epsilon_z, 2\epsilon_z, \ldots, L\epsilon_z$ with corresponding good dual solutions $\alpha^{(0)}, \alpha^{(1)}, \ldots, \alpha^{(L)}$ such that each consecutive pair $\alpha^{(l)}, \alpha^{(l+1)}$ of dual solutions is close. We let ϵ_z be a small step size, and $L = n^{O(\epsilon^{-1}\log n)}$. Thus the list is of quasipolynomial length. We interpolate between each pair of consecutive dual solutions, so the number of open facilities doesn't change by more than one at any step. Thus we will find a solution that opens exactly k facilities.

To begin, fix $\epsilon_z = n^{-3-30\log_{1+\epsilon} n}$ and $L = 4n^7 \cdot \epsilon_z^{-1} = n^{O(\epsilon^{-1}\log n)}$. For any real value $x \in \mathbb{R}$, we define $B(x) = 0$ if $x < 1$ and $B(x) = 1 + \lfloor \log_{1+\epsilon}(x) \rfloor$ if $x \geq 1$. We say that $B(x)$ is the index of the *bucket* that contains x. We note that the α-values for any two clients in the same bucket differ by at most $1 + \epsilon$.

For each solution $\alpha = \alpha^{(l)}$, we would like the following invariant to hold: every client $j \in C$ has a tight edge to a temporarily open facility $w(j) \in F$ (its witness) such that $B(t_{w(j)}) \leq B(\alpha_j)$. Under this invariant, every client j will have some facility $i = w(j)$ where $\sqrt{\alpha_j} \geq d(i,j)$ and $\sqrt{(1+\epsilon)\delta\alpha_j} \geq \sqrt{\delta t_i}$. Then

$$(1 + \sqrt{\delta} + \epsilon)\sqrt{\alpha_j} \geq \left(1 + \sqrt{(1+\epsilon)\delta}\right)\sqrt{\alpha_j} \geq d(i,j) + \sqrt{\delta t_i},$$

which implies that α is good.

4.2.1 QuasiSweep

We use the QuasiSweep procedure to generate a list of good, close dual solutions. Let $\alpha^{(0)}$ be the dual solution for $\lambda = 0$ where each client is assigned to its nearest facility. QuasiSweep will take a good dual solution for parameter λ and generate a new good dual solution for parameter $\lambda + \epsilon_z$.

Begin with good dual solution α for parameter λ, and raise the facility price to $\lambda + \epsilon_z$. Thus no facility is currently paid for. Start with a set of active clients $A = \emptyset$, and a threshold $\theta = 0$. We increase θ continuously:

- Whenever $\theta = \alpha_j$, we add j to A.
- We remove j from A whenever j has a tight edge to some temporarily open facility i where $B(\alpha_j) \geq B(t_i)$ (this could happen as soon as j enters A).
- To prevent facilities getting too many contributions, we also decrease each α_j with $B(\alpha_j) > B(\theta)$ at a rate of $|A|$ times the rate that θ is increasing.

We stop increasing θ once every client has been added and removed from A. We output the augmented dual solution α' for parameter $\lambda + \epsilon_z$.

By construction, α' is good. We note that once a facility is temporarily opened, contributions to it do not increase any more. Every client j, once removed from A, will have a witness $w(j)$ for the rest of the algorithm. Thus $B(t_{w(j)}) \leq B(\alpha_j)$, which satisfies the invariant (implying that α' is good).

We also need to show that each consecutive pair of solutions $\alpha^{(l)}$ and $\alpha^{(l+1)}$ is close. That is, we want all clients $j \in C$ to have $|\alpha_j^{(l)} - \alpha_j^{(l+1)}| \leq 1/n^2$. By fixing our distances, we note that the largest possible α-value is at most $(\lambda+\epsilon_z)+n^6 \leq L\epsilon_z+n^6 = 4n^7+n^6 \leq 5n^7$. Thus $B(\alpha_j) \leq 1+\lfloor \log_{1+\epsilon}(5n^7) \rfloor \leq 10\log_{1+\epsilon}(n)$ for any client j and dual solution α.

We claim that any α_j increases by at most $\epsilon_z n^{3b}$ while $B(\theta) \leq b$. Proceeding by induction on $b = 0, 1, \ldots, 10\log_{1+\epsilon}(n)$, we note that this is trivially true for $b = 0$ since by the way we have fixed our distances no clients have $B(\alpha_j) = 0$. Now assume that the claim holds for $0, 1, \ldots, b-1$. The proof proceeds by showing that if α_j was decreased during QuasiSweep for $B(\theta) < b$, then it needs to increase by at most $\epsilon_z n^{3b-2}$ to return it to its original level. And if α_j continues to increase, it can only increase by at most $(n-1)\epsilon_z n^{3b-2} + \epsilon_z$ before its witness facility becomes fully paid for. The total increase of α_j is thus bounded by $\epsilon_z n^{3b}$.

Therefore the increase of $\alpha_j^{(l)}$ to $\alpha_j^{(l+1)}$ is no more than $\epsilon_z n^{3\cdot 10\log_{1+\epsilon}(n)} = n^{-3-30\log_{1+\epsilon}(n)+30\log_{1+\epsilon}(n)} = n^{-3} \leq n^{-2}$. Some α values may decrease, but the proof of the claim also shows that this is no more than n times the maximum α value increase. Thus $\alpha_j^{(l+1)} - \alpha_j^{(l)}$ is also bounded by n^{-2} in this case.

4.2.2 QuasiGraphUpdate

To make sure we return a solution that opens exactly k facilities, we introduce the QuasiGraphUpdate procedure. Each dual solution $\alpha^{(l)}$ in our sequence has a graph $G^{(l)}$ of facilities generated after Stage 1 of the JV(δ) algorithm. We also recall the graph $H^{(l)}$ that is a bipartite graph of temporarily open facilities on one side and clients on the other, with edges between if some client positively contributes to a facility. Between every two good dual solutions, we generate polynomially many intermediate graphs $G^{(l)} = G^{(l,0)}, G^{(l,1)}, \ldots, G^{(l,p_l)} = G^{(l+1)}$. We generate a list of maximal independent sets of these graphs $\mathrm{IS}^{(l)} = \mathrm{IS}^{(l,0)}, \mathrm{IS}^{(l,1)}, \ldots, \mathrm{IS}^{(l,p_l)} = \mathrm{IS}^{(l+1)}$ such that the size of the independent set decreases by no more than one after each step. We return the first independent set we encounter of size k.

The QuasiGraphUpdate algorithm takes $G^{(l)}, G^{(l+1)}$, and $\mathrm{IS}^{(l)}$ (of size greater than k) as input. Create distinct copies of the facilities found in $G^{(l)}$ and $G^{(l+1)}$, and put them into the disjoint sets $V^{(l)}$ and $V^{(l+1)}$. Create a new bipartite graph G' with facilities $V^{(l)} \cup V^{(l+1)}$ on one side, and all clients on the other. Create an edge from client j to facility $i \in V^{(l)}$ if (j,i) is present in $H^{(l)}$, and an edge from j to $i \in V^{(l+1)}$ if (j,i) is present in $H^{(l+1)}$. Generate the graph $G^{(l,1)}$ on G', where the induced subgraph of $G^{(l,1)}$ on $V^{(l)}$ equals $G^{(l)} = G^{(l,0)}$. We note that $\mathrm{IS}^{(l)} = \mathrm{IS}^{(l,0)}$ is already an independent set for $H^{(l,1)}$, and we greedily extend it to a maximal independent set $\mathrm{IS}^{(l,1)}$ for $H^{(l,1)}$. We generate the next graphs in the sequence by removing the facilities in $V^{(l)}$ from G' one by one. After $p_l = |V^{(l)}|$ steps, we arrive at $G^{(l,p_l)} = G^{(l+1)}$. The size of each intermediate IS is reduced by at most one at each step.

If we happen to encounter a maximal independent set of size k that corresponds directly to a dual solution (e.g. some $\mathrm{IS}^{(l)}$), then previous proofs will guarantee our approximation result. The tricky part comes if we happen to return some $\mathrm{IS} = \mathrm{IS}^{(l,s)}$ where $1 \leq s \leq p_l$.

Let $G = G^{(l,s)}$, and let H denote the "hybrid" client-facility graph that G is based on. We form a hybrid solution α by setting $\alpha_j = \min(\alpha_j^{(l)}, \alpha_j^{(l+1)})$. We note that $\alpha \leq \alpha^{(l)}$ is a feasible solution for the dual with parameter $\lambda = l \cdot \epsilon_z$. Since $\alpha^{(l)}$ and $\alpha^{(l+1)}$ are close, we have $\alpha_j \geq \alpha^{(l)} - \frac{1}{n^2}$ and $\alpha_j \geq \alpha^{(l+1)} - \frac{1}{n^2}$. For each client j, we define the set $S_j \subseteq \mathrm{IS}$ where $i \in S_j$ if $\alpha_j > d(j,i)^2$. We have the following lemmas:

Lemma 4.3. *For any client j with $|S_j| > 0$, $d(j, IS)^2 \leq \rho \cdot \left(\alpha_j - \sum_{i \in S_j} \beta_{ij}\right)$.*

Lemma 4.4. *For any client j with $|S_j| = 0$, $d(j, IS)^2 \leq (1+5\epsilon)\rho \cdot \alpha_j$.*

Lemma 4.5. *For any $i \in IS$, $\sum_{j \in C} \beta_{ij} \geq \lambda - \frac{1}{n}$.*

Combining the first two lemmas, we see that

$$\sum_{j \in C} d(j, \mathrm{IS})^2 \leq (1+5\epsilon)\rho \cdot \sum_{j \in C} \left(\alpha_j - \sum_{i \in S_j} \beta_{ij}\right).$$

By the third lemma,

$$\sum_{j \in C} \left(\alpha_j - \sum_{i \in S_j} \beta_{ij} \right) \leq \sum_{j \in C} \alpha_j - |\text{IS}| \left(\lambda - \frac{1}{n} \right) = \sum_{j \in C} \alpha_j - k\lambda + \frac{k}{n} \leq \text{OPT}_k + 1,$$

where OPT_k denotes the value of the optimal solution with k facilities. In all, we have that $\sum_{j \in C} d(j, \text{IS})^2 \leq (1 + 5\epsilon)\rho \cdot \text{OPT}_k$. This gives our $\rho + \epsilon$ guarantee for k-means and k-median.

We note that the number of dual solutions generated is quasipolynomial in length, making this a quasipolynomial time algorithm.

4.3 A Polynomial Time Algorithm

The main bottleneck of the previous algorithm is the number of dual solutions required to guarantee the approximation result. We will now strive to reduce this list to a polynomial number of dual solutions.

Recall that in the original UFL problem we allowed each facility to have its own price. We will relax the uniform facility costs that we have been using to letting each facility have cost $z_i \in \{\lambda, \lambda + 1/n\}$ for a given parameter λ. We will designate a set $F_S \subseteq F$ of *special facilities* to open, even if they are not fully paid for. For each $i \in F_S$, we assign a set of *special clients* $C_S(i) \subseteq C$ that are allowed to pay for i. For each $i \in F_S$, we define the time $\tau_i = \max_{j \in N(i) \cap C_S(i)} \alpha_j$. For each $i \in F \setminus F_S$, we define $\tau_i = t_i = \max_{j \in N(i)} \alpha_j$ as before. We say that $\tau_i = 0$ if the max is performed over an empty set. We require the total payments for all facilities in F_S to be about equal to $\lambda |F_S|$.

Like the good dual solutions of the previous section, we expect most clients j to satisfy $(1 + \sqrt{\delta} + 10\epsilon)\sqrt{\alpha_j} \geq d(j, i) + \sqrt{\delta \tau_i}$. However, we will allow for a set of "bad" clients C_B that only satisfy $6\sqrt{\alpha_j} \geq d(j, i) + \sqrt{\delta \tau_i}$. We will ensure that the total contribution of bad clients to the overall solution is small. A dual solution (α, z, F_S, C_S) that satisfies these constraints is called λ-*roundable*.

The algorithm will maintain a base facility cost λ, a current λ-roundable solution $S^{(0)}$, and a corresponding integral solution $\text{IS}^{(0)}$. We will again iterate over a list of facility costs $\lambda = 0, 1 \cdot \epsilon_z, \ldots, L \cdot \epsilon_z$ where $L = 4n^7 \cdot \epsilon_z^{-1}$. However, now we will let $\epsilon_z = n^{-O(1)}$. We will use a new procedure called RaisePrice to increase the costs of a given facility i to $\lambda + \epsilon_z$, and produce a close sequence of roundable solutions $S^{(1)}, \ldots, S^{(q)}$. The GraphUpdate procedure is similar to the QuasiGraphUpdate procedure from the previous section.

To construct the first $S^{(0)}$, we begin with $\lambda = 0$, $F_S = \emptyset$, and $\alpha_j = 0$ for each client j. We increase all α-values uniformly. We stop increasing α_j if j gains a tight edge to some facility i, or $2\sqrt{\alpha_j} \geq d(j, j') + 6\sqrt{\alpha_j}$ for some other client j'. We initialize the current integral solution $\text{IS}^{(0)} = F$. We loop over $\lambda = 0, 1 \cdot \epsilon_z, 2 \cdot \epsilon_z, \ldots, L \cdot \epsilon_z$:

- While some facility i still has cost λ:
  - Call RaisePrice on i and produce a sequence $S^{(1)}, \ldots, S^{(q)}$ of λ-roundable solutions
  - For $l = 0$ to $q - 1$:
    - Call GraphUpdate on $S^{(l)}$, $S^{(l+1)}$ to produce a sequence $\text{IS}^{(l,0)}, \ldots, \text{IS}^{(l,p_l)}$
    - if one of these has size k, then return it
    - else, set $\text{IS}^{(l+1)} = \text{IS}^{(l,p_l)}$
  - Reset $S^{(0)} = S^{(q)}$, $\text{IS}^{(0)} = \text{IS}^{(q)}$

We note that the list of facility costs now has polynomial length, and the sequences produced by RaisePrice and GraphUpdate are also of polynomial length. Each process runs in polynomial time, which guarantees a polynomial runtime overall. Since the GraphUpdate procedure is similar to the QuasiGraphUpdate procedure, we will focus on describing how RaisePrice works.

4.3.1 RaisePrice

We will first introduce a few definitions:

- A client j is *witnessed* if it has a tight edge to a facility i with $B(\alpha_j) \geq B(t_i)$. Then i is a *witness* for j, and we have $(1 + \epsilon)\alpha_j \geq t_i$.
- A client j is *stopped* if $2\sqrt{\alpha_j} \geq d(j, j') + 6\sqrt{\alpha_{j'}}$ for some other client j'. If j' stops j, then $\sqrt{\alpha_j} \geq 3\sqrt{\alpha_{j'}}$ and $\alpha_j \geq 9\alpha_{j'}$.

- A client j is *undecided* if it is neither witnessed nor stopped.

We will also require the following invariants to hold throughout the RaisePrice procedure:

1. (Feasibility): For all $j \in C$, $\alpha_j \geq 1$. For all $i \in F$, $\sum_{j \in C} \beta_{ij} \leq z_i$.
2. (No strict containment): For any two clients $j, j' \in C$, $\sqrt{\alpha_j} \leq d(j, j') + \sqrt{\alpha_{j'}}$.
3. (Completely decided): Every client in $(\alpha^{(0)}, z^{(0)})$ is decided.

RaisePrice takes the following as input: $(\alpha^{(0)}, z^{(0)})$, a λ-roundable solution satisfying the three invariants; $\text{IS}^{(0)}$, a maximal independent set of conflict graph $G^{(0)}$ of $(\alpha^{(0)}, z^{(0)})$; i^+, a facility whose price z_{i^+} is being raised from λ to $\lambda + \epsilon_z$. As output, RaisePrice produces a sequence $S^{(1)} = (\alpha^{(1)}, z^{(1)}, F_S^{(1)}, C_S^{(1)}), \ldots, (\alpha^{(q)}, z^{(q)}, F_S^{(q)}, C_S^{(q)})$ of close λ-roundable solutions where all clients are decided in $S^{(q)}$.

To begin, assume $|\text{IS}^{(0)}| \geq k$. For facility i^+, we raise z_{i^+} to $z_{i^+} + \epsilon_z$. Some clients using i^+ as a witness may now be undecided. These are clients that are note stopped, and have no witness except i^+ in $(\alpha^{(0)}, z^{(0)})$. We let $U^{(0)}$ be the set of initially undecided clients. We maintain a set U of currently undecided clients, and repair our solution in stages by calling the Sweep procedure. Each repair stage s has a threshold θ_s, and makes multiple calls to Sweep. Each call produces a new solution α. RaisePrice constructs a roundable solution $S = (\alpha, z, F_S, C_S)$ from each new α, and returns the sequence $S^{(1)}, \ldots, S^{(q)}$ in the order it was constructed. We finish when we reach a solution where all clients are decided.

The analysis of RaisePrice shows that it only requires $O(\log n)$ stages, where each stage makes polynomially many calls to Sweep.

4.3.2 Sweep

The Sweep procedure is an adaptation of QuasiSweep, and only requires a few adjustments. We begin with a the initial solution $(\alpha^{(0)}, z^{(0)})$ given to RaisePrice, the previously constructed solution α, a threshold θ_s, and the set U of currently undecided clients. We set $A = \emptyset$, and $\theta = 0$. We increase θ continuously. Whenever $\theta = \alpha_j$, we add j to A. We remove j from A whenever:

- j has some witness i;
- j is stopped by some client j';
- $j \in U$, and α_j is ϵ_z larger that its value at the start of Sweep;
- $\alpha_j \geq \theta_s$ and $\alpha_j \geq \alpha_j^{(0)}$;
- There is a client j' already removed from A such that $\sqrt{\alpha_j} \geq d(j, j') + \sqrt{\alpha_{j'}}$.

We say that facility i is *potentially tight* if either there exists $j \in N(i)$ with $\alpha_j > \alpha_j^{(0)}$, or for all $j \in N^{(0)}(i)$ we have $\alpha_j \geq \alpha_j^{(0)}$. We decrease $\alpha_{j'}$ (at the rate of $|A|$ times the rate that θ is increasing) if $B(\alpha_{j'}) > B(\theta)$ and if for some potentially tight facility i with $j' \in N(i)$ and $|N(i) \cap A| \geq 1$ we have $\alpha_{j'} = t_i$. We stop increasing θ once every client has been added and removed from A. We output the augmented dual solution α'.

4.3.3 Analysis

We observe that all parts of the new algorithm are polynomial in length, and have polynomial running times. Though the run time is not good (the outermost loop is $O(n^8)$), the algorithm does indeed run in polynomial time. We can also show that the solutions produced satisfy the $\rho + \epsilon$ approximation bound.

5 Conclusion

Though we have now constructed a continuous polynomial time algorithm for k-means and k-median, it would be interesting to know if we can find such an algorithm that has an efficient running time. Further, it would be interesting to explore possibilities to adapt other UFL solvers with better LMP guarantees (such as the Jain et al. 2-approximation algorithm for k-median). These algorithms may require completely different strategies to make them continuous.

References

[1] S. Ahmadian, A. Norouzi-Fard, O. Svensson, and J. Ward. Better guarantees for k-means and euclidean k-median by primal-dual algorithms. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 61–72. IEEE, 2017.

[2] A. Archer, R. Rajagopalan, and D. B. Shmoys. Lagrangian relaxation for the k-median problem: new insights and continuity properties. In *European Symposium on Algorithms*, pages 31–42. Springer, 2003.

[3] M. L. Balinski. On finding integer solutions to linear programs. *Proceedings of the IBM Scientific Computing Symposium on Combinatorial Problems*, pages 225–248, 1966.

[4] J. Byrka, T. Pensyl, B. Rybicki, A. Srinivasan, and K. Trinh. An improved approximation for k-median, and positive correlation in budgeted optimization. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 737–756. SIAM, 2014.

[5] D. Feldman, M. Monemizadeh, and C. Sohler. A ptas for k-means clustering based on weak coresets. In *Proceedings of the twenty-third annual symposium on Computational geometry*, pages 11–18. ACM, 2007.

[6] A. Gupta and K. Tangwongsan. Simpler analyses of local search algorithms for facility location. *arXiv preprint arXiv:0809.2554*, 2008.

[7] K. Jain and V. V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *Association for Computing Machinery*, 48(2):274–296, 2001.

[8] K. Jain, M. Mahdian, and A. Saberi. A new greedy approach for facility location problems. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 731–740. ACM, 2002.

[9] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. A local search approximation algorithm for k-means clustering. *Computational Geometry*, 28(2-3):89–112, 2004.

[10] E. Lee, M. Schmidt, and J. Wright. Improved and simplified inapproximability for k-means. *Information Processing Letters*, 120:40–43, 2017.

[11] S. Li and O. Svensson. Approximating k-median via pseudo-approximation. *SIAM Journal on Computing*, 45(2):530–547, 2016.

[12] D. P. Williamson and D. B. Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011.