APPLIED MECHANISM DESIGN FOR SOCIAL GOOD

JOHN P DICKERSON & MARINA KNITTEL

Lecture #23 – 04/18/2022
Lecture #24 – 04/20/2022

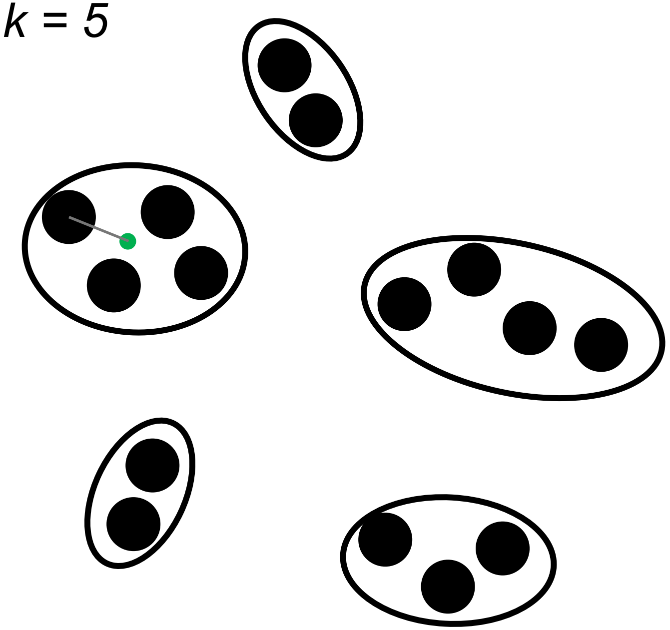
CMSC498T
Mondays & Wednesdays
2:00pm – 3:15pm

COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

CLUSTERING PRIMER

Clustering is the problem of grouping data based off of the distance (or similarity score) between points.

$k = 5$



k-center: minimize the maximum distance from a point to its cluster center.

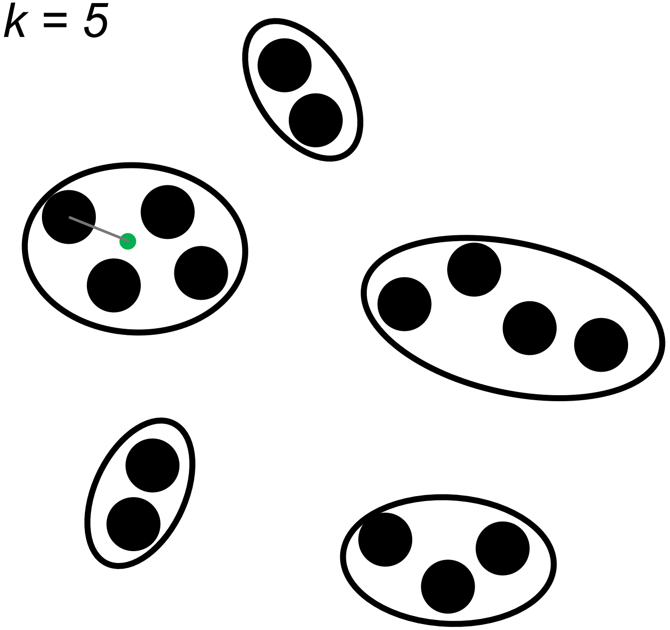
k-median: minimize the sum of distances from points to cluster centers.

k-means: minimize the sum of squares of distances from points to cluster centers.

CLUSTERING PRIMER

Clustering is the problem of grouping data based off of the distance (or similarity score) between points.

k = 5



Generally, we minimize:

$$\sum_{x \in X} ||\varphi(x) - x||^p$$

X is our point set.

φ tells us the center for a point.

p defines what we maximize.

k-center: p = ∞

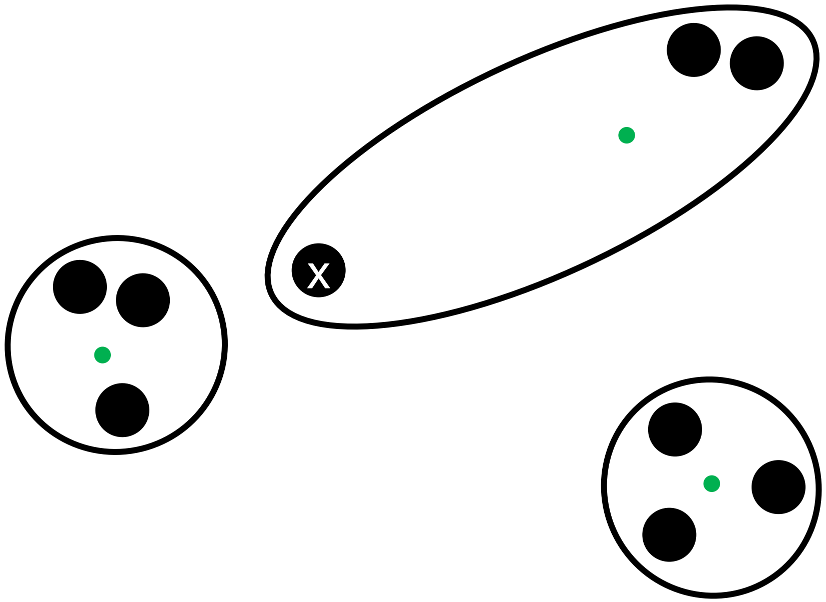
k-median: p = 1

k-center: p = 2

…

CLUSTERING PRIMER

Clustering *x* poorly is only one of many costs in *k-median* since we sum over many points. But for *k-center*, since it is the furthest point from the cluster center, it is very costly, in fact it defines the *k-center* cost.



CLUSTERING PRIMER

We can write an integer linear program to assign points to clusters.

Let $C = c_1, c_2, \dots, c_k$ be the cluster centers.

Let x_i denote that x is assigned to center i.

This may look nonlinear, but it's a constant!

Objective: minimize $\sum_{c_i \in C} \sum_{x \in X} x_i \times \boxed{||c_i - x||^p}$

Constraints: $\sum_{c_i \in C} x_i = 1$ for all $x \in X$ (only assign to 1 center)

$x_i \in \{0,1\}$ for all $x \in X$ and $c_i \in C$

CLUSTERING PRIMER

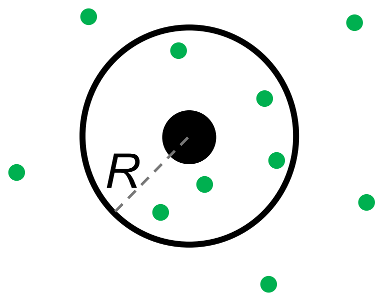
For *k-centers*, we do it a bit differently. We do *not* put anything in the objective. We guess *R* as an upper bound on the distance to centers. *R* is the cost of the solution. We use binary search to find the smallest *R* with a feasible solution.

Objective: None!

Constraints: $\sum_{c_i \in C \cap B_R(x)} x_i = 1$ for all $x \in X$ (only assign to 1 center)

$x_i = 0$ for all $x \in X$ and $c_i \in C \setminus B_R(x)$

$x_i \in \{0,1\}$ for all $x \in X$ and $c_i \in C$



Note: $B_R(x)$ is the ball of radius *R* around *x* (i.e., points within *R* from *x*).

DISPARATE IMPACT

Griggs vs Duke Power Co:

- North Carolina, 1970
- Required high school diploma and standardized testing for promotion
- Sued for discrimination
- Ruled discriminatory by SC because it had “a disproportionate and adverse impact on certain individuals”

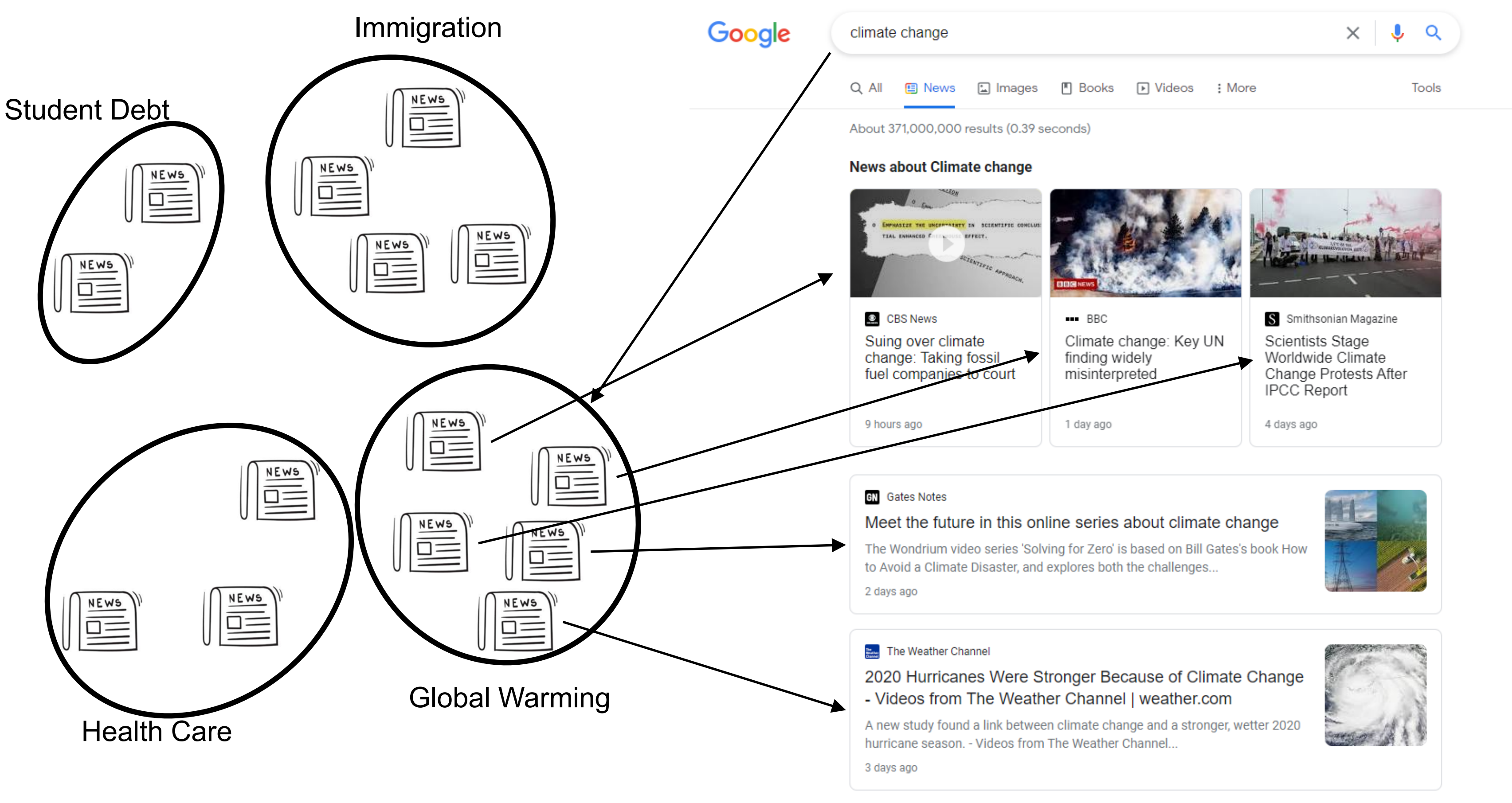
DISPARATE IMPACT IN CLUSTERING

Applying to ML: Ensure the impact of a system across protected groups is proportionate. “Group fairness.”

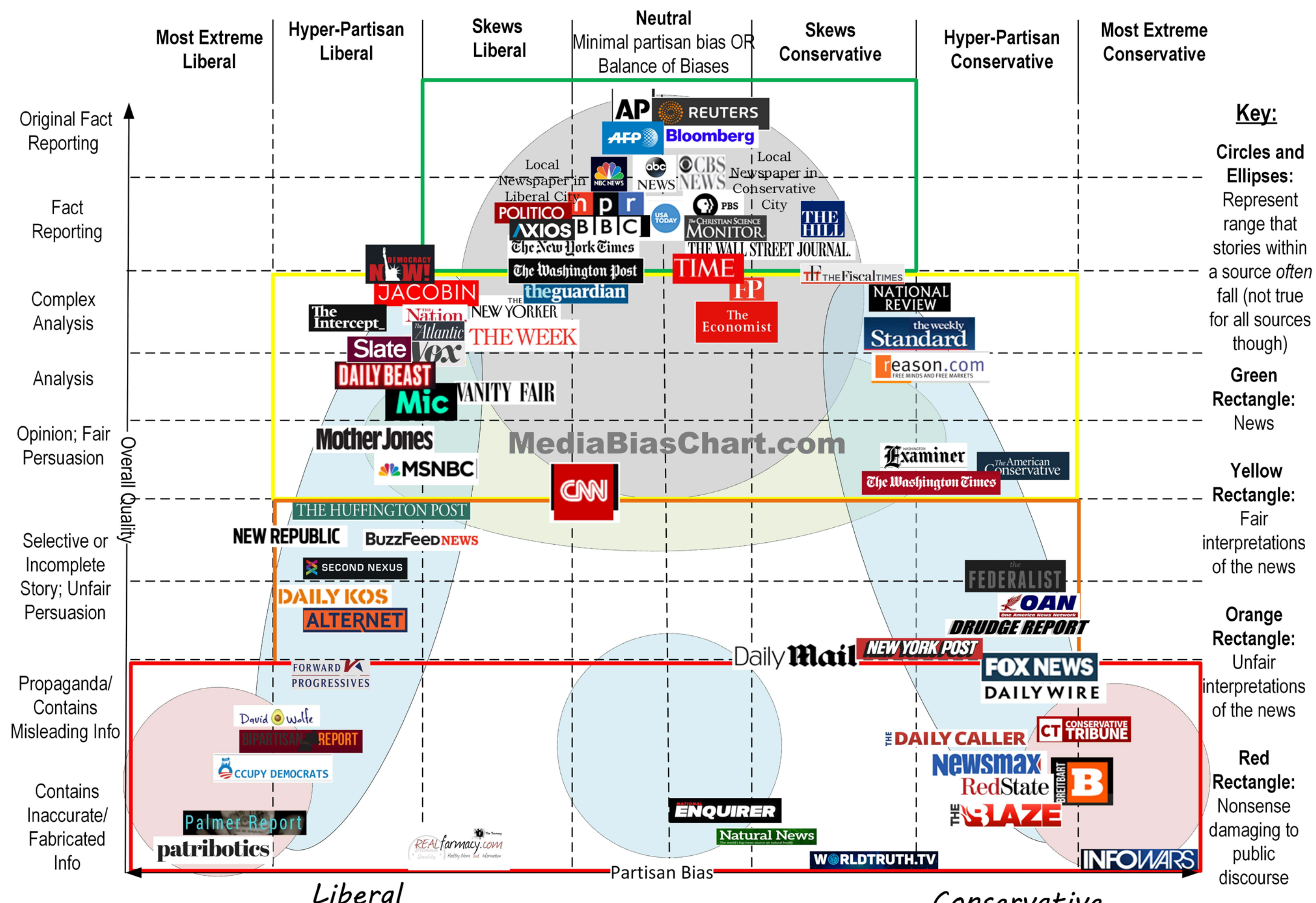
Applying to clustering:

- How do we measure the impact of a system on a protected group?
  - How many individuals are in a cluster
- How do we prevent disparate impact?
  - Ensure the number of individuals from any group in any cluster is proportionate to group size.

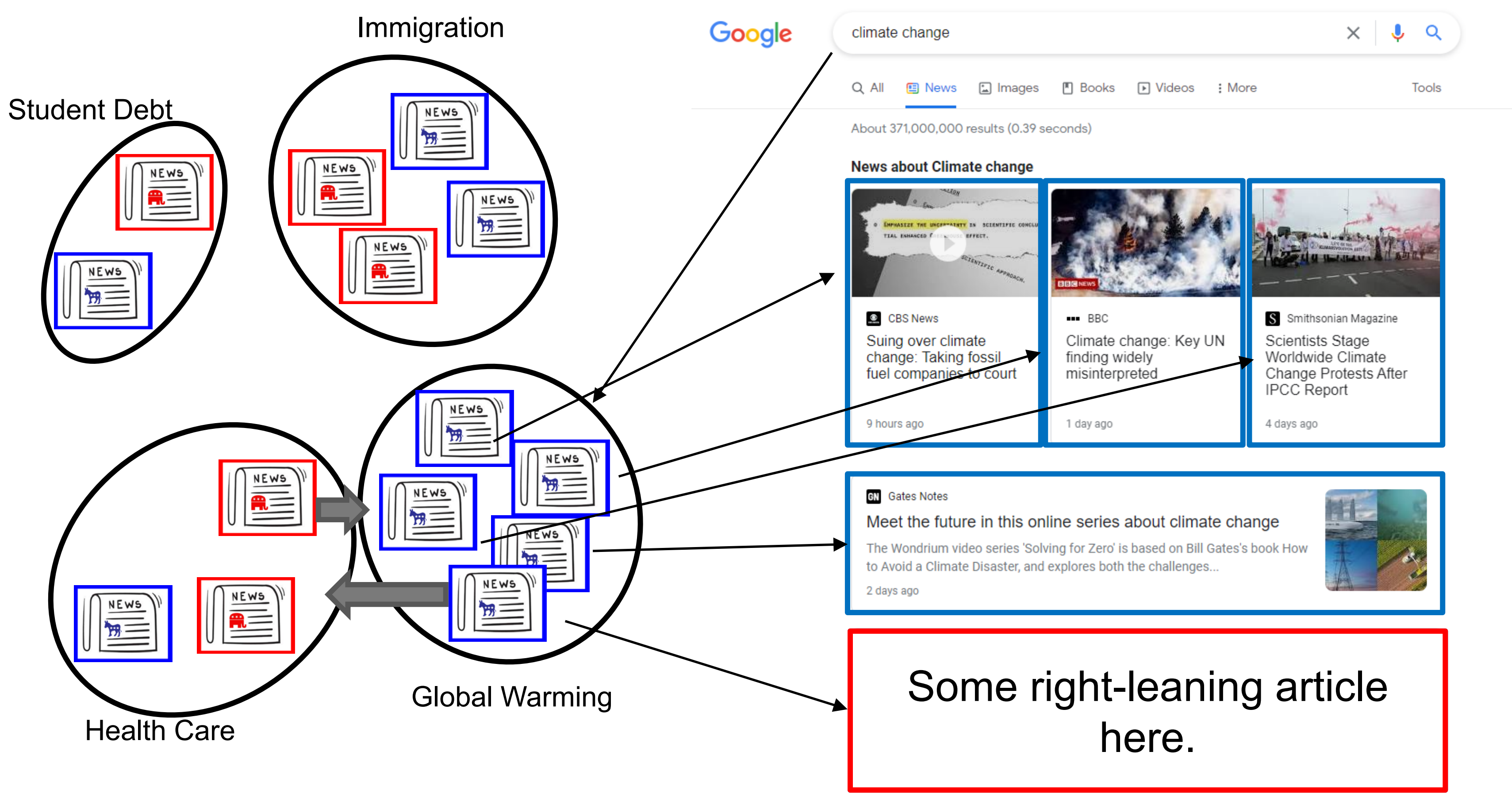
CLUSTERING NEWS ARTICLES



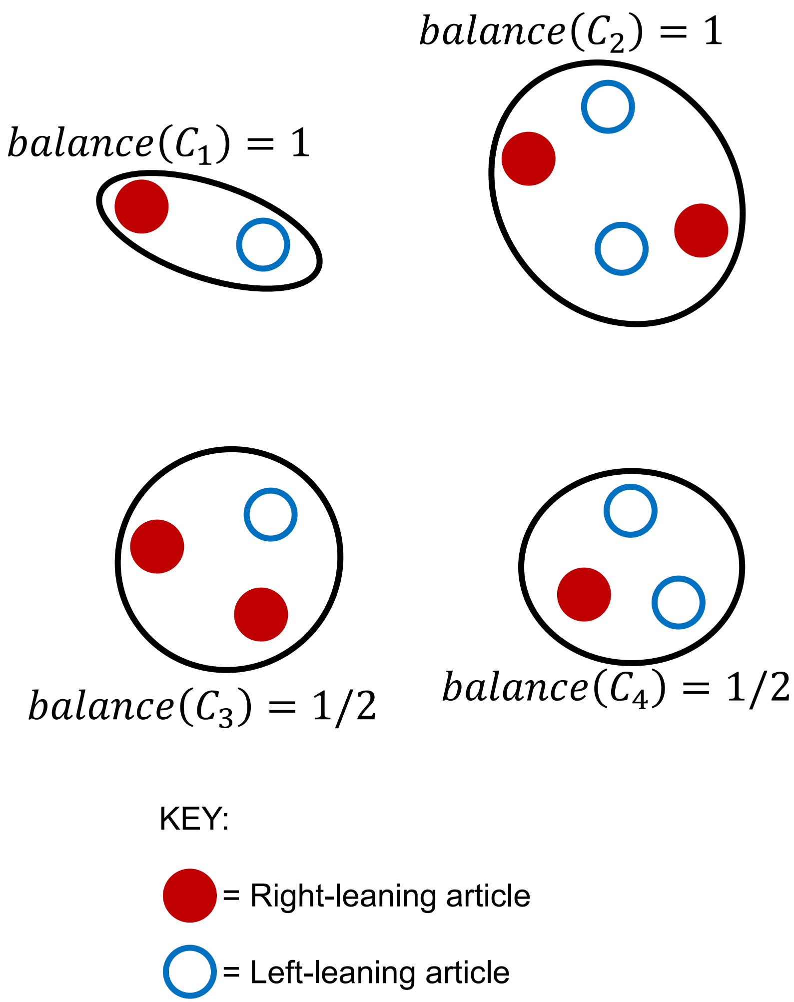
RUNNING EXAMPLE: NEWS SEARCH



CLUSTERING NEWS ARTICLES



GROUP FAIRNESS IN CLUSTERING: FORMAL



For a cluster C:

$$balance(C) \coloneqq \min\left(\frac{\#red(C)}{\#blue(C)}, \frac{\#blue(C)}{\#red(C)}\right)$$

For a clustering S:

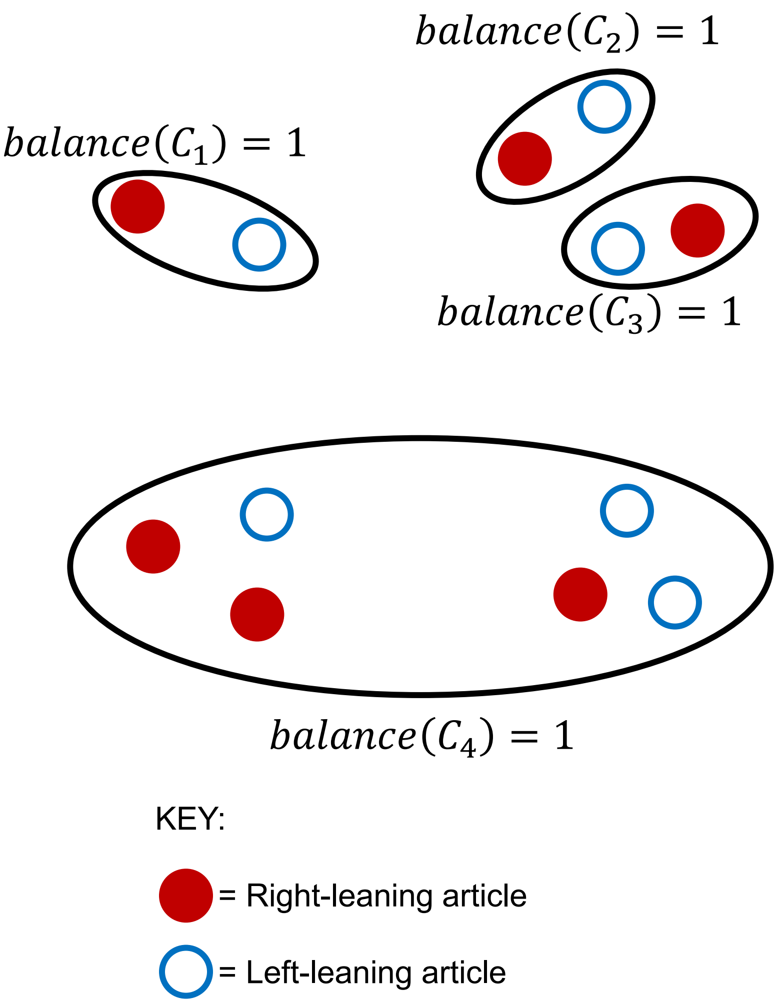
$$balance(S) \coloneqq \min_{C \in S} balance(C)$$

We want balance to be high (close to 1).

Before: $balance(S) = 1/2$

After:

GROUP FAIRNESS IN CLUSTERING: FORMAL



For a cluster C:

$$balance(C) \coloneqq \min\left(\frac{\#red(C)}{\#blue(C)}, \frac{\#blue(C)}{\#red(C)}\right)$$

For a clustering S:

$$balance(S) \coloneqq \min_{C \in S} balance(C)$$

We want balance to be high (close to 1).

Before: $balance(S) = 1/2$

After: $balance(S) = 1$

FINDING FAIR CLUSTERINGS: FAIRLETS

This is the best balance we can achieve

Let:

b/r = 1/3 is the minimum ratio between reds and blues

R = number of remaining reds = 10

B = number of remaining blues = 5

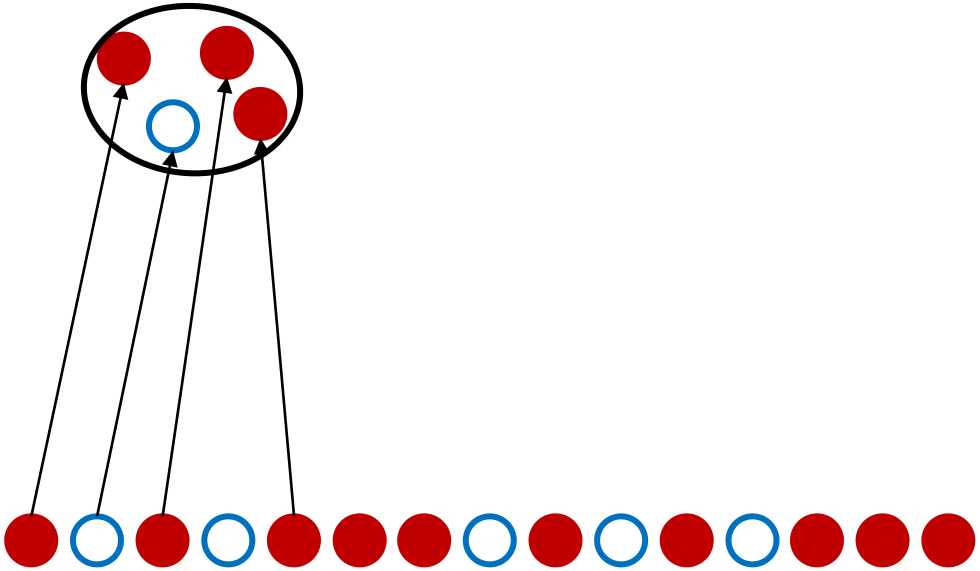
α = 1/3 is our fair parameter

Iteratively…

IF (R-B) ≥ (r-b): make a cluster of r reds and b blues.

ELSE: use (R-B)+b red and b blue points.

When r=b: simply match points



FINDING FAIR CLUSTERINGS: FAIRLETS

This is the best balance we can achieve

Let:

b/r = 1/3 is the minimum ratio between reds and blues

R = number of remaining reds = 7

B = number of remaining blues = 4

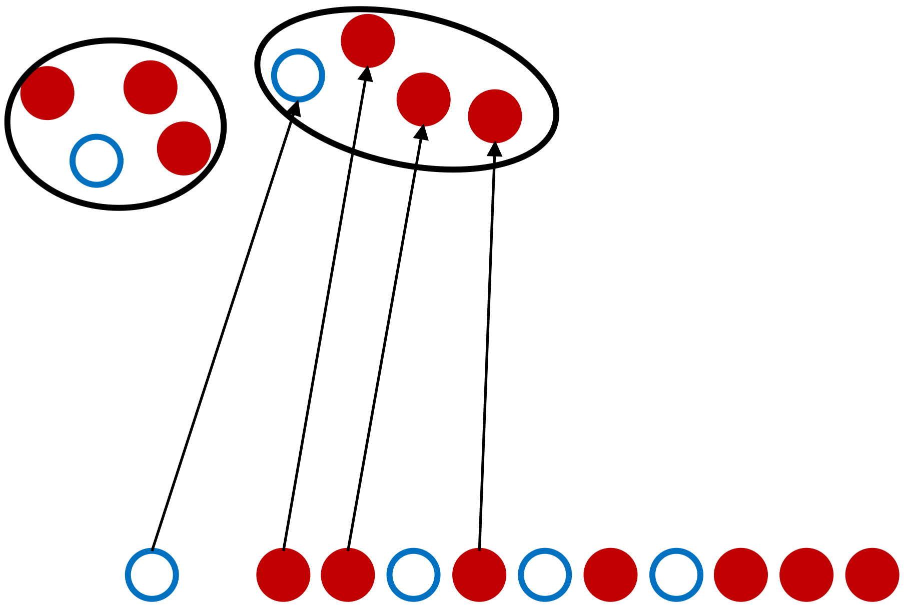
α = 1/3 is our fair parameter

Iteratively…

IF (R-B) ≥ (r-b): make a cluster of r reds and b blues.

ELSE: use (R-B)+b red and b blue points.

When r=b: simply match points



FINDING FAIR CLUSTERINGS: FAIRLETS

This is the best balance we can achieve

Let:

b/r = 1/3 is the minimum ratio between reds and blues

R = number of remaining reds = 4

B = number of remaining blues = 3

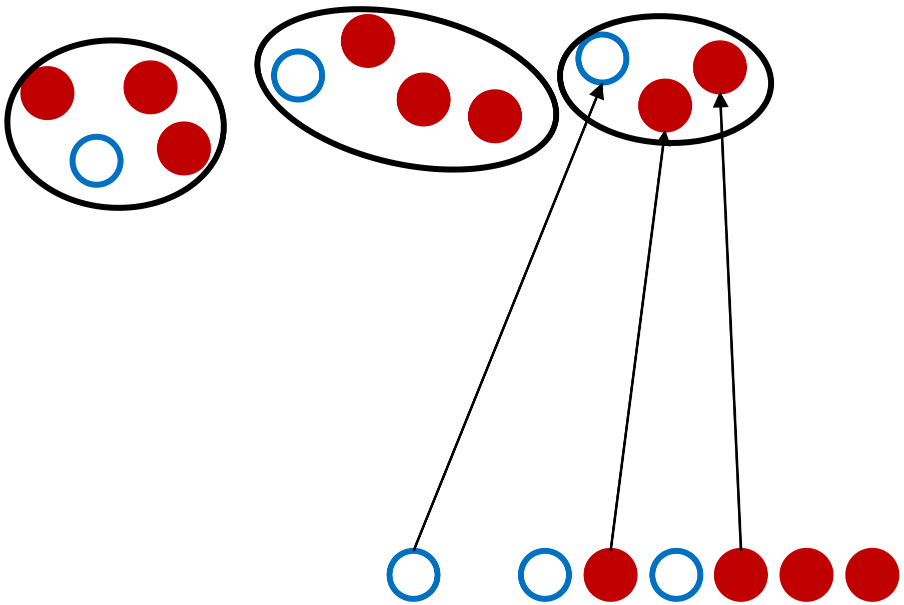
α = 1/3 is our fair parameter

Iteratively…

IF (R-B) ≥ (r-b): make a cluster of r reds and b blues.

ELSE: use (R-B)+b red and b blue points.

When r=b: simply match points



FINDING FAIR CLUSTERINGS: FAIRLETS

This is the best balance we can achieve

Let:

$b/r = 1/3$ is the minimum ratio between reds and blues

R = number of remaining reds = 2

B = number of remaining blues = 2

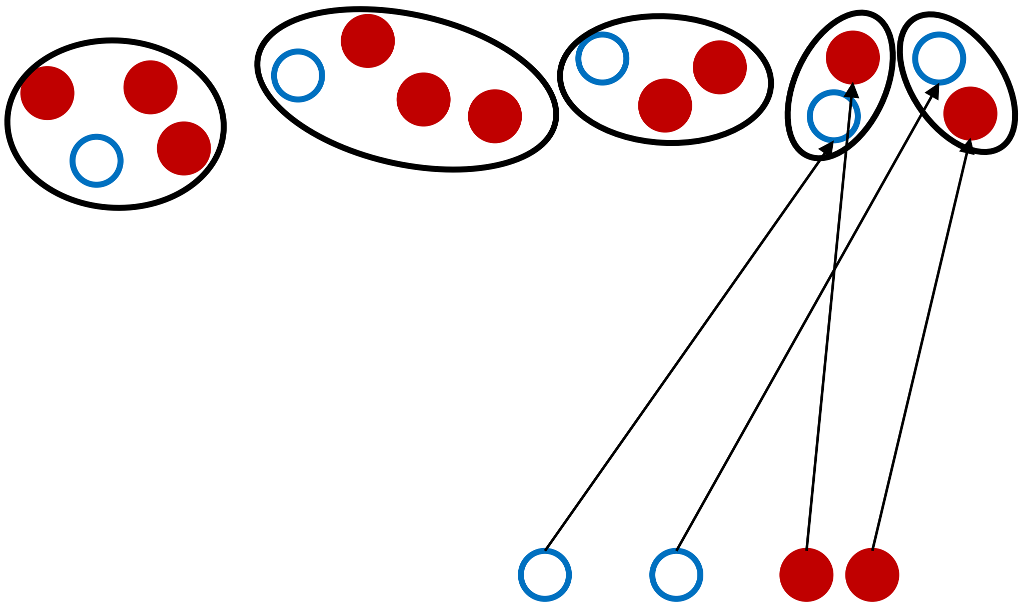
$\alpha = 1/3$ is our fair parameter

Iteratively…

IF $(R-B) \geq (r-b)$: make a cluster of r reds and b blues.

ELSE: use $(R-B)+b$ red and b blue points.

When r=b: simply match points



FINDING FAIR CLUSTERINGS: FAIRLETS

This is
the best balance
we can achieve

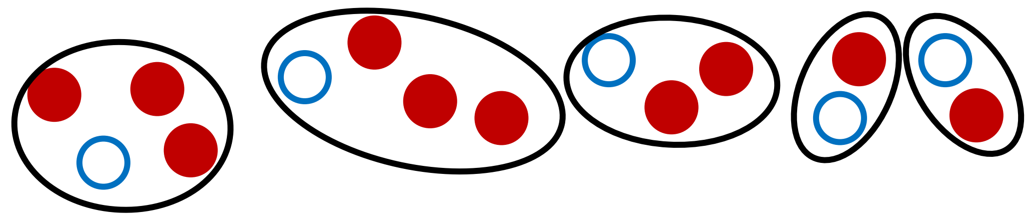
Let:

$\boxed{b/r = 1/3}$ is the minimum ratio between reds and blues

R = number of remaining reds = 0

B = number of remaining blues = 0

α = 1/3 is our fair parameter



Iteratively…

IF (R-B) ≥ (r-b): make a cluster of r reds and b blues.

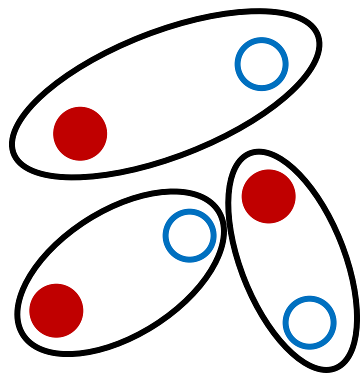
ELSE: use (R-B)+b red and b blue points.

When r=b: simply match points

Idea: Find a "fairlet decomposition" with small fairlets. Find a way to merge them into a good, appropriately-sized clustering.

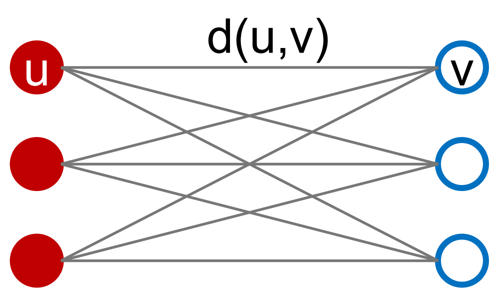
FAIRLETS WHEN B/R=1

Say half of your points are red and half are blue.



Make a bipartite graph between red and blue points. Edge weights are distances.

Find a perfect, maximum weight matching.



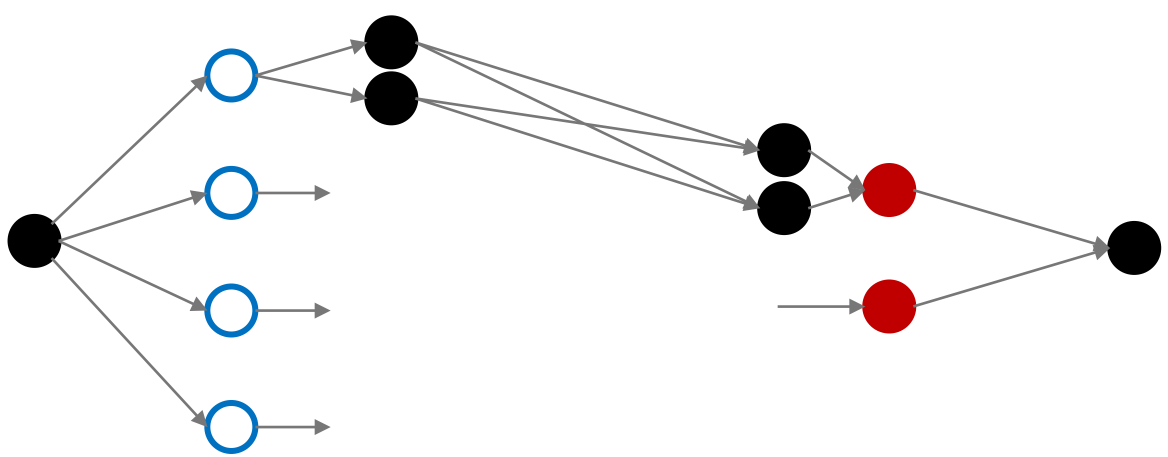
Construct 2-sized fairlets made of the matches.

FAIRLETS WHEN B/R=1/T'

Let B be the number of blues, R be the number of reds. Assume B/R = 1/t' for an integer t'.

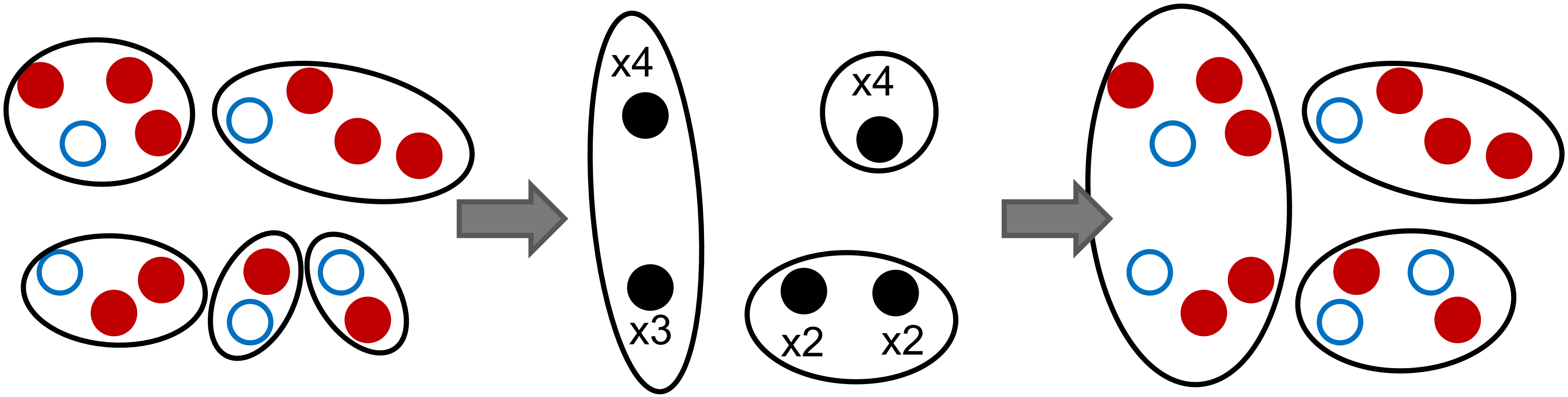
Fairlets can be found using *minimum cost flow* (a network flow problem where edges are labeled with associated edge costs, edge flow capacities, and a flow supply/demand at vertices.

t' = 2



USING FAIRLETS FOR FAIR CLUSTERING

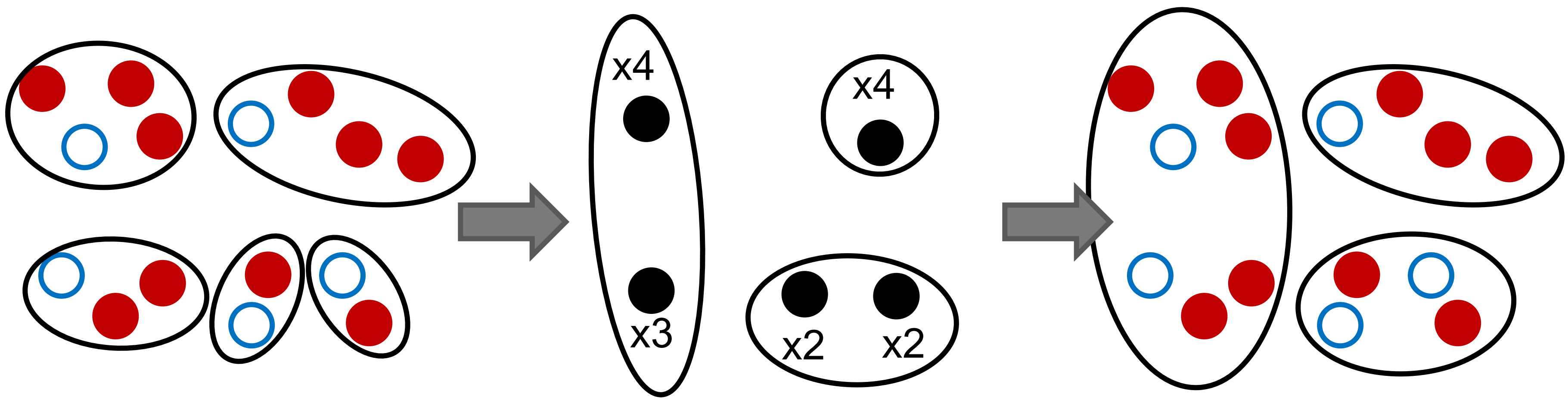
1. Find a *good* fairlet decomposition.
2. Replace each fairlet with a single point. Duplicate that point according to the fairlet size.
3. Run a “vanilla” clustering on these points.
4. Apply this clustering to the original points.



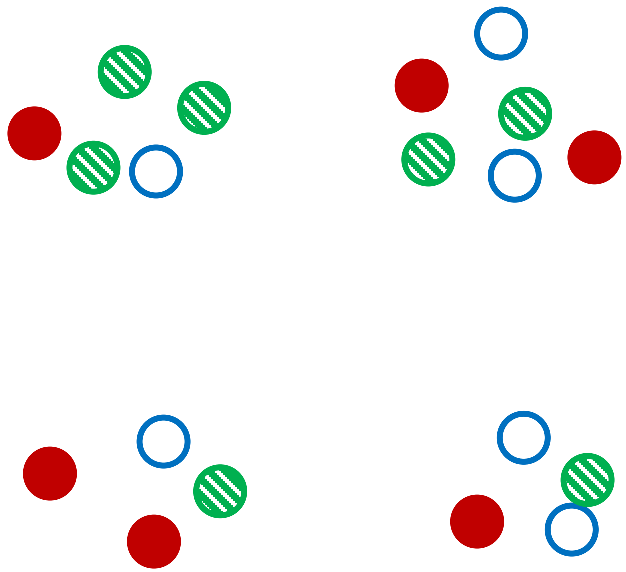
USING FAIRLETS FOR FAIR CLUSTERING

Let: *Y* be our fairlet decomposition, *Y'* be our transformed point set, and *S* be our final clustering.

Theorem: for k-median and k-center: $cost(S) = cost(Y) + cost(Y')$



GENERALIZING TO MORE COLORS



For each color *i*, we get bounds α_i, β_i to bound the proportional representation of *i*.

A cluster *C* is fair if for all colors *i*:

$$\alpha_i \times |C| \leq i(C) \leq \beta_i \times |C|$$

Where $i(C)$ is the number of i colored points in C.

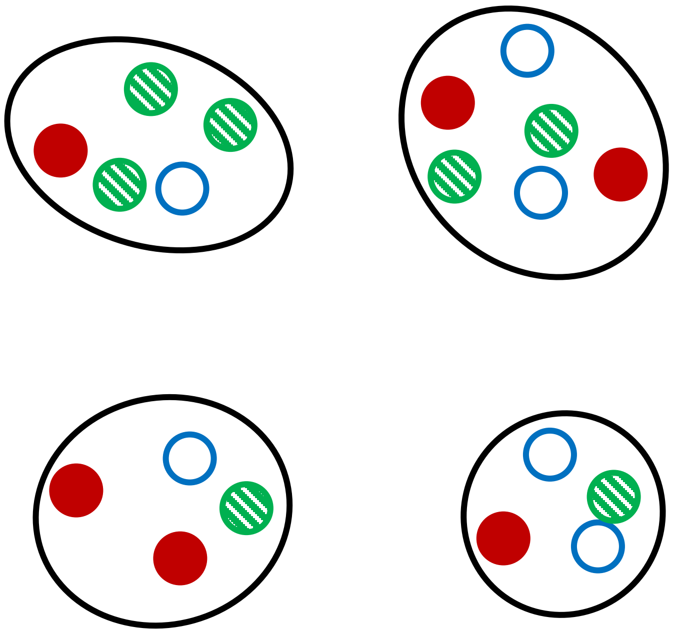
KEY:

= Right-leaning article

= Left-leaning article

= Green party-leaning article

GENERALIZING TO MORE COLORS



KEY:

= Right-leaning article

= Left-leaning article

= Green party-leaning article

Is this clustering fair for the following values?

α_{red}= 1/4 ✗ β_{red} = 1/2 ✓

α_{green}= 1/3 ✗ β_{green} = 1/2 ✗

α_{blue}= 1/4 ✓ β_{blue} = 1/2 ✓

What about…

α_{red}= 1/5 ✓ β_{red} = 1/2 ✓

α_{green}= 1/4 ✓ β_{green} = 3/5 ✓

α_{blue}= 1/4 ✓ β_{blue} = 1/2 ✓

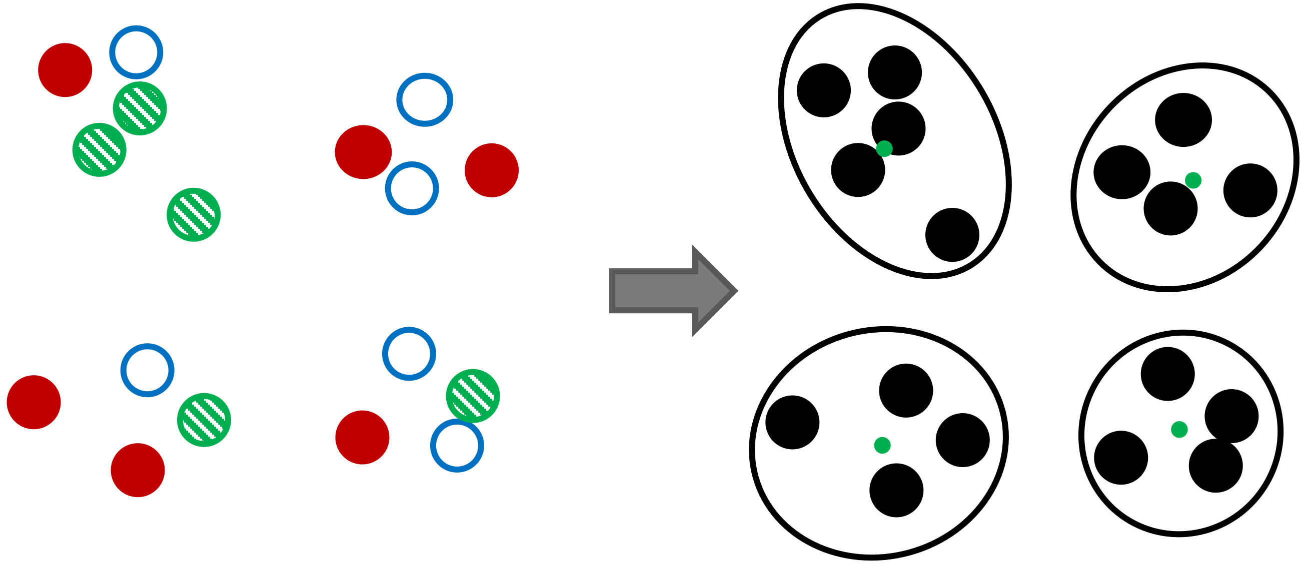
ILP FOR FAIR CLUSTERING

For this approach, we will use a linear program as an intermediate step, *but it will not be used to solve the whole problem*.

1. Ignore fairness. Find a good vanilla clustering. This gives cluster centers.
2. Use a LP relaxation of an ILP to fairly assign points to centers.
3. Round the LP to an integer solution.

ILP FOR FAIR CLUSTERING STEP 1

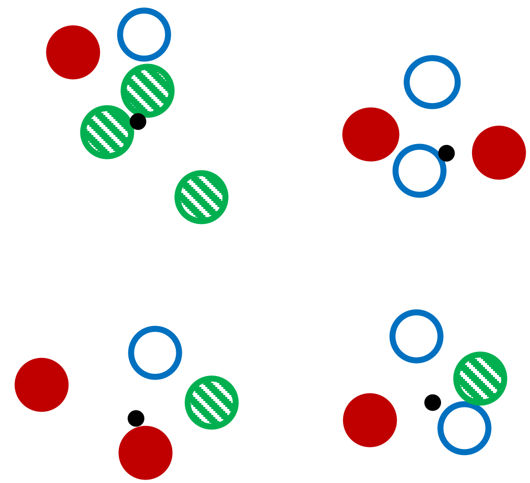
Step 1: Ignore fairness. Find a good vanilla clustering.



ILP FOR FAIR CLUSTERING STEP 2

Step 2: Use a LP relaxation of an ILP to fairly assign points to centers.

Let $C = c_1, c_2, \ldots, c_k$ be our given cluster centers.
Let x_i denote that x is assigned to center i.
Let *R* be our distance guess.



Objective: None!

Constraints: $\sum_{c_i \in C \cap B_R(x)} x_i = 1$ for all $x \in X$

$\alpha_i \sum_{x \in X} x_i \leq \sum_{j \text{ colored } x \in X} x_i \leq \beta_i \sum_{x \in X} x_i$ for all $c_i \in C$ and colors *j*

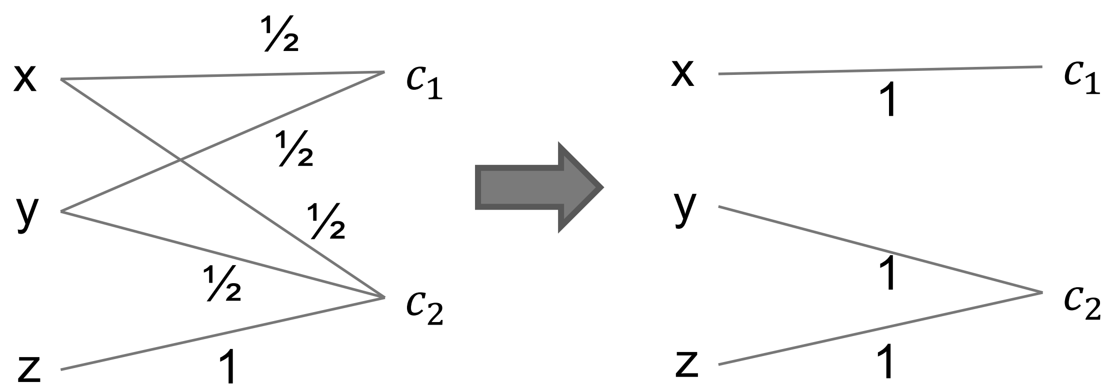
$0 \leq x_i \leq 1$ for all $x \in X$ and $c_i \in C$

ILP FOR FAIR CLUSTERING STEP 3

Now we have a set of cluster centers and a fair *fractional assignment* of points to centers.

In other words, *x* can be ½ assigned to one center and ½ assigned to another!

Step 3: Round these to whole numbers (i.e., a real assignment).



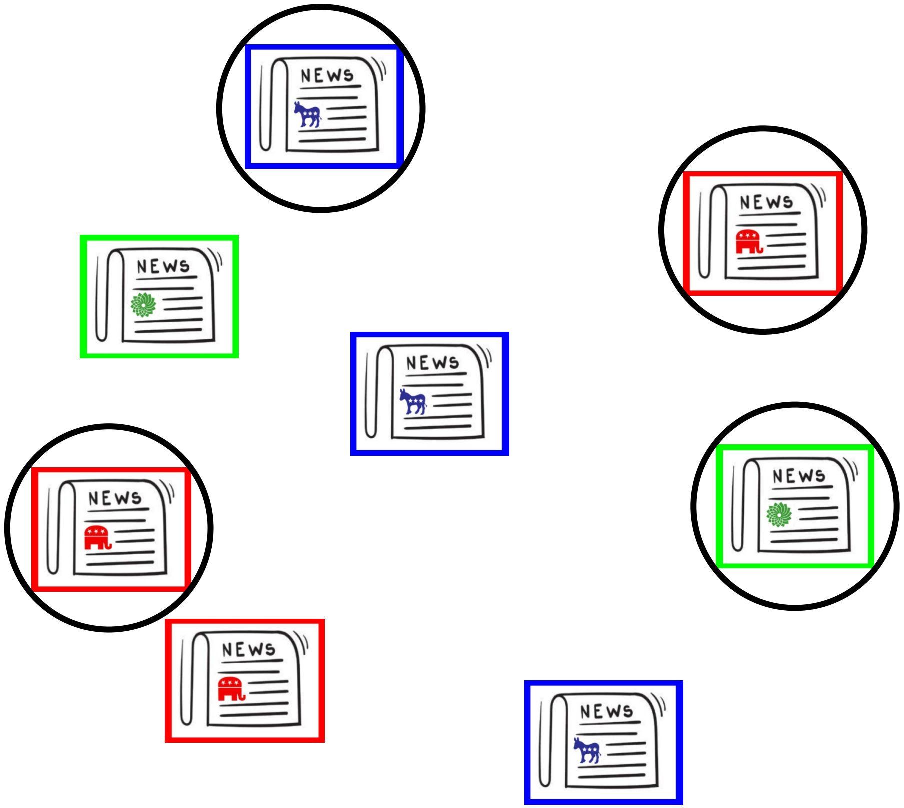
FAIR DATA SUMMARIZATION

Data summarization: Select a subset of points that "represent" your data.

Method: Do a clustering, and select the cluster centers.

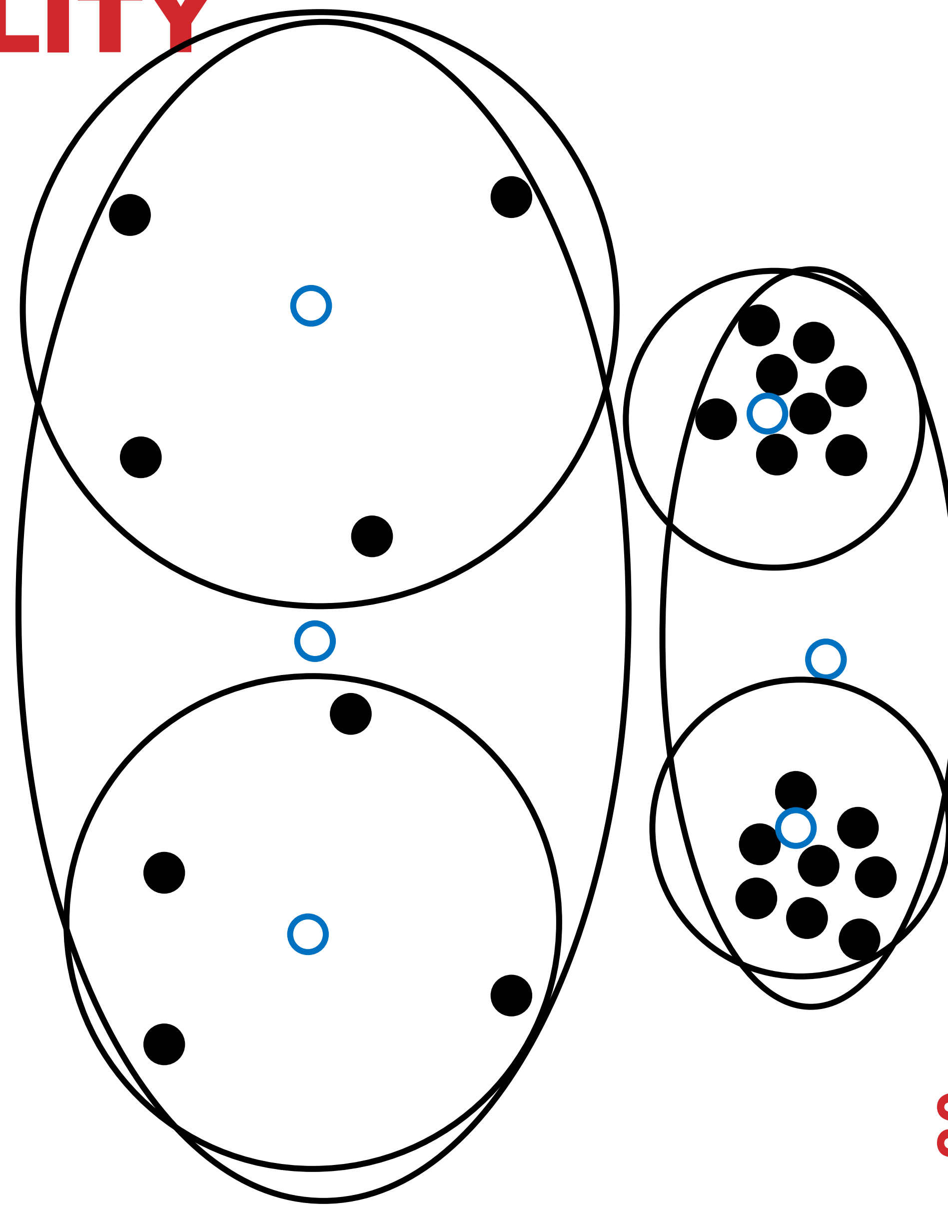
Fairness: for every color *i*, there must be at least k_i representatives of that color.

Say $k_{red} = k_{blue} = k_{\text{green}} = 1$



FAIRNESS THROUGH PROPORTIONALITY

Say you want to build parks 3 parks. You have two dense cities and 1 rural area. There are 6 park location options (blue-rimmed circles).



Blocking coalition: a set of *n/3* people such that there is 1 park location that they *all* prefer to their given location.

Proportional clustering: one with no blocking coalition.