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

Nathan Cordner

Boston University

4 November 2019

Clustering Problems

Given discrete subsets C and F of metric space (X, d)

Clustering Problems

Given discrete subsets C and F of metric space (X, d)

- k-Median problem:

Clustering Problems

Given discrete subsets C and F of metric space (X, d)

- k-Median problem:
  - Choose k centers in F

Clustering Problems

Given discrete subsets C and F of metric space (X, d)

- k-Median problem:
  - Choose k centers in F
  - Minimize distances from each point in C to its nearest center

Clustering Problems

Given discrete subsets C and F of metric space (X, d)

- k-Median problem:
  - Choose k centers in F
  - Minimize distances from each point in C to its nearest center
- k-Means problem:

Clustering Problems

Given discrete subsets C and F of metric space (X, d)

- k-Median problem:
  - Choose k centers in F
  - Minimize distances from each point in C to its nearest center
- k-Means problem:
  - Choose k centers within X

Clustering Problems

Given discrete subsets C and F of metric space (X, d)

- k-Median problem:
  - Choose k centers in F
  - Minimize distances from each point in C to its nearest center
- k-Means problem:
  - Choose k centers within X
  - Minimize **squared** distances from each point in C to its nearest center

Clustering Problems

Given discrete subsets C and F of metric space (X, d)

- k-Median problem:
  - Choose k centers in F
  - Minimize distances from each point in C to its nearest center
- k-Means problem:
  - Choose k centers within X
  - Minimize **squared** distances from each point in C to its nearest center
  - Can discretize the possible choices for centers with negligible approximation loss

Approximation Bounds Overview

k-Median:

Approximation Bounds Overview

k-Median:

- 2001: Jain and Vazirani 6-approx.

Approximation Bounds Overview

k-Median:

- 2001: Jain and Vazirani 6-approx.
- 2002: Jain et al. 4-approx.

Approximation Bounds Overview

k-Median:

- 2001: Jain and Vazirani 6-approx.
- 2002: Jain et al. 4-approx.
- 2003: **Archer et al.** exponential 3-approx.

Approximation Bounds Overview

k-Median:

- 2001: Jain and Vazirani 6-approx.
- 2002: Jain et al. 4-approx.
- 2003: **Archer et al.** exponential 3-approx.
- 2012: Li and Svensson 2.732-approx.

Approximation Bounds Overview

k-Median:

- 2001: Jain and Vazirani 6-approx.
- 2002: Jain et al. 4-approx.
- 2003: **Archer et al.** exponential 3-approx.
- 2012: Li and Svensson 2.732-approx.
- 2014: Byrka et al. 2.675-approx.

Approximation Bounds Overview

k-Median:

- 2001: Jain and Vazirani 6-approx.
- 2002: Jain et al. 4-approx.
- 2003: **Archer et al.** exponential 3-approx.
- 2012: Li and Svensson 2.732-approx.
- 2014: Byrka et al. 2.675-approx.
- 2017: **Ahmadian et al.** 2.633-approx. (Euclidean)

Approximation Bounds Overview

k-Median:

- 2001: Jain and Vazirani 6-approx.
- 2002: Jain et al. 4-approx.
- 2003: **Archer et al.** exponential 3-approx.
- 2012: Li and Svensson 2.732-approx.
- 2014: Byrka et al. 2.675-approx.
- 2017: **Ahmadian et al.** 2.633-approx. (Euclidean)

Hardness: 1.736

Approximation Bounds Overview

k-Means:

Approximation Bounds Overview

k-Means:

- 2001: Jain and Vazirani 54-approx.

Approximation Bounds Overview

k-Means:

- 2001: Jain and Vazirani 54-approx.
- 2003: **Archer et al.** exponential 9-approx.

Approximation Bounds Overview

k-Means:

- 2001: Jain and Vazirani 54-approx.
- 2003: **Archer et al.** exponential 9-approx.
- 2004: Kanungo et al. 9-approx. (Euclidean)

Approximation Bounds Overview

k-Means:

- 2001: Jain and Vazirani 54-approx.
- 2003: **Archer et al.** exponential 9-approx.
- 2004: Kanungo et al. 9-approx. (Euclidean)
- 2008: Gupta and Tangwongsan 16-approx.

Approximation Bounds Overview

k-Means:

- 2001: Jain and Vazirani 54-approx.
- 2003: **Archer et al.** exponential 9-approx.
- 2004: Kanungo et al. 9-approx. (Euclidean)
- 2008: Gupta and Tangwongsan 16-approx.
- 2017: **Ahmadian et al.** 9-approx.

Approximation Bounds Overview

k-Means:

- 2001: Jain and Vazirani 54-approx.
- 2003: **Archer et al.** exponential 9-approx.
- 2004: Kanungo et al. 9-approx. (Euclidean)
- 2008: Gupta and Tangwongsan 16-approx.
- 2017: **Ahmadian et al.** 9-approx.
- 2017: **Ahmadian et al.** 6.357-approx. (Euclidean)

Approximation Bounds Overview

k-Means:

- 2001: Jain and Vazirani 54-approx.
- 2003: **Archer et al.** exponential 9-approx.
- 2004: Kanungo et al. 9-approx. (Euclidean)
- 2008: Gupta and Tangwongsan 16-approx.
- 2017: **Ahmadian et al.** 9-approx.
- 2017: **Ahmadian et al.** 6.357-approx. (Euclidean)

Hardness: 3.94 (General), 1.0013 (Euclidean)

Talk Outline

Facility Location Problem

Talk Outline

Facility Location Problem

- Jain and Vazirani (JV) Algorithm

Talk Outline

Facility Location Problem

- Jain and Vazirani (JV) Algorithm
- Relation to k-Means and k-Median

Talk Outline

Facility Location Problem

- Jain and Vazirani (JV) Algorithm
- Relation to k-Means and k-Median

Continuous Adaptations of the JV Algorithm

Talk Outline

Facility Location Problem

- Jain and Vazirani (JV) Algorithm
- Relation to k-Means and k-Median

Continuous Adaptations of the JV Algorithm

- Archer et al. Exponential Algorithm

Talk Outline

Facility Location Problem

- Jain and Vazirani (JV) Algorithm
- Relation to k-Means and k-Median

Continuous Adaptations of the JV Algorithm

- Archer et al. Exponential Algorithm
- Ahmadian et al. Quasipolynomial Algorithm

Talk Outline

Facility Location Problem

- Jain and Vazirani (JV) Algorithm
- Relation to k-Means and k-Median

Continuous Adaptations of the JV Algorithm

- Archer et al. Exponential Algorithm
- Ahmadian et al. Quasipolynomial Algorithm
- Ahmadian et al. Polynomial Algorithm

Uncapacitated Facility Location

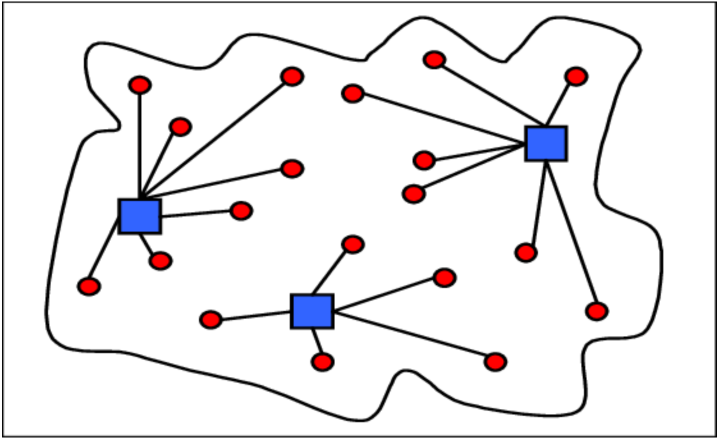

Figure: researchgate.net/figure/Facility-location-problem-example_fig1_221182599

Uncapacitated Facility Location

Given a set of *facilities* F, and a set of *clients* C

Uncapacitated Facility Location

Given a set of *facilities* F, and a set of *clients* C

- Each facility i has an *opening cost*

Uncapacitated Facility Location

Given a set of *facilities* F, and a set of *clients* C

- Each facility i has an *opening cost*
- Want to minimize facility opening costs plus distances of clients to their nearest facility

Uncapacitated Facility Location

Primal LP:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i\in F, j\in C} c_{ij}x_{ij} && + \sum_{i\in F} f_i y_i \\
\text{subject to} \quad & \sum_{i\in F} x_{ij} \geq 1, && \forall j \in C, \\
& y_i - x_{ij} \geq 0, && \forall i \in F, j \in C, \\
& x_{ij} \geq 0, && \forall i \in F, j \in C, \\
& y_i \geq 0, && \forall i \in F.
\end{aligned}
$$

c_{ij} = distance, f_i = facility cost,
x_{ij} = client connection, y_i = facility open

Uncapacitated Facility Location

Dual LP:

$$
\begin{array}{lll}
\text{maximize} & \displaystyle\sum_{j \in C} \alpha_j & \\
\text{subject to} & \alpha_j - \beta_{ij} \le c_{ij}, & \forall i \in F, j \in C, \\
& \displaystyle\sum_{j \in C} \beta_{ij} \le f_i, & \forall i \in F, \\
& \alpha_j \ge 0, & \forall j \in C, \\
& \beta_{ij} \ge 0, & \forall i \in F, j \in C.
\end{array}
$$

c_{ij} = distance, f_i = facility cost,
β_{ij} = client contribution, α_j = client value

JV Primal-Dual Algorithm (Stage 1)

Stage 1

JV Primal-Dual Algorithm (Stage 1)

Stage 1

- Each client is initially *unconnected*

JV Primal-Dual Algorithm (Stage 1)

Stage 1

- Each client is initially *unconnected*
- Each facility is not *tight* or *temporarily open*

JV Primal-Dual Algorithm (Stage 1)

Stage 1

- Each client is initially *unconnected*
- Each facility is not *tight* or *temporarily open*
- Event *time* starts at $t = 0$

JV Primal-Dual Algorithm (Stage 1)

Stage 1

- Each client is initially *unconnected*
- Each facility is not *tight* or *temporarily open*
- Event *time* starts at $t = 0$
- As t increases, each α_j also increases at the same rate until an event occurs

JV Primal-Dual Algorithm (Stage 1)

Stage 1

- Each client is initially *unconnected*
- Each facility is not *tight* or *temporarily open*
- Event *time* starts at $t = 0$
- As t increases, each α_j also increases at the same rate until an event occurs
- Stage 1 ends when no unconnected clients remain

JV Primal-Dual Algorithm (Stage 1)

An *edge goes tight* when some $\alpha_j = c_{ij}$

JV Primal-Dual Algorithm (Stage 1)

An *edge goes tight* when some $\alpha_j = c_{ij}$

- If facility i is not temporarily open, then start increasing β_{ij}

JV Primal-Dual Algorithm (Stage 1)

An *edge goes tight* when some $\alpha_j = c_{ij}$

- If facility i is not temporarily open, then start increasing β_{ij}
  - Client j is now *contributing* to facility i

JV Primal-Dual Algorithm (Stage 1)

An *edge goes tight* when some $\alpha_j = c_{ij}$

- If facility i is not temporarily open, then start increasing β_{ij}
  - Client j is now *contributing* to facility i
- If facility i is temporarily open, then declare client j to be connected

JV Primal-Dual Algorithm (Stage 1)

An *edge goes tight* when some $\alpha_j = c_{ij}$

- If facility i is not temporarily open, then start increasing β_{ij}
  - Client j is now *contributing* to facility i
- If facility i is temporarily open, then declare client j to be connected
  - Stop increasing α_j and each β_{hj} for all facilities $h \in F$

JV Primal-Dual Algorithm (Stage 1)

An *edge goes tight* when some $\alpha_j = c_{ij}$

- If facility i is not temporarily open, then start increasing β_{ij}
  - Client j is now *contributing* to facility i
- If facility i is temporarily open, then declare client j to be connected
  - Stop increasing α_j and each β_{hj} for all facilities $h \in F$
  - Facility i is the *connecting witness* for client j

JV Primal-Dual Algorithm (Stage 1)

Facility i is *paid for* when $\sum_{j \in C} \beta_{ij} = f_i$

JV Primal-Dual Algorithm (Stage 1)

Facility i is *paid for* when $\sum_{j \in C} \beta_{ij} = f_i$

- Declare facility i to be temporarily open

JV Primal-Dual Algorithm (Stage 1)

Facility i is *paid for* when $\sum_{j \in C} \beta_{ij} = f_i$

- Declare facility i to be temporarily open
- Each unconnected client j that was contributing to facility i is now declared to be connected

JV Primal-Dual Algorithm (Stage 1)

Facility i is *paid for* when $\sum_{j \in C} \beta_{ij} = f_i$

- Declare facility i to be temporarily open
- Each unconnected client j that was contributing to facility i is now declared to be connected
- Facility i is the connecting witness for these clients

JV Primal-Dual Algorithm (Stage 1)

Facility i is *paid for* when $\sum_{j \in C} \beta_{ij} = f_i$

- Declare facility i to be temporarily open
- Each unconnected client j that was contributing to facility i is now declared to be connected
- Facility i is the connecting witness for these clients
- The dual variables for each of these clients now stops increasing

JV Primal-Dual Algorithm (Stage 2)

Stage 2

JV Primal-Dual Algorithm (Stage 2)

Stage 2

- Construct a graph G with vertices given by the temporarily opened facilities from Stage 1

JV Primal-Dual Algorithm (Stage 2)

Stage 2

- Construct a graph G with vertices given by the temporarily opened facilities from Stage 1
- Allow an edge between facilities $i \neq i'$ if some client j made positive contributions to both

JV Primal-Dual Algorithm (Stage 2)

Stage 2

- Construct a graph G with vertices given by the temporarily opened facilities from Stage 1
- Allow an edge between facilities $i \neq i'$ if some client j made positive contributions to both
- Return any maximal independent set of G

Relating UFL to k-Median and k-Means

JV Algorithm Approximation Bound:

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

Relating UFL to k-Median and k-Means

JV Algorithm Approximation Bound:

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

The JV algorithm also satisfies a *Lagrange-multiplier preserving* (LMP) property:

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

Relating UFL to k-Median and k-Means

Setting each facility cost $f_i = \lambda \geq 0$ yields

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

Relating UFL to k-Median and k-Means

Setting each facility cost $f_i = \lambda \geq 0$ yields

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

This corresponds to the primal and dual objectives of k-median when the number of opened facilities equals k

Relating UFL to k-Median and k-Means

Setting each facility cost $f_i = \lambda \geq 0$ yields

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

This corresponds to the primal and dual objectives of k-median when the number of opened facilities equals k

- Bound for (discrete) k-means is 9

JV Algorithm Continuity

Given a UFL instance with uniform facility cost $\lambda \geq 0$

JV Algorithm Continuity

Given a UFL instance with uniform facility cost $\lambda \geq 0$

- When $\lambda = 0$, all facilities open

JV Algorithm Continuity

Given a UFL instance with uniform facility cost $\lambda \geq 0$

- When $\lambda = 0$, all facilities open
- When λ is large enough, only one facility opens

JV Algorithm Continuity

Given a UFL instance with uniform facility cost $\lambda \geq 0$

- When $\lambda = 0$, all facilities open
- When λ is large enough, only one facility opens
- The JV algorithm is *continuous* if, as λ increases, the total number of opened facilities never jumps by more than 1 at a time

JV Algorithm Continuity

A bad example:

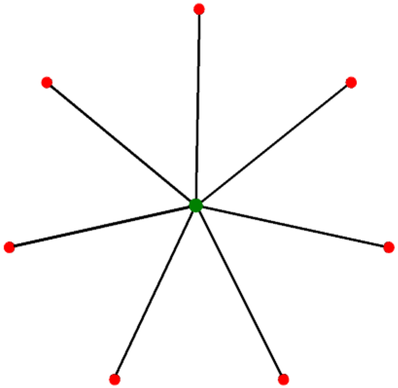

Figure: https://en.wikipedia.org/wiki/Star_(graph_theory)

JV Algorithm Continuity

Fixing bad examples:

JV Algorithm Continuity

Fixing bad examples:

- Perturb the distances so that no $c_{ij} = c_{i'j'}$

JV Algorithm Continuity

Fixing bad examples:

- Perturb the distances so that no $c_{ij} = c_{i'j'}$
- Always choose a *maximum* independent set of facilities in Stage 2

JV Algorithm Continuity

Fixing bad examples:

- Perturb the distances so that no $c_{ij} = c_{i'j'}$
- Always choose a *maximum* independent set of facilities in Stage 2

Theorem (Archer et al.): as λ increases, the number of opened facilities changes by at most 1 at a time

JV Algorithm Continuity

Fixing bad examples:

- Perturb the distances so that no $c_{ij} = c_{i'j'}$
- Always choose a *maximum* independent set of facilities in Stage 2

Theorem (Archer et al.): as λ increases, the number of opened facilities changes by at most 1 at a time

- Exponential time algorithm

A New Approach

Instead of choosing maximum IS, we will just allow for "larger" maximal IS

A New Approach

Instead of choosing maximum IS, we will just allow for "larger" maximal IS

- Let t_i be the time that facility i opens in Stage 1

A New Approach

Instead of choosing maximum IS, we will just allow for "larger" maximal IS

- Let t_i be the time that facility i opens in Stage 1
- Stage 2: edge between facilities i, i' if some client has positive contributions to both **and if** $c_{ii'} \leq \delta \min\{t_i, t_{i'}\}$

A New Approach

Instead of choosing maximum IS, we will just allow for "larger" maximal IS

- Let t_i be the time that facility i opens in Stage 1
- Stage 2: edge between facilities i, i' if some client has positive contributions to both **and if** $c_{ii'} \leq \delta \min\{t_i, t_{i'}\}$

Note that $\delta = \infty$ yields the original JV algorithm

LMP Properties of JV(δ)

The JV(δ) algorithm satisfies the LMP property with constant

LMP Properties of JV(δ)

The JV(δ) algorithm satisfies the LMP property with constant

- 9 for k-means in general metrics ($\delta = \infty$)

LMP Properties of JV(δ)

The JV(δ) algorithm satisfies the LMP property with constant

- 9 for k-means in general metrics ($\delta = \infty$)
- 6.3574 for k-means in the Euclidean metric ($\delta = 2.3146$)

LMP Properties of JV(δ)

The JV(δ) algorithm satisfies the LMP property with constant

- 9 for k-means in general metrics ($\delta = \infty$)
- 6.3574 for k-means in the Euclidean metric ($\delta = 2.3146$)
- 3 for k-median in general metrics ($\delta = \infty$)

LMP Properties of JV(δ)

The JV(δ) algorithm satisfies the LMP property with constant

- 9 for k-means in general metrics ($\delta = \infty$)
- 6.3574 for k-means in the Euclidean metric ($\delta = 2.3146$)
- 3 for k-median in general metrics ($\delta = \infty$)
- 2.633 for k-median in the Euclidean metric ($\delta = 1.633$)

LMP Analysis

Consider JV(∞) for k-median:

LMP Analysis

Consider JV(∞) for k-median:

- Given a maximal independent set IS of facilities

LMP Analysis

Consider JV(∞) for k-median:

- Given a maximal independent set IS of facilities
- Let i be the witness facility for some client j

LMP Analysis

Consider JV(∞) for k-median:

- Given a maximal independent set IS of facilities
- Let i be the witness facility for some client j
- If $i \in$ IS, the distance c_{ij} is bounded by α_j

LMP Analysis

Consider JV(∞) for k-median:

- Given a maximal independent set IS of facilities
- Let i be the witness facility for some client j
- If $i \in$ IS, the distance c_{ij} is bounded by α_j
- If $i \notin$ IS, then some client j' contributed to both i and some $i' \in$ IS: $c_{i'j} \leq c_{ij} + c_{ij'} + c_{i'j'} \leq 3\alpha_j$

LMP Analysis

Consider JV(∞) for k-median:

- Given a maximal independent set IS of facilities
- Let i be the witness facility for some client j
- If $i \in$ IS, the distance c_{ij} is bounded by α_j
- If $i \notin$ IS, then some client j' contributed to both i and some $i' \in$ IS: $c_{i'j} \leq c_{ij} + c_{ij'} + c_{i'j'} \leq 3\alpha_j$
- Leads to overall 3-approximation result

LMP Analysis

Consider JV(δ) for k-median (Euclidean metric):

LMP Analysis

Consider JV(δ) for k-median (Euclidean metric):

- Let ρ be our approximation bound

LMP Analysis

Consider JV(δ) for k-median (Euclidean metric):

- Let ρ be our approximation bound
- Now j may have contributed to $s > 1$ facilities in IS

LMP Analysis

Consider JV(δ) for k-median (Euclidean metric):

- Let ρ be our approximation bound
- Now j may have contributed to $s > 1$ facilities in IS
- A centroid inequality yields $s < \delta^2/(\delta^2 - 2)$

LMP Analysis

Consider JV(δ) for k-median (Euclidean metric):

- Let ρ be our approximation bound
- Now j may have contributed to $s > 1$ facilities in IS
- A centroid inequality yields $s < \delta^2/(\delta^2 - 2)$
- If $s = 0$, then $\rho \geq 1 + \delta$

LMP Analysis

Consider JV(δ) for k-median (Euclidean metric):

- Let ρ be our approximation bound
- Now j may have contributed to $s > 1$ facilities in IS
- A centroid inequality yields $s < \delta^2/(\delta^2 - 2)$
- If $s = 0$, then $\rho \geq 1 + \delta$
- If $s > 1$, then $\rho \geq 1/\left(\frac{1}{s-1}\binom{s}{2}\delta - (s-1)\right)$

LMP Analysis

Consider JV(δ) for k-median (Euclidean metric):

- Let ρ be our approximation bound
- Now j may have contributed to $s > 1$ facilities in IS
- A centroid inequality yields $s < \delta^2/(\delta^2 - 2)$
- If $s = 0$, then $\rho \geq 1 + \delta$
- If $s > 1$, then $\rho \geq 1/\left(\frac{1}{s-1}\binom{s}{2}\delta - (s-1)\right)$

Best result: $\delta = \sqrt{8/3}$ yields $s < 4$ and $\rho \approx 2.633$

Quasipolynomial Algorithm

Goal: find "good, close" dual solutions $\alpha^{(0)}, \ldots, \alpha^{(L)}$

Quasipolynomial Algorithm

Goal: find "good, close" dual solutions $\alpha^{(0)}, \ldots, \alpha^{(L)}$

- We also control the change in number of open facilities between solutions

Quasipolynomial Algorithm

Goal: find "good, close" dual solutions $\alpha^{(0)}, \dots, \alpha^{(L)}$

- We also control the change in number of open facilities between solutions

Parameter values $\lambda = 0, \epsilon_z, 2\epsilon_z, \dots, L\epsilon_z$

Quasipolynomial Algorithm

Goal: find "good, close" dual solutions $\alpha^{(0)}, \ldots, \alpha^{(L)}$

- We also control the change in number of open facilities between solutions

Parameter values $\lambda = 0, \epsilon_z, 2\epsilon_z, \ldots, L\epsilon_z$

- Let ϵ be an approximation error factor, let $n = |C|$

Quasipolynomial Algorithm

Goal: find "good, close" dual solutions $\alpha^{(0)}, \ldots, \alpha^{(L)}$

- We also control the change in number of open facilities between solutions

Parameter values $\lambda = 0, \epsilon_z, 2\epsilon_z, \ldots, L\epsilon_z$

- Let ϵ be an approximation error factor, let $n = |C|$
- Fix $\epsilon_z = n^{-3-30\log_{1+\epsilon} n}$, $L = 4n^7 \cdot \epsilon_z^{-1} = n^{O(\epsilon^{-1}\log n)}$

Quasipolynomial Algorithm

Goal: find "good, close" dual solutions $\alpha^{(0)}, \ldots, \alpha^{(L)}$

- We also control the change in number of open facilities between solutions

Parameter values $\lambda = 0, \epsilon_z, 2\epsilon_z, \ldots, L\epsilon_z$

- Let ϵ be an approximation error factor, let $n = |C|$
- Fix $\epsilon_z = n^{-3-30\log_{1+\epsilon} n}$, $L = 4n^7 \cdot \epsilon_z^{-1} = n^{O(\epsilon^{-1}\log n)}$
- List is quasipolynomial in length

Quasipolynomial Algorithm

Goal: find "good, close" dual solutions $\alpha^{(0)}, \dots, \alpha^{(L)}$

- We also control the change in number of open facilities between solutions

Parameter values $\lambda = 0, \epsilon_z, 2\epsilon_z, \dots, L\epsilon_z$

- Let ϵ be an approximation error factor, let $n = |C|$
- Fix $\epsilon_z = n^{-3-30\log_{1+\epsilon} n}$, $L = 4n^7 \cdot \epsilon_z^{-1} = n^{O(\epsilon^{-1}\log n)}$
- List is quasipolynomial in length

Two parts: $\textsc{QuasiSweep}$, $\textsc{QuasiGraphUpdate}$

QuasiSweep

For $x \in \mathbb{R}$, define

$$B(x) = \begin{cases} 0, & \text{if } x < 1, \\ 1 + \lfloor \log_{1+\epsilon}(x) \rfloor, & \text{if } x \geq 1 \end{cases}$$

QuasiSweep

For $x \in \mathbb{R}$, define

$$B(x) = \begin{cases} 0, & \text{if } x < 1, \\ 1 + \lfloor \log_{1+\epsilon}(x) \rfloor, & \text{if } x \geq 1 \end{cases}$$

- $B(x)$ is the index of the *bucket* that contains x

QuasiSweep

For $x \in \mathbb{R}$, define

$$B(x) = \begin{cases} 0, & \text{if } x < 1, \\ 1 + \lfloor \log_{1+\epsilon}(x) \rfloor, & \text{if } x \geq 1 \end{cases}$$

- $B(x)$ is the index of the *bucket* that contains x
- The α-values for any two clients in the same bucket differ by at most $1 + \epsilon$

QuasiSweep

Begin with good dual solution $\alpha^{(l)}$ for parameter λ

QuasiSweep

Begin with good dual solution $\alpha^{(l)}$ for parameter λ

- Raise the facility price to $\lambda + \epsilon_z$

QuasiSweep

Begin with good dual solution $\alpha^{(l)}$ for parameter λ

- Raise the facility price to $\lambda + \epsilon_z$

Let $A = \emptyset$ and $\theta = 0$. Increase θ continuously:

QuasiSweep

Begin with good dual solution $\alpha^{(l)}$ for parameter λ

- Raise the facility price to $\lambda + \epsilon_z$

Let $A = \emptyset$ and $\theta = 0$. Increase θ continuously:

- Whenever $\theta = \alpha_j$, add j to A

QuasiSweep

Begin with good dual solution $\alpha^{(l)}$ for parameter λ

- Raise the facility price to $\lambda + \epsilon_z$

Let $A = \emptyset$ and $\theta = 0$. Increase θ continuously:

- Whenever $\theta = \alpha_j$, add j to A
- Remove j from A whenever j has a tight edge to a tight facility i where $B(\alpha_j) \geq B(t_i)$

QuasiSweep

Begin with good dual solution $\alpha^{(l)}$ for parameter λ

- Raise the facility price to $\lambda + \epsilon_z$

Let $A = \emptyset$ and $\theta = 0$. Increase θ continuously:

- Whenever $\theta = \alpha_j$, add j to A
- Remove j from A whenever j has a tight edge to a tight facility i where $B(\alpha_j) \geq B(t_i)$
- Decrease each α_j with $B(\alpha_j) > B(\theta)$ at a rate of $|A|$ times the rate that θ is increasing

QuasiSweep

Begin with good dual solution $\alpha^{(l)}$ for parameter λ

- Raise the facility price to $\lambda + \epsilon_z$

Let $A = \emptyset$ and $\theta = 0$. Increase θ continuously:

- Whenever $\theta = \alpha_j$, add j to A
- Remove j from A whenever j has a tight edge to a tight facility i where $B(\alpha_j) \geq B(t_i)$
- Decrease each α_j with $B(\alpha_j) > B(\theta)$ at a rate of $|A|$ times the rate that θ is increasing

Stop when every client is added and removed from A

QuasiSweep

Begin with good dual solution $\alpha^{(l)}$ for parameter λ

- Raise the facility price to $\lambda + \epsilon_z$

Let $A = \emptyset$ and $\theta = 0$. Increase θ continuously:

- Whenever $\theta = \alpha_j$, add j to A
- Remove j from A whenever j has a tight edge to a tight facility i where $B(\alpha_j) \geq B(t_i)$
- Decrease each α_j with $B(\alpha_j) > B(\theta)$ at a rate of $|A|$ times the rate that θ is increasing

Stop when every client is added and removed from A

Output dual solution $\alpha^{(l+1)}$ for parameter $\lambda + \epsilon_z$

QUASIGRAPHUPDATE

Each $\alpha^{(l)}$ has a graph $G^{(l)}$ of facilities

QuasiGraphUpdate

Each $\alpha^{(l)}$ has a graph $G^{(l)}$ of facilities

- Generate (polynomially many) intermediate graphs $G^{(l)} = G^{(l,0)}, G^{(l,1)}, \ldots, G^{(l,p_l)} = G^{(l+1)}$

QuasiGraphUpdate

Each $\alpha^{(l)}$ has a graph $G^{(l)}$ of facilities

- Generate (polynomially many) intermediate graphs $G^{(l)} = G^{(l,0)}, G^{(l,1)}, \ldots, G^{(l,p_l)} = G^{(l+1)}$
- Obtain maximal independent sets $\mathsf{IS}^{(l)} = \mathsf{IS}^{(l,0)}, \mathsf{IS}^{(l,1)}, \ldots, \mathsf{IS}^{(l,p_l)} = \mathsf{IS}^{(l+1)}$

QuasiGraphUpdate

Each $\alpha^{(l)}$ has a graph $G^{(l)}$ of facilities

- Generate (polynomially many) intermediate graphs $G^{(l)} = G^{(l,0)}, G^{(l,1)}, \ldots, G^{(l,p_l)} = G^{(l+1)}$
- Obtain maximal independent sets $\mathsf{IS}^{(l)} = \mathsf{IS}^{(l,0)}, \mathsf{IS}^{(l,1)}, \ldots, \mathsf{IS}^{(l,p_l)} = \mathsf{IS}^{(l+1)}$
- Size decreases by at most one after each step

QuasiGraphUpdate

Each $\alpha^{(l)}$ has a graph $G^{(l)}$ of facilities

- Generate (polynomially many) intermediate graphs $G^{(l)} = G^{(l,0)}, G^{(l,1)}, \ldots, G^{(l,p_l)} = G^{(l+1)}$
- Obtain maximal independent sets $\mathsf{IS}^{(l)} = \mathsf{IS}^{(l,0)}, \mathsf{IS}^{(l,1)}, \ldots, \mathsf{IS}^{(l,p_l)} = \mathsf{IS}^{(l+1)}$
- Size decreases by at most one after each step

Return first independent set we find with size k

QuasiGraphUpdate

Input: $G^{(l)}, G^{(l+1)}$, and $\mathsf{IS}^{(l)}$ (of size greater than k)

QuasiGraphUpdate

Input: $G^{(l)}, G^{(l+1)}$, and $\mathsf{IS}^{(l)}$ (of size greater than k)

- Copy $G^{(l)}$, $G^{(l+1)}$ into disjoint sets $V^{(l)}$, $V^{(l+1)}$

QuasiGraphUpdate

Input: $G^{(l)}, G^{(l+1)}$, and $\mathsf{IS}^{(l)}$ (of size greater than k)

- Copy $G^{(l)}$, $G^{(l+1)}$ into disjoint sets $V^{(l)}$, $V^{(l+1)}$
- Bipartite graph G' over $V^{(l)} \cup V^{(l+1)}$ and C

QuasiGraphUpdate

Input: $G^{(l)}, G^{(l+1)}$, and $\mathsf{IS}^{(l)}$ (of size greater than k)

- Copy $G^{(l)}$, $G^{(l+1)}$ into disjoint sets $V^{(l)}$, $V^{(l+1)}$
- Bipartite graph G' over $V^{(l)} \cup V^{(l+1)}$ and C
- Edge j to $i \in V^{(l)}$ if j contributes to i in $\alpha^{(l)}$, etc.

QuasiGraphUpdate

Input: $G^{(l)}, G^{(l+1)}$, and $\mathsf{IS}^{(l)}$ (of size greater than k)

- Copy $G^{(l)}$, $G^{(l+1)}$ into disjoint sets $V^{(l)}$, $V^{(l+1)}$
- Bipartite graph G' over $V^{(l)} \cup V^{(l+1)}$ and C
- Edge j to $i \in V^{(l)}$ if j contributes to i in $\alpha^{(l)}$, etc.
- Generate $G^{(l,1)}$ on G', where the induced subgraph of $G^{(l,1)}$ on $V^{(l)}$ equals $G^{(l)} = G^{(l,0)}$

QuasiGraphUpdate

Input: $G^{(l)}, G^{(l+1)}$, and $\mathsf{IS}^{(l)}$ (of size greater than k)

- Copy $G^{(l)}$, $G^{(l+1)}$ into disjoint sets $V^{(l)}$, $V^{(l+1)}$
- Bipartite graph G' over $V^{(l)} \cup V^{(l+1)}$ and C
- Edge j to $i \in V^{(l)}$ if j contributes to i in $\alpha^{(l)}$, etc.
- Generate $G^{(l,1)}$ on G', where the induced subgraph of $G^{(l,1)}$ on $V^{(l)}$ equals $G^{(l)} = G^{(l,0)}$
- Greedily extend $\mathsf{IS}^{(l)} = \mathsf{IS}^{(l,0)}$ to a maximal independent set $\mathsf{IS}^{(l,1)}$ for $G^{(l,1)}$

QuasiGraphUpdate

Input: $G^{(l)}, G^{(l+1)}$, and $\mathsf{IS}^{(l)}$ (of size greater than k)

- Copy $G^{(l)}$, $G^{(l+1)}$ into disjoint sets $V^{(l)}$, $V^{(l+1)}$
- Bipartite graph G' over $V^{(l)} \cup V^{(l+1)}$ and C
- Edge j to $i \in V^{(l)}$ if j contributes to i in $\alpha^{(l)}$, etc.
- Generate $G^{(l,1)}$ on G', where the induced subgraph of $G^{(l,1)}$ on $V^{(l)}$ equals $G^{(l)} = G^{(l,0)}$
- Greedily extend $\mathsf{IS}^{(l)} = \mathsf{IS}^{(l,0)}$ to a maximal independent set $\mathsf{IS}^{(l,1)}$ for $G^{(l,1)}$
- Continue generating graphs and IS's by removing facilities in $V^{(l)}$ from G' one by one

QuasiGraphUpdate

Input: $G^{(l)}, G^{(l+1)}$, and $\mathsf{IS}^{(l)}$ (of size greater than k)

- Copy $G^{(l)}$, $G^{(l+1)}$ into disjoint sets $V^{(l)}$, $V^{(l+1)}$
- Bipartite graph G' over $V^{(l)} \cup V^{(l+1)}$ and C
- Edge j to $i \in V^{(l)}$ if j contributes to i in $\alpha^{(l)}$, etc.
- Generate $G^{(l,1)}$ on G', where the induced subgraph of $G^{(l,1)}$ on $V^{(l)}$ equals $G^{(l)} = G^{(l,0)}$
- Greedily extend $\mathsf{IS}^{(l)} = \mathsf{IS}^{(l,0)}$ to a maximal independent set $\mathsf{IS}^{(l,1)}$ for $G^{(l,1)}$
- Continue generating graphs and IS's by removing facilities in $V^{(l)}$ from G' one by one
- After $p_l = |V^{(l)}|$ steps, arrive at $G^{(l,p_l)} = G^{(l+1)}$

Quasipolynomial Algorithm: in Review

$\textsc{QuasiSweep}$ generates a quasipolynomial length list of dual solutions $\alpha^{(0)}, \ldots, \alpha^{(L)}$

Quasipolynomial Algorithm: in Review

QuasiSweep generates a quasipolynomial length list of dual solutions $\alpha^{(0)}, \dots, \alpha^{(L)}$

QuasiGraphUpdate interpolates between every two solutions $\alpha^{(l)}$, $\alpha^{(l+1)}$ to make sure we eventually find a set of open facilities of size k

Quasipolynomial Algorithm: in Review

$\textsc{QuasiSweep}$ generates a quasipolynomial length list of dual solutions $\alpha^{(0)}, \ldots, \alpha^{(L)}$

$\textsc{QuasiGraphUpdate}$ interpolates between every two solutions $\alpha^{(l)}$, $\alpha^{(l+1)}$ to make sure we eventually find a set of open facilities of size k

Analysis: $\displaystyle\sum_{i \in F, j \in C} c_{ij} x_{ij} \leq (\rho + O(\epsilon)) \cdot \mathsf{OPT}_k$

Polynomial Algorithm: an Overview

Good dual solutions $\rightarrow$ λ-*roundable* dual solutions

Polynomial Algorithm: an Overview

Good dual solutions $\rightarrow$ λ-*roundable* dual solutions

- Facility costs $\lambda = 0, 1 \cdot \epsilon_z, \ldots, L \cdot \epsilon_z$ where $L = 4n^7 \cdot \epsilon_z^{-1}$ and $\epsilon_z = n^{-O(1)}$

Polynomial Algorithm: an Overview

Good dual solutions $\rightarrow$ λ-*roundable* dual solutions

- Facility costs $\lambda = 0, 1 \cdot \epsilon_z, \ldots, L \cdot \epsilon_z$ where $L = 4n^7 \cdot \epsilon_z^{-1}$ and $\epsilon_z = n^{-O(1)}$

$\textsc{RaisePrice}$ increases facility costs *one by one*

Polynomial Algorithm: an Overview

Good dual solutions $\rightarrow$ λ-*roundable* dual solutions

- Facility costs $\lambda = 0, 1 \cdot \epsilon_z, \ldots, L \cdot \epsilon_z$ where $L = 4n^7 \cdot \epsilon_z^{-1}$ and $\epsilon_z = n^{-O(1)}$

$\textsc{RaisePrice}$ increases facility costs *one by one*

- Given facility i raised from λ to $\lambda + \epsilon_z$

Polynomial Algorithm: an Overview

Good dual solutions $\rightarrow$ λ-*roundable* dual solutions

- Facility costs $\lambda = 0, 1 \cdot \epsilon_z, \ldots, L \cdot \epsilon_z$ where $L = 4n^7 \cdot \epsilon_z^{-1}$ and $\epsilon_z = n^{-O(1)}$

$\textsc{RaisePrice}$ increases facility costs *one by one*

- Given facility i raised from λ to $\lambda + \epsilon_z$
- Obtain close sequence of roundable solutions $S^{(1)}, \ldots, S^{(q)}$, using a $\textsc{Sweep}$ subroutine

Polynomial Algorithm: an Overview

Good dual solutions $\rightarrow$ λ-*roundable* dual solutions

- Facility costs $\lambda = 0, 1 \cdot \epsilon_z, \ldots, L \cdot \epsilon_z$ where $L = 4n^7 \cdot \epsilon_z^{-1}$ and $\epsilon_z = n^{-O(1)}$

$\textsc{RaisePrice}$ increases facility costs *one by one*

- Given facility i raised from λ to $\lambda + \epsilon_z$
- Obtain close sequence of roundable solutions $S^{(1)}, \ldots, S^{(q)}$, using a $\textsc{Sweep}$ subroutine

$\textsc{GraphUpdate}$ interpolates between each $S^{(l)}$, $S^{(l+1)}$

Polynomial Algorithm: an Overview

Good dual solutions $\rightarrow$ λ-*roundable* dual solutions

- Facility costs $\lambda = 0, 1 \cdot \epsilon_z, \ldots, L \cdot \epsilon_z$ where $L = 4n^7 \cdot \epsilon_z^{-1}$ and $\epsilon_z = n^{-O(1)}$

$\textsc{RaisePrice}$ increases facility costs *one by one*

- Given facility i raised from λ to $\lambda + \epsilon_z$
- Obtain close sequence of roundable solutions $S^{(1)}, \ldots, S^{(q)}$, using a $\textsc{Sweep}$ subroutine

$\textsc{GraphUpdate}$ interpolates between each $S^{(l)}$, $S^{(l+1)}$

- Similar to $\textsc{QuasiGraphUpdate}$

Polynomial Algorithm: an Overview

Initialize the current integral solution $\mathsf{IS}^{(0)} = F$

Polynomial Algorithm: an Overview

Initialize the current integral solution $\mathsf{IS}^{(0)} = F$
Loop over $\lambda = 0, \epsilon_z, 2\epsilon_z, \ldots, L\epsilon_z$:

Polynomial Algorithm: an Overview

Initialize the current integral solution $\mathsf{IS}^{(0)} = F$

Loop over $\lambda = 0, \epsilon_z, 2\epsilon_z, \ldots, L\epsilon_z$:

- While some facility i still has cost λ:

Polynomial Algorithm: an Overview

Initialize the current integral solution $\mathsf{IS}^{(0)} = F$

Loop over $\lambda = 0, \epsilon_z, 2\epsilon_z, \ldots, L\epsilon_z$:

- While some facility i still has cost λ:
  - Call $\textsc{RaisePrice}$ on i and produce a sequence $S^{(1)}, \ldots, S^{(q)}$ of λ-roundable solutions

Polynomial Algorithm: an Overview

Initialize the current integral solution $\text{IS}^{(0)} = F$

Loop over $\lambda = 0, \epsilon_z, 2\epsilon_z, \ldots, L\epsilon_z$:

- While some facility i still has cost λ:
  - Call $\textsc{RaisePrice}$ on i and produce a sequence $S^{(1)}, \ldots, S^{(q)}$ of λ-roundable solutions
  - For $l = 0$ to $q - 1$:

Polynomial Algorithm: an Overview

Initialize the current integral solution $\mathsf{IS}^{(0)} = F$

Loop over $\lambda = 0, \epsilon_z, 2\epsilon_z, \ldots, L\epsilon_z$:

- While some facility i still has cost λ:
  - Call $\textsc{RaisePrice}$ on i and produce a sequence $S^{(1)}, \ldots, S^{(q)}$ of λ-roundable solutions
  - For $l = 0$ to $q - 1$:
    - Call $\textsc{GraphUpdate}$ on $S^{(l)}$, $S^{(l+1)}$ to produce a sequence $\mathsf{IS}^{(l,0)}, \ldots, \mathsf{IS}^{(l,p_l)}$

Polynomial Algorithm: an Overview

Initialize the current integral solution $\mathsf{IS}^{(0)} = F$

Loop over $\lambda = 0, \epsilon_z, 2\epsilon_z, \ldots, L\epsilon_z$:

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

Polynomial Algorithm: an Overview

Initialize the current integral solution $\mathsf{IS}^{(0)} = F$

Loop over $\lambda = 0, \epsilon_z, 2\epsilon_z, \ldots, L\epsilon_z$:

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

Polynomial Algorithm: an Overview

Initialize the current integral solution $\mathsf{IS}^{(0)} = F$

Loop over $\lambda = 0, \epsilon_z, 2\epsilon_z, \ldots, L\epsilon_z$:

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

Polynomial Algorithm: an Overview

Analysis:

Polynomial Algorithm: an Overview

Analysis:

- Each step of the algorithm runs in polynomial time

Polynomial Algorithm: an Overview

Analysis:

- Each step of the algorithm runs in polynomial time
- Still not efficient: outer loop is $O(n^8)$

Polynomial Algorithm: an Overview

Analysis:

- Each step of the algorithm runs in polynomial time
- Still not efficient: outer loop is $O(n^8)$
    - Small values of k require more iterations

Polynomial Algorithm: an Overview

Analysis:

- Each step of the algorithm runs in polynomial time
- Still not efficient: outer loop is $O(n^8)$
  - Small values of k require more iterations
- Returned solution is a $(\rho + O(\epsilon))$-approximation

In Summary

We discussed the JV facility location algorithm

In Summary

We discussed the JV facility location algorithm

- Good approximations to k-means and k-median when it opens k facilities

In Summary

We discussed the JV facility location algorithm

- Good approximations to k-means and k-median when it opens k facilities

We overviewed three modifications to make the JV algorithm *continuous*

In Summary

We discussed the JV facility location algorithm

- Good approximations to k-means and k-median when it opens k facilities

We overviewed three modifications to make the JV algorithm *continuous*

- Guaranteed to find k facilities for any value of k

In Summary

We discussed the JV facility location algorithm

- Good approximations to k-means and k-median when it opens k facilities

We overviewed three modifications to make the JV algorithm *continuous*

- Guaranteed to find k facilities for any value of k

Can we do the same for other LMP algorithms?