Fair k-centers via Maximum Matching

by Huy Nguyen, Matthew Jones, Thy Nguyen

June 15, 2020

Content

- Introduction
- The fair k-centers problem
- Approach using maximum matching
- Experiments

Clustering - using a small set of centers to approximate a large data set.

k-centers clustering - minimize the maximum cluster radius

Formally:
Input: k, a set S of n points, a metric d
Find:

$$\arg \min_{S' \subseteq S, |S'|=k} \max_{s \in S} d(s, S')$$

where $d(s, S') = \min_{s' \in S'} d(s, s')$.

- The k-centers problem is NP-hard (up to a 2-approximation)

- The k-centers problem is NP-hard (up to a 2-approximation)
- **Gonzalez** gives a greedy 2-approximation algorithm

- The k-centers problem is NP-hard (up to a 2-approximation)
- **Gonzalez** gives a greedy 2-approximation algorithm
  - Choose the first center arbitrarily

- The k-centers problem is NP-hard (up to a 2-approximation)
- **Gonzalez** gives a greedy 2-approximation algorithm
  - Choose the first center arbitrarily
  - Choose each center as the farthest from the previously selected centers

- The k-centers problem is NP-hard (up to a 2-approximation)
- **Gonzalez** gives a greedy 2-approximation algorithm
  - Choose the first center arbitrarily
  - Choose each center as the farthest from the previously selected centers
  - $O(n)$ time to choose each center, whole algorithm is $O(nk)$

Fairness - removing inherent bias in an algorithm.

- Not necessarily an inherent mathematical concept

To add fairness:

- Items in S have a **demographic group** property
- Each dem. group i gets k_i centers
- $\sum_{i=1}^{m} k_i = k$

In these slides, we use "fair" to mean satisfying all k_i as upper bounds.

Multiple papers present algorithms for fair k-centers:

- **Chen et al.** presented a 3-approximation algorithm, runs in $\Omega(n^2 \log n)$

Multiple papers present algorithms for fair k-centers:

- **Chen et al.** presented a 3-approximation algorithm, runs in $\Omega(n^2 \log n)$
- **Kleindessner et al.** introduced an $O(nkm^2 + km^4)$ algorithm with guaranteed approximation factor $3 \cdot 2^{m-1} - 1$

Multiple papers present algorithms for fair k-centers:

- **Chen et al.** presented a 3-approximation algorithm, runs in $\Omega(n^2 \log n)$
- **Kleindessner et al.** introduced an $O(nkm^2 + km^4)$ algorithm with guaranteed approximation factor $3 \cdot 2^{m-1} - 1$

We present an $O(nk)$-time 3-approximation algorithm for fair k-centers

Our Approach
Overview

A high-level overview of the algorithm is as follows:

- Obtain k initial (unfair) centers, using **Gonzalez**

Our Approach
Overview

A high-level overview of the algorithm is as follows:

- Obtain k initial (unfair) centers, using **Gonzalez**
- Find the largest prefix of these which can be "shifted fairly"

Our Approach

Overview

A high-level overview of the algorithm is as follows:

- Obtain k initial (unfair) centers, using **Gonzalez**
- Find the largest prefix of these which can be "shifted fairly"
- Shift these centers, choose the rest arbitrarily

Our Approach
Overview

A high-level overview of the algorithm is as follows:

- Obtain k initial (unfair) centers, using **Gonzalez**
- Find the largest prefix of these which can be "shifted fairly"
- Shift these centers, choose the rest arbitrarily

The first step is well-defined, how do we accomplish the second and third steps?

Fair Shift - replacing each point with a "neighbor" such that the new set is fair

Does a fair shift exist within radius r for some set of points P?

Fair Shift - replacing each point with a "neighbor" such that the new set is fair

Does a fair shift exist within radius r for some set of points P?

- Draw balls of radius r around the centers

Fair Shift - replacing each point with a "neighbor" such that the new set is fair

Does a fair shift exist within radius r for some set of points P?

- Draw balls of radius r around the centers
- Reduce to matching:

Fair Shift - replacing each point with a "neighbor" such that the new set is fair

Does a fair shift exist within radius r for some set of points P?

- Draw balls of radius r around the centers
- Reduce to matching:
  - Each point in P gets one point in partition A

Fair Shift - replacing each point with a "neighbor" such that the new set is fair

Does a fair shift exist within radius r for some set of points P?

- Draw balls of radius r around the centers
- Reduce to matching:
  - Each point in P gets one point in partition A
  - Each demographic group gets k_i points in partition B

Fair Shift - replacing each point with a "neighbor" such that the new set is fair

Does a fair shift exist within radius r for some set of points P?

- Draw balls of radius r around the centers
- Reduce to matching:
    - Each point in P gets one point in partition A
    - Each demographic group gets k_i points in partition B
    - $ab \in E$ iff point a (in P) has demographic group b in its ball (including a itself)

Fair Shift - replacing each point with a "neighbor" such that the new set is fair

Does a fair shift exist within radius r for some set of points P?

- Draw balls of radius r around the centers
- Reduce to matching:
  - Each point in P gets one point in partition A
  - Each demographic group gets k_i points in partition B
  - $ab \in E$ iff point a (in P) has demographic group b in its ball (including a itself)
- Edges in match of size $|P|$ give a fair shift iff one exists

For runtime, it is more efficient to view this as a maximum flow problem:

- Partition B gets 1 point per demographic group.

Our Approach
Optimizing the Algorithm

For runtime, it is more efficient to view this as a maximum flow problem:

- Partition B gets 1 point per demographic group.
- Add edges from s to Partition A with capacity 1 and from Partition B to t with capacity k_i.

For runtime, it is more efficient to view this as a maximum flow problem:

- Partition B gets 1 point per demographic group.
- Add edges from s to Partition A with capacity 1 and from Partition B to t with capacity k_i.

- Now, each point in S yields at most 1 edge $ab \in E$ so $|E| \leq n + O(k) = O(n)$ and $|V| = 2 + 2k + m = O(k)$

For runtime, it is more efficient to view this as a maximum flow problem:

- Partition B gets 1 point per demographic group.
- Add edges from s to Partition A with capacity 1 and from Partition B to t with capacity k_i.

- Now, each point in S yields at most 1 edge $ab \in E$ so $|E| \leq n + O(k) = O(n)$ and $|V| = 2 + 2k + m = O(k)$

$\Rightarrow O(nk^{1/2})$ time to check for a fair shift, using Dinitz's algorithm.

$O(nk^{1/2})$ time to check for a fair shift
$\Rightarrow$ Use binary search, checking fair shift at each level

- Largest prefix of initial centers with a fair shift

$O(nk^{1/2})$ time to check for a fair shift
$\Rightarrow$ Use binary search, checking fair shift at each level

- Largest prefix of initial centers with a fair shift
    - binary search over k initial centers

Our Approach
Optimizing the Algorithm

$O(nk^{1/2})$ time to check for a fair shift
$\Rightarrow$ Use binary search, checking fair shift at each level

- Largest prefix of initial centers with a fair shift
  - binary search over k initial centers
  - r is maximized such that balls are non-overlapping.

$O(nk^{1/2})$ time to check for a fair shift
$\Rightarrow$ Use binary search, checking fair shift at each level

- Largest prefix of initial centers with a fair shift
  - binary search over k initial centers
  - r is maximized such that balls are non-overlapping.
- Optimizing the shift radius on largest prefix

$O(nk^{1/2})$ time to check for a fair shift
$\Rightarrow$ Use binary search, checking fair shift at each level

- Largest prefix of initial centers with a fair shift
  - binary search over k initial centers
  - r is maximized such that balls are non-overlapping.
- Optimizing the shift radius on largest prefix
  - binary search over discrete shift radii as r

$O(nk^{1/2})$ time to check for a fair shift
$\Rightarrow$ Use binary search, checking fair shift at each level

- Largest prefix of initial centers with a fair shift
  - binary search over k initial centers
  - r is maximized such that balls are non-overlapping.
- Optimizing the shift radius on largest prefix
  - binary search over discrete shift radii as r
  - at most $km \leq k^2$ such values, so $\log k$ levels

$O(nk^{1/2})$ time to check for a fair shift
$\Rightarrow$ Use binary search, checking fair shift at each level

- Largest prefix of initial centers with a fair shift
  - binary search over k initial centers
  - r is maximized such that balls are non-overlapping.
- Optimizing the shift radius on largest prefix
  - binary search over discrete shift radii as r
  - at most $km \leq k^2$ such values, so $\log k$ levels

$O(nk)$ time total to build all the graphs, so each binary search has time complexity $O(nk + nk^{1/2} \log k) = O(nk)$.

Our Approach
Analysis

Therefore, it takes $O(nk)$ time each to

- run **Gonzalez**'s algorithm

Therefore, it takes $O(nk)$ time each to

- run **Gonzalez**'s algorithm
- find the largest prefix of initial centers

Therefore, it takes $O(nk)$ time each to

- run **Gonzalez**'s algorithm
- find the largest prefix of initial centers
- optimize the fair shift

Therefore, it takes $O(nk)$ time each to

- run **Gonzalez**'s algorithm
- find the largest prefix of initial centers
- optimize the fair shift
- heuristically fill the remaining centers (similar to **Gonzalez**)

Therefore, it takes $O(nk)$ time each to

- run **Gonzalez**'s algorithm
- find the largest prefix of initial centers
- optimize the fair shift
- heuristically fill the remaining centers (similar to **Gonzalez**)

For performance,

Therefore, it takes $O(nk)$ time each to

- run **Gonzalez**'s algorithm
- find the largest prefix of initial centers
- optimize the fair shift
- heuristically fill the remaining centers (similar to **Gonzalez**)

For performance,

- fair shift costs no more than $cost_{OPT}$

Our Approach
Analysis

Therefore, it takes $O(nk)$ time each to

- run **Gonzalez**'s algorithm
- find the largest prefix of initial centers
- optimize the fair shift
- heuristically fill the remaining centers (similar to **Gonzalez**)

For performance,

- fair shift costs no more than cost_{OPT}
- largest prefix with a fair shift has objective cost at most $2 \cdot \text{cost}_{OPT}$

Therefore, it takes $O(nk)$ time each to

- run **Gonzalez**'s algorithm
- find the largest prefix of initial centers
- optimize the fair shift
- heuristically fill the remaining centers (similar to **Gonzalez**)

For performance,

- fair shift costs no more than cost_{OPT}
- largest prefix with a fair shift has objective cost at most $2 \cdot \text{cost}_{OPT}$

$\Rightarrow$ 3-approximation algorithm with $O(nk)$ runtime.

We compared the following methods with **Kleindessner et al.**:

- **Alg 2-Seq** - our fair k-centers algorithm, arbitrarily picks centers at the last step
- **Alg 2-Heu B** - our fair k-centers algorithm, uses Heuristic B at the last step.
- **Heuristic A** - runs **Gonzalez** for each demographic group i.
- **Heuristic B** - runs **Gonzalez** but only keep centers that don't violate fairness.
- **Heuristic C** - similar to A, but use distance to centers from all demographic groups.

Experiments

Simulated Data

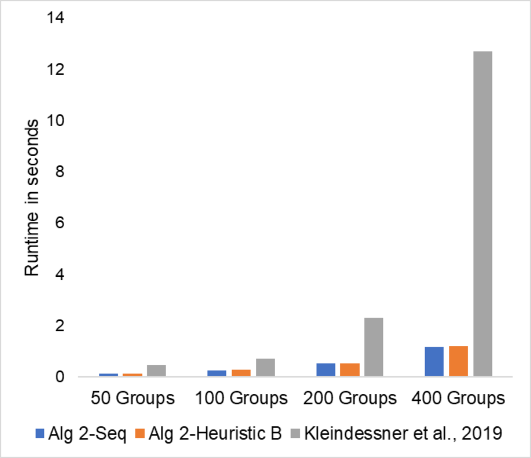

Figure: Mean runtime in seconds on simulated data

Table: Mean and standard deviation of objective value on simulated data

Algo \ Task	50 Groups	100 Groups	250 Groups	500 Groups
Alg 2-Seq	**6.89 (0.2)**	6.52 (0.31)	**6.5 (0.41)**	6.46 (0.38)
Alg 2-Heu B	6.91 (0.26)	**6.48 (0.25)**	6.51 (0.43)	**6.44 (0.38)**
Kleindessner et al.	7.01 (0.46)	6.88 (0.75)	7.45 (0.78)	7.26 (0.51)
Heuristic A	21.38 (2.84)	17.7 (1.55)	16.61 (1.57)	13.87 (1.33)
Heuristic B	7.66 (1.09)	8.16 (0.94)	7.81 (0.71)	7.8 (0.62)
Heuristic C	7.26 (1.17)	7.43 (0.87)	7.44 (0.6)	7.42 (0.62)

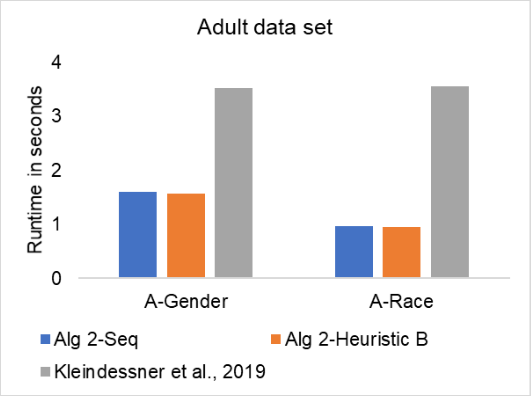

Figure: Adult dataset runtime

Experiments
Real Data

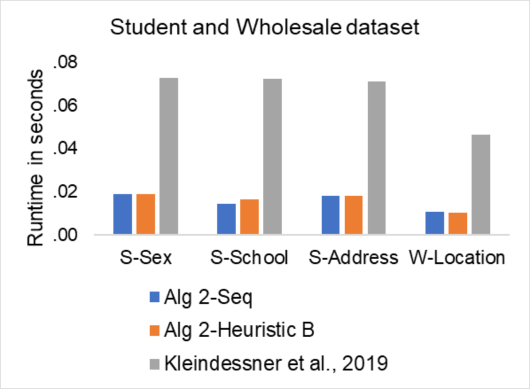

Figure: Student and wholesale dataset runtime

Experiments
Real Data

Table: Mean and standard deviation of objective value on real data

Algo \ Task	A-Gender	A-Race	S-Sex	S-School	S-Address	W-Location
Alg 2-Seq	**0.32 (0.01)**	**0.32 (0.01)**	1.29 (0.04)	1.3 (0.04)	1.31 (0.05)	0.26 (0.01)
Alg 2-Heu B	**0.32 (0.01)**	**0.32 (0.01)**	**1.28 (0.03)**	**1.28 (0.04)**	**1.3 (0.04)**	0.26 (0.01)
Kleindessner et al.	0.36 (0.03)	0.34 (0.02)	1.29 (0.05)	1.29 (0.06)	**1.3 (0.05)**	0.27 (0.03)
Heuristic A	0.41 (0.02)	0.35 (0.03)	1.36 (0.02)	1.39 (0.04)	1.37 (0.04)	0.28 (0.01)
Heuristic B	0.37 (0.02)	**0.32 (0.01)**	1.29 (0.03)	1.3 (0.04)	**1.3 (0.04)**	0.27 (0.01)
Heuristic C	0.4 (0.02)	**0.32 (0.02)**	1.29 (0.03)	1.29 (0.02)	1.35 (0.05)	**0.24 (0.02)**

Summary

Our Algorithm:

- 3-approximation
- $O(nk)$ runtime
- Best algorithm in both runtime and performance
- Experimental support