Individual Fairness for *k*-Clustering

Sepideh Mahabadi
TTIC

Ali Vakilian
University of Wisconsin-Madison

Fairness

Classic **Objective** in Algorithms

- Accuracy
- Runtime
- Space

Fairness

Classic **<u>Objective</u>** in Algorithms

- Accuracy
- Runtime
- Space

But, user interactions with algorithms have other implications

- Loans Approval
- Candidate for Job Interviews
- Candidate for Medical Treatments

Bias in Algorithms?

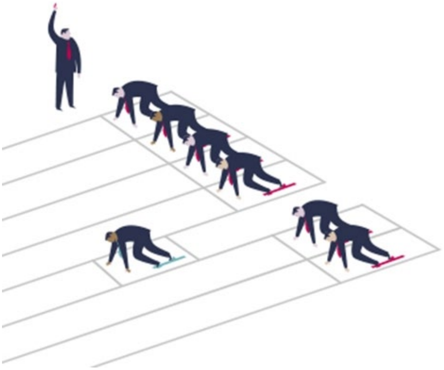
Notions of Fairness

Notions of Fairness

Group Fairness

Notions of Fairness

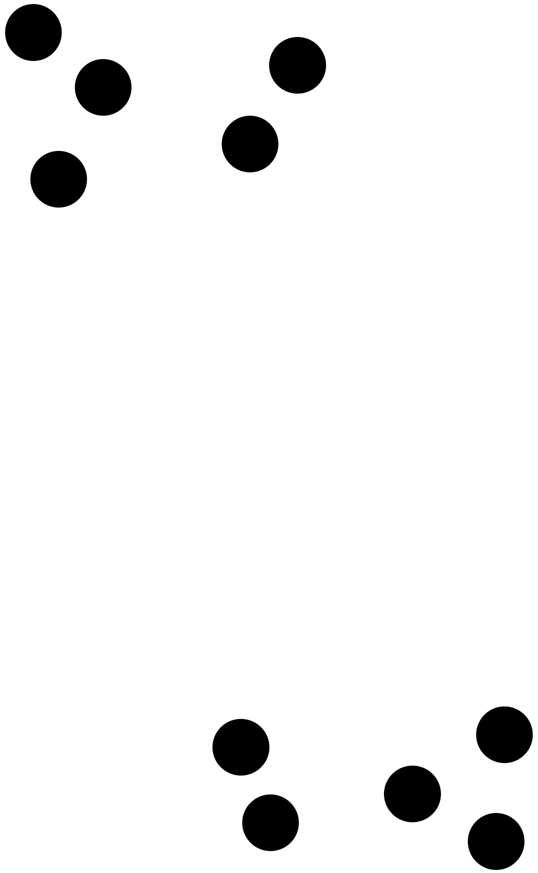
Group Fairness



Individual Fairness

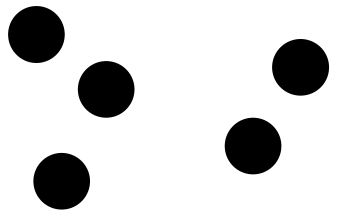
Fairness for Clustering

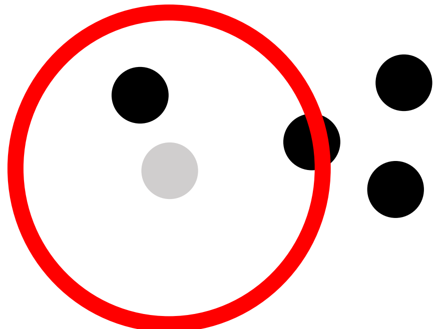
- **Clustering**: one of the most important *unsupervised learning* tasks



Fairness for Clustering

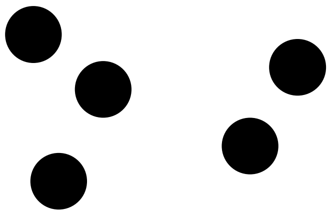
- **Clustering**: one of the most important *unsupervised learning* tasks
- **Individual Fairness** for Clustering [Jung-Kannan-Lutz'20]





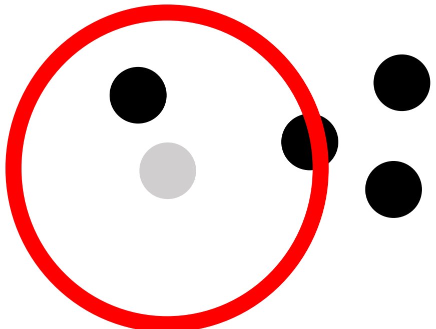
Fairness for Clustering

- **Clustering**: one of the most important *unsupervised learning* tasks
- **Individual Fairness** for Clustering [Jung-Kannan-Lutz'20]



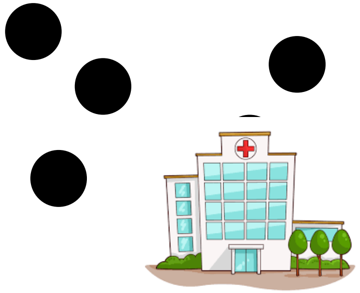
Picking k centers,

Each expects to see a center among its $\frac{n}{k}$ closest neighbors



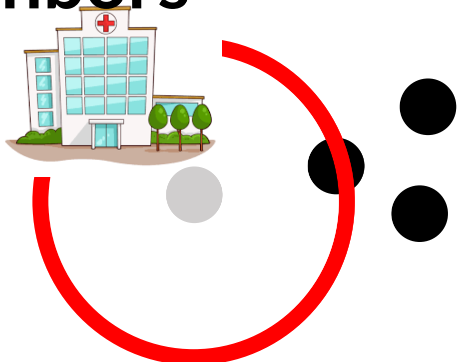
Fairness for Clustering

- **Clustering**: one of the most important *unsupervised learning* tasks
- **Individual Fairness** for Clustering [Jung-Kannan-Lutz'20]

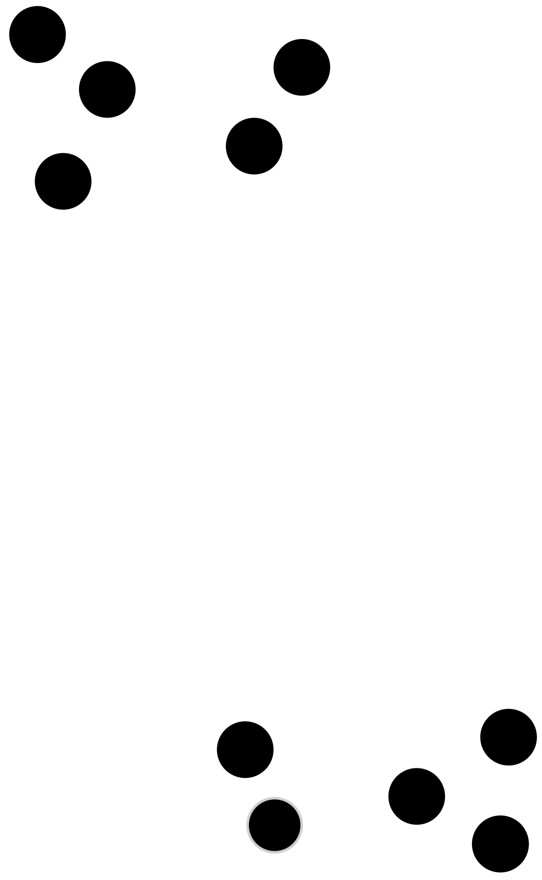


Picking k centers,

Each expects to see a center among its $\frac{n}{k}$ closest neighbors

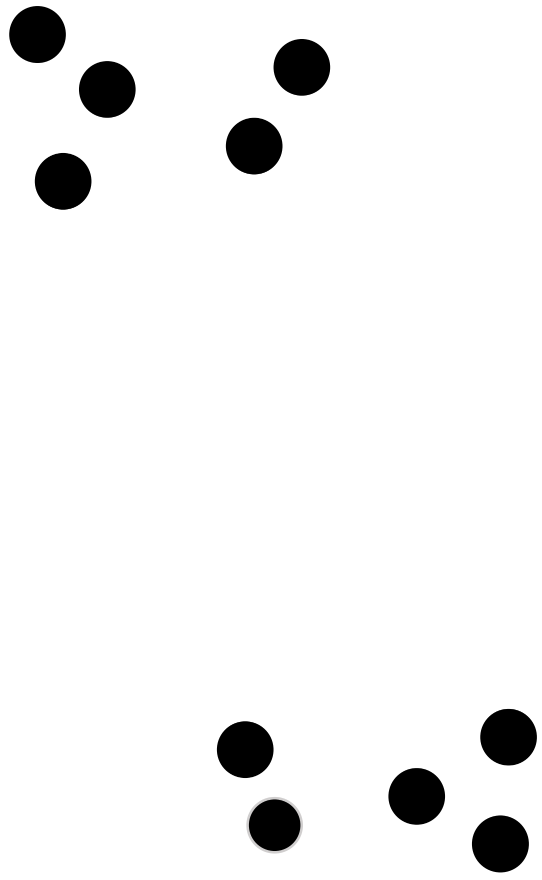


Clustering with Individual Fairness



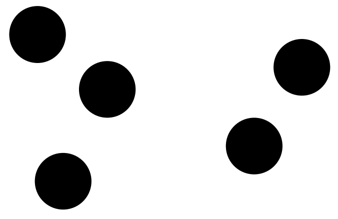
Clustering with Individual Fairness

- Collection of n points in a metric space

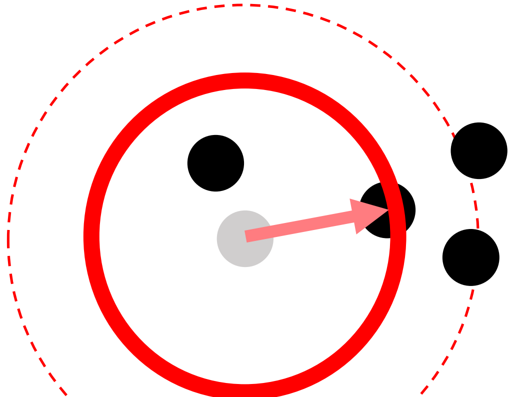


Clustering with Individual Fairness

- Collection of n points in a metric space

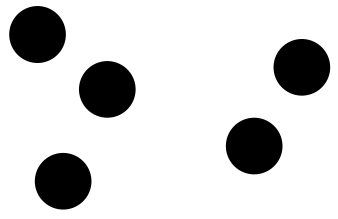


- **Fair Radius ($r_{\text{fair}}(v)$):**
  distance of v to its (n/k)-th closest neighbor



Clustering with Individual Fairness

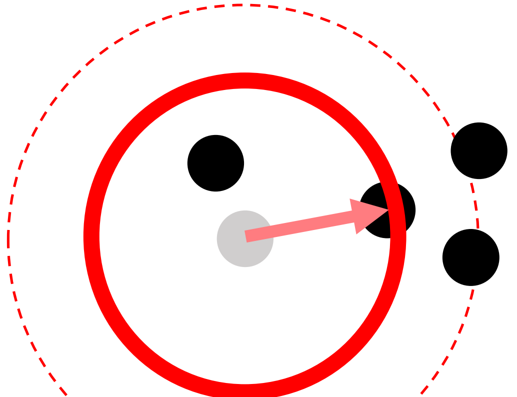
- Collection of n points in a metric space



- **Fair Radius** ($r_{\text{fair}}(v)$):

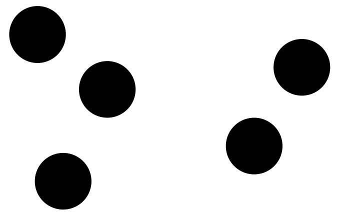
  distance of v to its (n/k)-th closest neighbor

[JKL20] **NP-hard** to decide whether there exists a fair solution



Clustering with Individual Fairness

- Collection of n points in a metric space

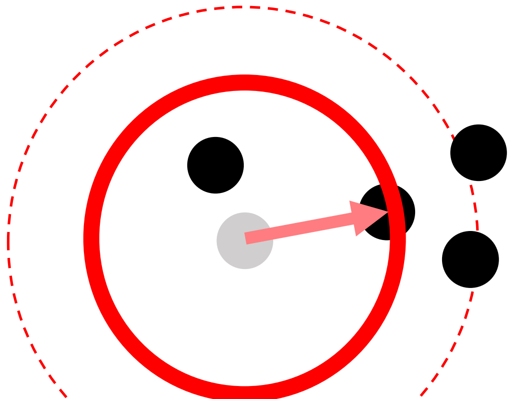


- **Fair Radius ($r_{\text{fair}}(v)$):**

  distance of v to its (n/k)-th closest neighbor

[JKL20] **NP-hard** to decide whether there exists a fair solution

- **α-Fair k-Clustering**: Find **k centers** s.t. each v has a center in distance at most $\alpha \times$ its fair radius (i.e., $r_{\text{fair}}(v)$)



Prior Work

[Jung, Kannan, Lutz'20] introduces this notion of fairness

Prior Work

[Jung, Kannan, Lutz'20] introduces this notion of fairness

- Finds a **2-fair** solution

Prior Work

[Jung, Kannan, Lutz'20] introduces this notion of fairness

- Finds a **2-fair** solution
  - ❑ Each v has a center in distance at most $2 \cdot r_{\text{fair}}(v)$

Prior Work

[Jung, Kannan, Lutz'20] introduces this notion of fairness

- Finds a **2-fair** solution
  - ❑ Each v has a center in distance at most $2 \cdot r_{\text{fair}}(v)$
  - ❑ In [Chan-Dinitz-Gupta'06] & [Charikar-Makarychev-Makarychev'10]

Prior Work

[Jung, Kannan, Lutz'20] introduces this notion of fairness

- Finds a **2-fair** solution
  - ❑ Each v has a center in distance at most $2 \cdot r_{\text{fair}}(v)$
  - ❑ In [Chan-Dinitz-Gupta'06] & [Charikar-Makarychev-Makarychev'10]

Only a fair solution, without optimizing the clustering cost

Prior Work

[Jung, Kannan, Lutz'20] introduces this notion of fairness

- Finds a **2-fair** solution
  - ❑ Each v has a center in distance at most $2 \cdot r_{\text{fair}}(v)$
  - ❑ In [Chan-Dinitz-Gupta'06] & [Charikar-Makarychev-Makarychev'10]

Only a fair solution, without optimizing the clustering cost

- ***k-median***: total distance of points to their centers
- ***k-means***: total squared distance of points to their centers
- ***k-center***: maximum distance of any point to its center

Our Results

[This work] Local search returns (O(1),O(1)) approximation for α-fair k-median clustering.

Our Results

[This work] Local search returns (O(1),O(1)) approximation for α-fair k-median clustering. *I.e., finds a set of k centers* ***C*** *s.t.*

- ***C*** *is O(α)-fair and*
- *cost of* ***C*** *is at most constant times the cost of an optimal α-fair solution*

Our Results

[This work] Local search returns (O(1),O(1)) approximation for α-fair k-median clustering. *I.e., finds a set of k centers* ***C*** *s.t.*

- ***C*** *is O(α)-fair and*
- *cost of* ***C*** *is at most constant times the cost of an optimal α-fair solution*

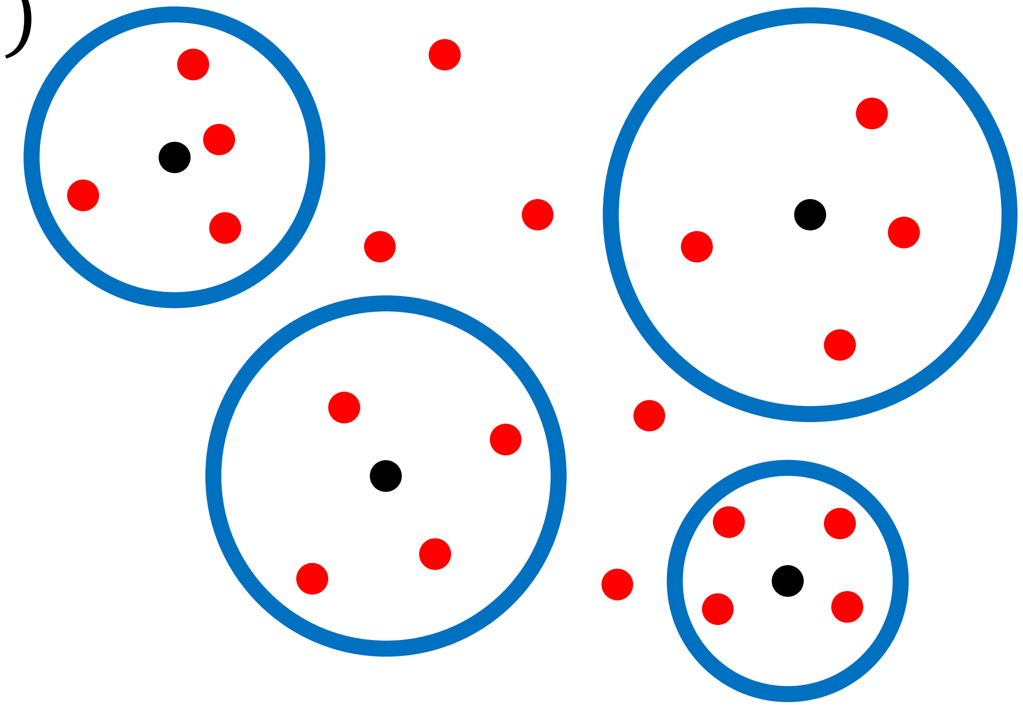
Our algorithm works for any ℓ_p*-norm objective (e.g., k-means & k-center)*

- ***k-means:*** *O(α)-fair and cost is constant times the optimal α-fair k-means*
- ***k-center:*** *O(α)-fair and cost is O($\log n$) times the optimal α-fair k-center*

Outline of Our Algorithm

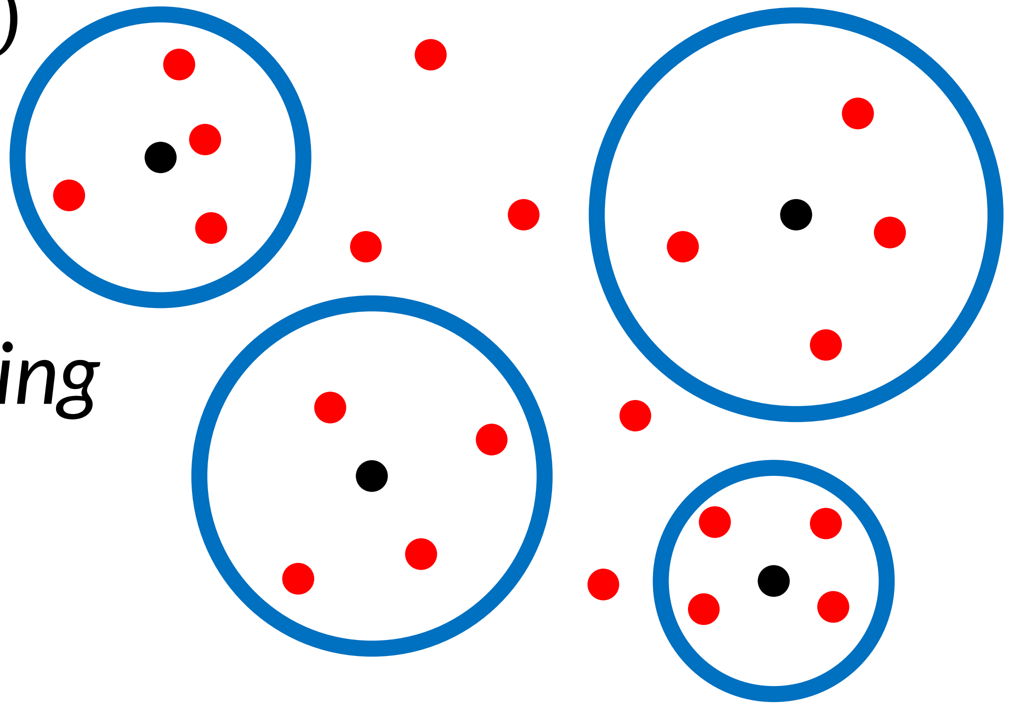
Step 1. Find a set of **critical balls** of size $\ell \leq k$

$\boldsymbol{B}\big(c_1, \alpha r_{\text{fair}}(c_1)\big), \ldots, \boldsymbol{B}(c_k, \alpha r_{\text{fair}}(c_\ell))$



Outline of Our Algorithm

Step 1. Find a set of **critical balls** of size $\ell \leq k$
$\boldsymbol{B}\left(c_1, \alpha r_{\text{fair}}(c_1)\right), \ldots, \boldsymbol{B}(c_k, \alpha r_{\text{fair}}(c_\ell))$



Lemma. *Any set of k centers intersecting with critical balls is O(α)-fair.*

Outline of Our Algorithm

Step 1. Find a set of **critical balls** of size $\ell \leq k$

Step 2. Run **local search** algorithm

Outline of Our Algorithm

Step 1. Find a set of **critical balls** of size $\ell \leq k$

Step 2. Run **local search** algorithm

Step 2-a. Find an **initial (fair) solution**

Outline of Our Algorithm

Step 1. Find a set of **critical balls** of size $\ell \leq k$

Step 2. Run **local search** algorithm

Step 2-a. Find an **initial (fair) solution**

I. Pick the centers of critical balls
II. In $k - \ell$ remaining iterations
    I. Pick the furthest point from the current set of centers

Outline of Our Algorithm

Step 1. Find a set of **critical balls** of size $\ell \leq k$

Step 2. Run **local search** algorithm

Step 2-a. Find an **initial (fair) solution** ***O(n)*-approximation**

I. Pick the centers of critical balls
II. In $k - \ell$ remaining iterations
    I. Pick the furthest point from the current set of centers

Outline of Our Algorithm

Step 1. Find a set of **critical balls** of size $\ell \leq k$

Step 2. Run **local search** algorithm

- *Step 2-a.* Find an **initial (fair) solution**
- *Step 2-b.* **Local Update**

***O(n)*-approximation**

Outline of Our Algorithm

Step 1. Find a set of **critical balls** of size $\ell \leq k$

Step 2. Run **local search** algorithm

Step 2-a. Find an **initial (fair) solution** **O(*n*)-approximation**

Step 2-b. **Local Update**

*Is there a **swap** of size at most t reducing cost by (1+ϵ)?*

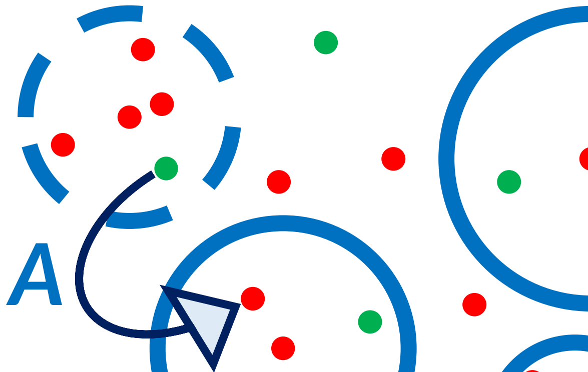
Outline of Our Algorithm

Step 1. Find a set of **critical balls** of size $\ell \leq k$

Step 2. Run **local search** algorithm

- *Step 2-a.* Find an **initial (fair) solution** — **$O(n)$-approximation**
- *Step 2-b.* **Local Update**

  *Is there a **swap** of size at most t reducing cost by (1+ϵ)?*



Outline of Our Algorithm

Step 1. Find a set of **critical balls** of size $\ell \leq k$

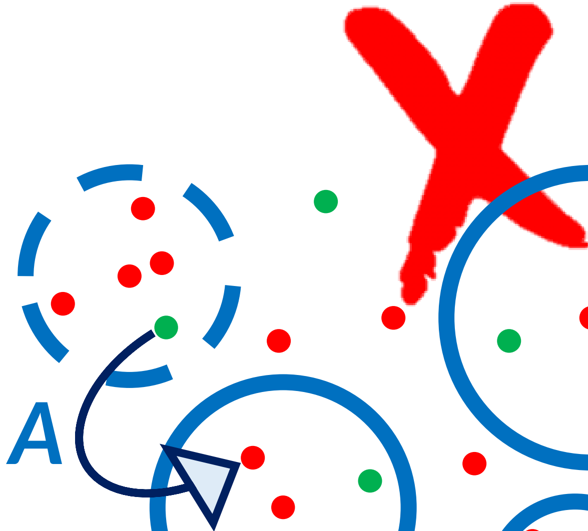
Step 2. Run **local search** algorithm

Step 2-a. Find an **initial (fair) solution** **$O(n)$-approximation**

Step 2-b. **Local Update**

*Is there a **swap** of size at most t reducing cost by (1+ϵ)?*

- Not all swaps are feasible



Outline of Our Algorithm

Step 1. Find a set of **critical balls** of size $\ell \leq k$

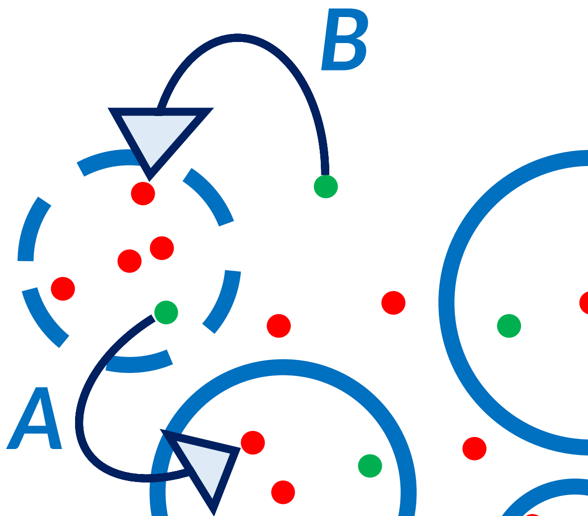
Step 2. Run **local search** algorithm

Step 2-a. Find an **initial (fair) solution** **$O(n)$-approximation**

Step 2-b. **Local Update**

*Is there a **swap** of size at most t reducing cost by (1+ϵ)?*

- Not all swaps are feasible



Outline of Our Algorithm

Step 1. Find a set of **critical balls** of size $\ell \leq k$

Step 2. Run **local search** algorithm

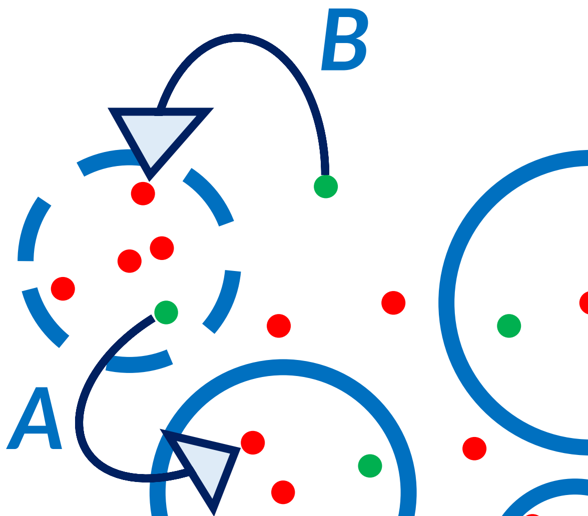
Step 2-a. Find an **initial (fair) solution** **$O(n)$-approximation**

Step 2-b. **Local Update**

*Is there a **swap** of size at most t reducing cost by (1+ϵ)?*

- Not all swaps are feasible

requires more complex analysis



Outline of Our Algorithm

Step 1. Find a set of **critical balls** of size $\ell \leq k$

Step 2. Run **local search** algorithm

Step 2-a. Find an **initial (fair) solution** ***O(n)*-approximation**

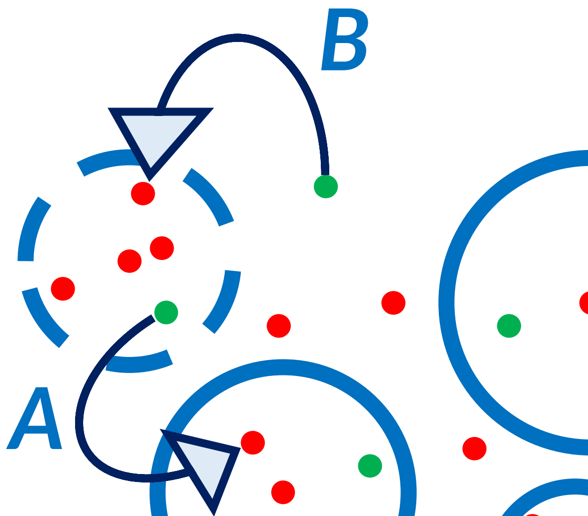
Step 2-b. **Local Update**

Is there a ***swap*** *of size at most t reducing cost by (1+ϵ)?*

- Not all swaps are feasible

  requires more complex analysis

- “valid” swaps of size at most four suffices



Outline of Our Algorithm

Step 1. Find a set of **critical balls** of size $\ell \leq k$

Step 2. Run **local search** algorithm

Step 2-a. Find an **initial (fair) solution** O(n)-approximation

Step 2-b. **Local Update**

*Is there a **swap** of size at most t reducing cost by (1+ϵ)?*

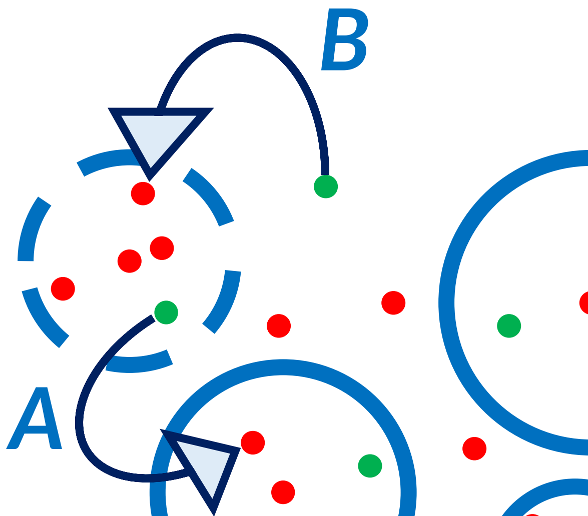
- Not all swaps are feasible

  requires more complex analysis

- "valid" swaps of size at most four suffices

❑ *The algorithm terminates after* $O(\log n/\epsilon)$ *iterations*

❑ *Each iteration runs in* $O(k^4 n^4)$



Empirical Evaluation

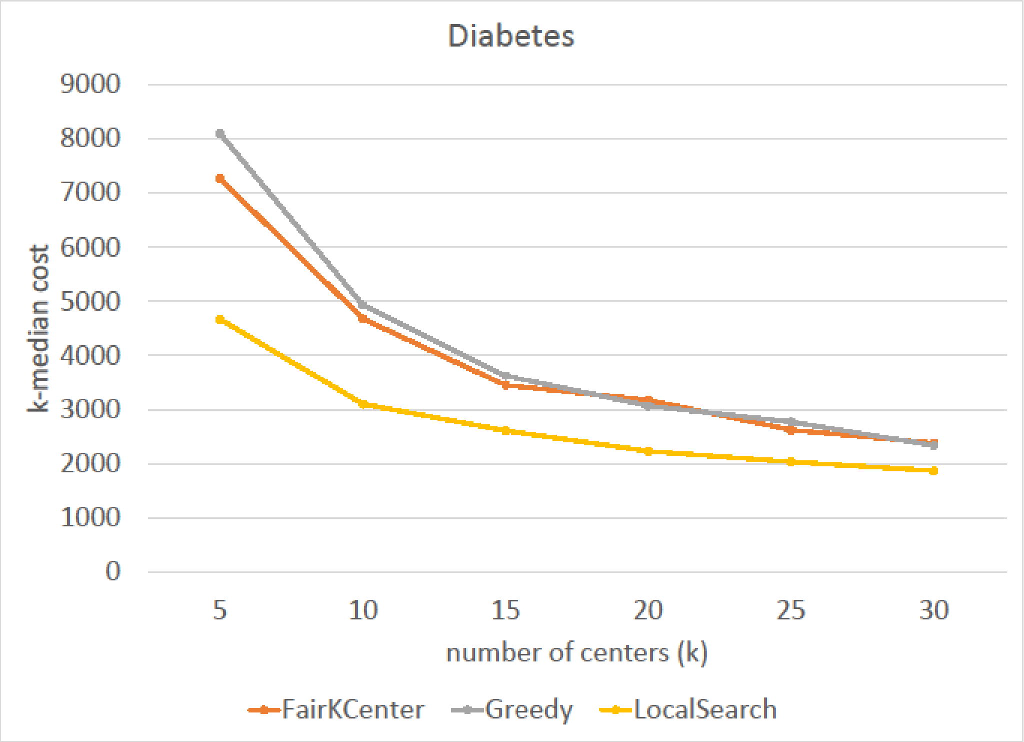
- UCI datasets: Diabetes, Bank, Census

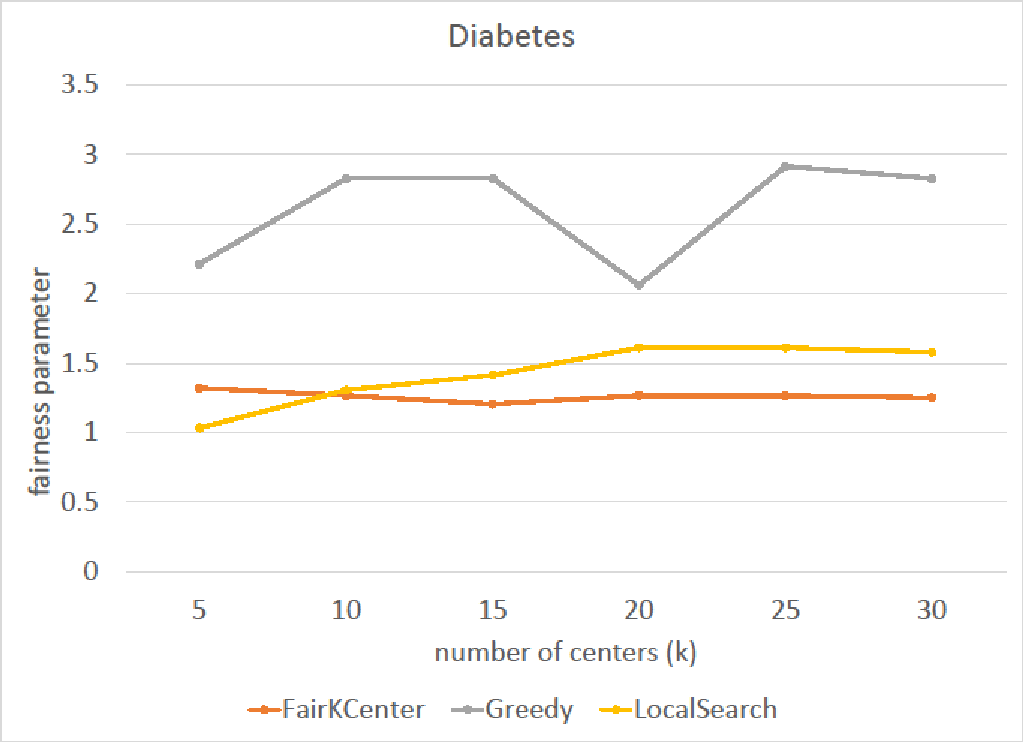
Dataset	Dimension	#Points
Diabetes	2	101,765
Bank	3	4,520
Census	5	32,560

- **Local Search** reports a solution with
    - Better cost than [JKL20] by a factor of 1.4, 2.25, and 1.93
    - Worse fairness than [JKL20] by a factor of 1.13, 1.5, and 1.16

Empirical Evaluation

- UCI datasets: Diabetes, Bank, Census





Future Directions

- Scalable algorithms w.r.t this notion of fairness?
- Better analysis of Local Search algorithm (with smaller swap sizes)?
- Constant-factor approximation for fair *k*-center problem?

  Our result implies a (O(1), O(log n))-approximation algorithm

Future Directions

- Scalable algorithms w.r.t this notion of fairness?
- Better analysis of Local Search algorithm (with smaller swap sizes)?
- Constant-factor approximation for fair *k*-center problem?

  Our result implies a (O(1), O(log n))-approximation algorithm

Thank You