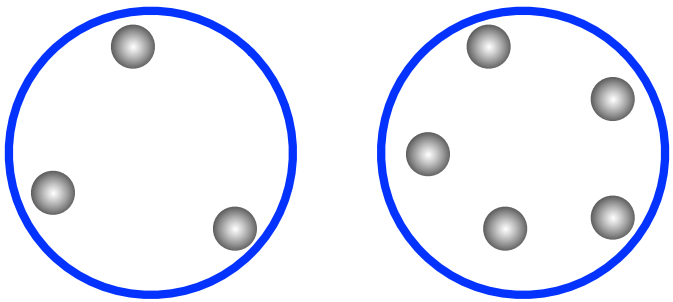
Balanced k-Center Clustering When k Is A Constant

Hu Ding

Computer Science and Engineering, Michigan State University

Ordinary K-Center Clustering

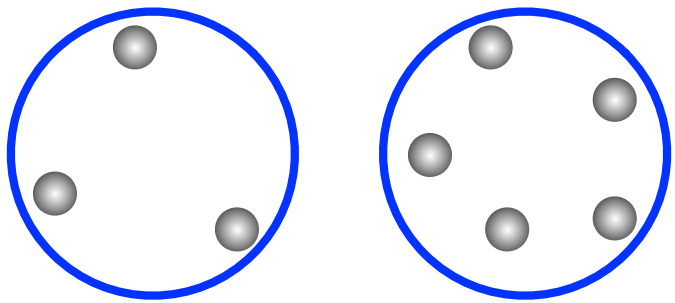


- Given n points in $\mathbb{R}^d$ and an integer $k > 0$, k-center clustering is to find k balls to cover all the points and minimize the maximum radius.
- Many applications in data analysis, networking, etc.

Why Balanced K-Center Clustering?

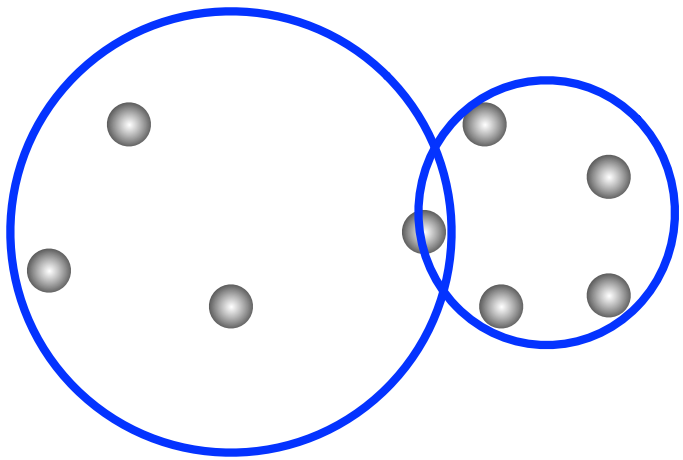
- Add upper and lower bound on cluster size: resource allocation, big data, etc.

Balanced K-Center Clustering



- The size of each resulting cluster should be bounded by the given $[L, U]$.
- $1 \leq L \leq n/k \leq U \leq n$.

Balanced K-Center Clustering



- The size of each resulting cluster should be bounded by the given $[L, U]$.
- $1 \leq L \leq n/k \leq U \leq n$.

Previous Results

- Ordinary k-center clustering: 2-approximation and the hardness for $c < 2$ by Gonzalez 1985, Hochbaum and Shmoys 1985.
- With upper bound (capacitated): Khuller and Sussmann 2000, Cygan et al. 2012, An et al. 2015, etc.
- With lower bound: Aggarwal et al. 2010, Ene et al. 2013, Ahmadian and Swamy 2016, etc.
- With upper and lower bounds: Ding et al. 2017 provide a 6-approximation via linear programming relaxation and rounding techniques.

Our Contribution

- d is high and k is constant: a nearly linear time 4-approximation algorithm.

Our Contribution

- d is high and k is constant: a nearly linear time 4-approximation algorithm.
- Why assume k is constant?

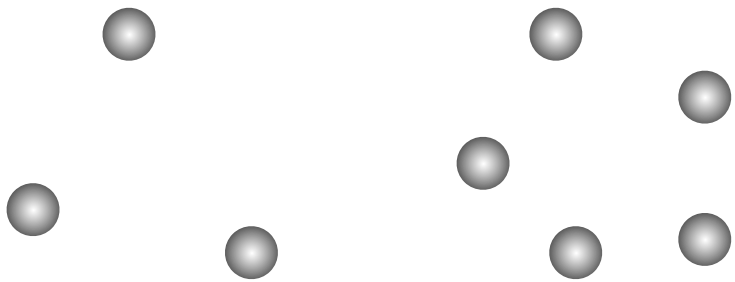
Our Contribution

- d is high and k is constant: a nearly linear time 4-approximation algorithm.
- Why assume k is constant?
  - Any improvement in theory.

Our Contribution

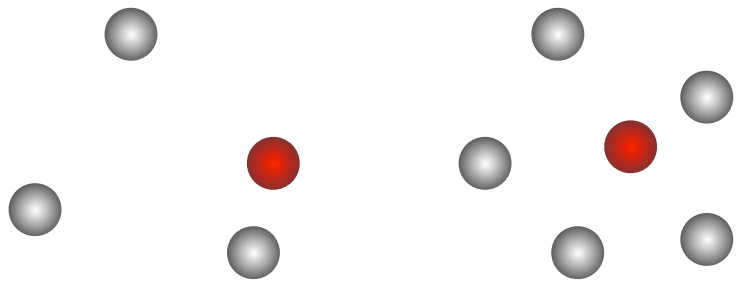
- d is high and k is constant: a nearly linear time 4-approximation algorithm.
- Why assume k is constant?
  - Any improvement in theory.
  - k is often not large in practice.

Overview of Our Approach



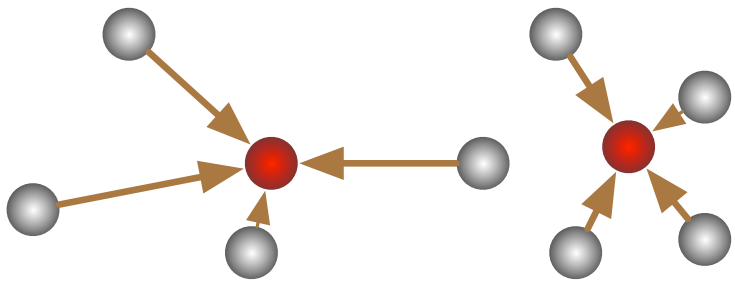
- **Step 1:** find the k cluster centers. Not necessary from the input, though our algorithm finds them from the input.
- **Step 2:** find the balanced assignment.

Overview of Our Approach



- **Step 1:** find the k cluster centers. Not necessary from the input, though our algorithm finds them from the input.
- **Step 2:** find the balanced assignment.

Overview of Our Approach



- **Step 1:** find the k cluster centers. Not necessary from the input, though our algorithm finds them from the input.
- **Step 2:** find the balanced assignment.

Find The K Cluster Centers

Gonzalez's algorithm:

1. Arbitrarily select one point as s_1, and set $S = \{s_1\}$. $j = 2$.
2. Repeat the following steps $k - 1$ times:
   - Let s_j be the point having the largest distance to S.
   - $S = S \cup \{s_j\}$. $j = j + 1$.

Find The K Cluster Centers

Gonzalez's algorithm:

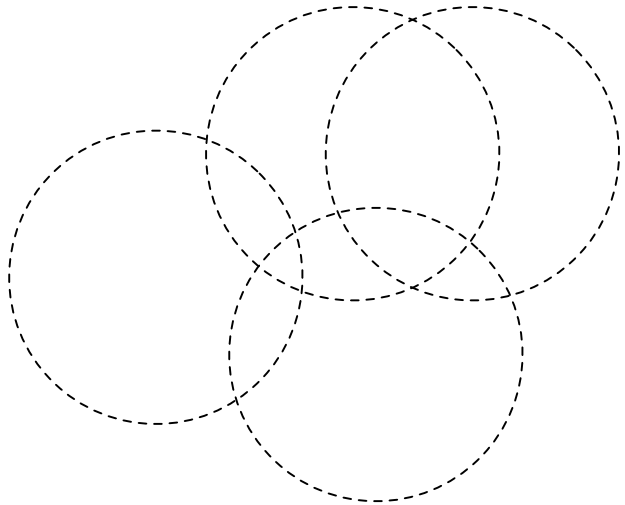
1. Arbitrarily select one point as s_1, and set $S = \{s_1\}$. $j = 2$.
2. Repeat the following steps $k - 1$ times:
   - Let s_j be the point having the largest distance to S.
   - $S = S \cup \{s_j\}$. $j = j + 1$.

- The resulting approximation ratio is 2 for ordinary clustering, but could be arbitrarily large for balanced clustering.

Find The K Cluster Centers

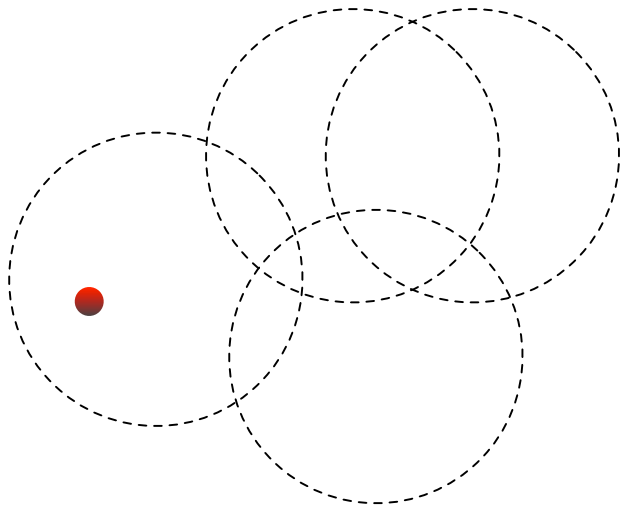
- **Lemma:** replace $S = \{s_1, \cdots, s_k\}$ by the Cartesian product $S^k = S \times \cdots \times S$, and at least one k-tuple from S^k yields 4-approximation.

Find The K Cluster Centers



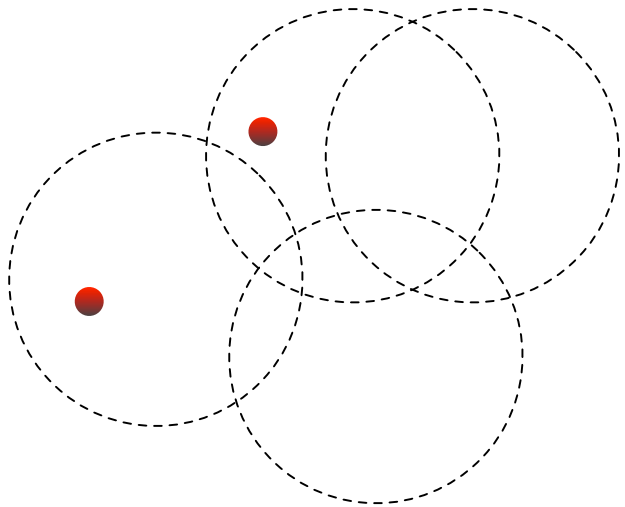
- **Lemma:** replace $S = \{s_1, \cdots, s_k\}$ by the Cartesian product $S^k = S \times \cdots \times S$, and at least one k-tuple from S^k yields 4-approximation.
- Basic idea: imagine the k hidden optimal clusters,

Find The K Cluster Centers



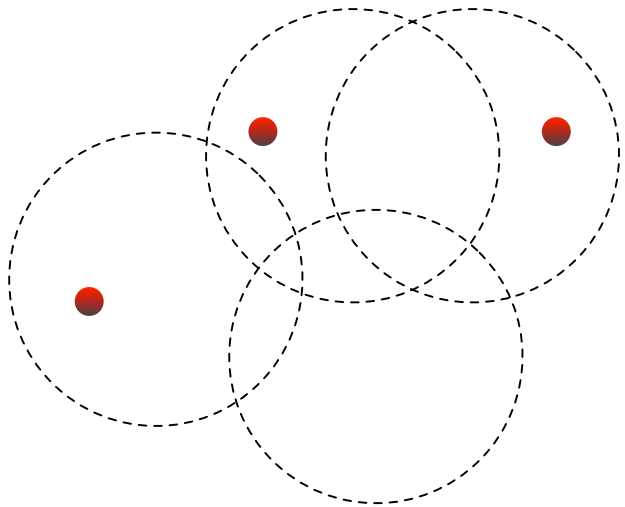
- **Lemma:** replace $S = \{s_1, \cdots, s_k\}$ by the Cartesian product $S^k = S \times \cdots \times S$, and at least one k-tuple from S^k yields 4-approximation.
- Basic idea: imagine the k hidden optimal clusters,
  - if $s_1, \cdots, s_k$ fall into different clusters, S yields 2-approximation;

Find The K Cluster Centers



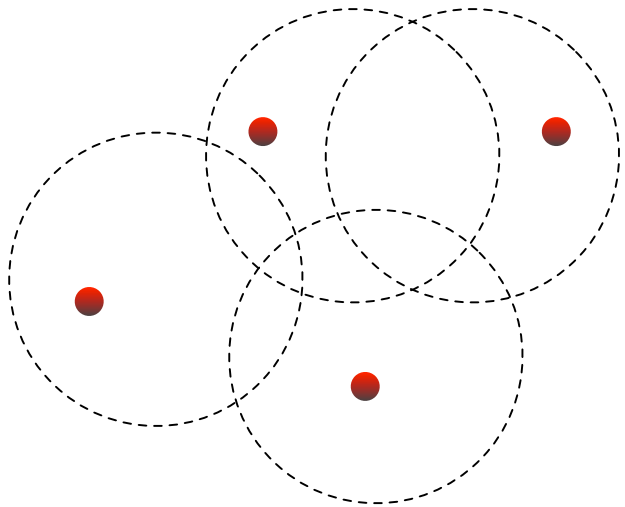
- **Lemma:** replace $S = \{s_1, \cdots, s_k\}$ by the Cartesian product $S^k = S \times \cdots \times S$, and at least one k-tuple from S^k yields 4-approximation.
- Basic idea: imagine the k hidden optimal clusters,
  - if $s_1, \cdots, s_k$ fall into different clusters, S yields 2-approximation;

Find The K Cluster Centers



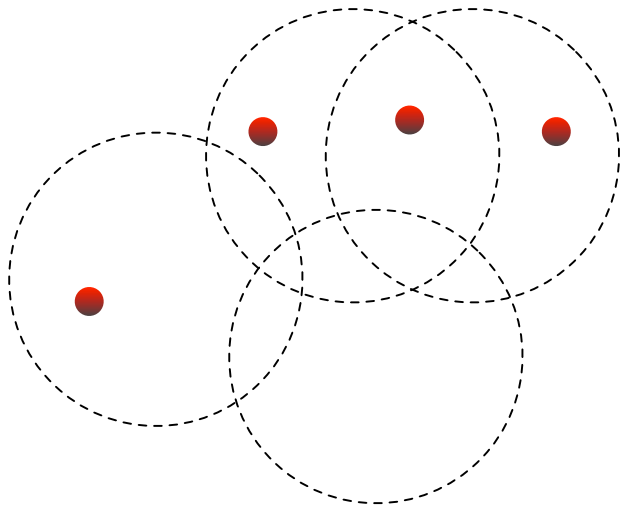
- **Lemma:** replace $S = \{s_1, \cdots, s_k\}$ by the Cartesian product $S^k = S \times \cdots \times S$, and at least one k-tuple from S^k yields 4-approximation.
- Basic idea: imagine the k hidden optimal clusters,
  - if $s_1, \cdots, s_k$ fall into different clusters, S yields 2-approximation;

Find The K Cluster Centers



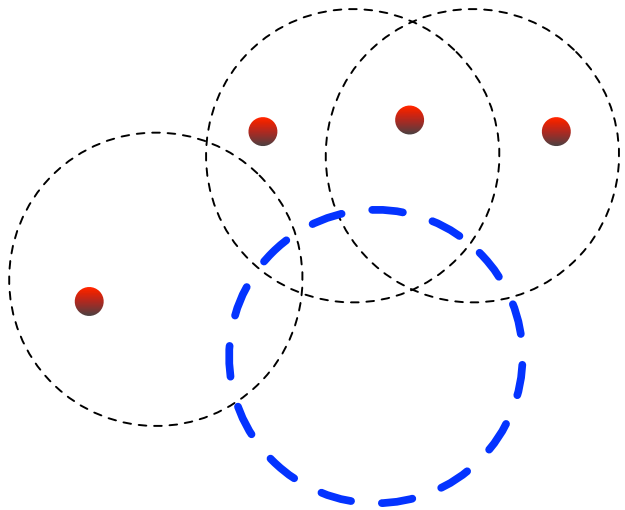
- **Lemma:** replace $S = \{s_1, \cdots, s_k\}$ by the Cartesian product $S^k = S \times \cdots \times S$, and at least one k-tuple from S^k yields 4-approximation.
- Basic idea: imagine the k hidden optimal clusters,
  - if $s_1, \cdots, s_k$ fall into different clusters, S yields 2-approximation;

Find The K Cluster Centers



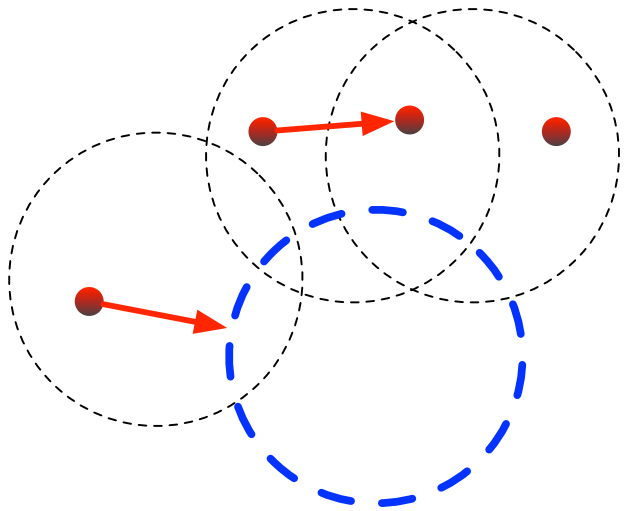
- **Lemma:** replace $S = \{s_1, \cdots, s_k\}$ by the Cartesian product $S^k = S \times \cdots \times S$, and at least one k-tuple from S^k yields 4-approximation.
- Basic idea: imagine the k hidden optimal clusters,
  - if $s_1, \cdots, s_k$ fall into different clusters, S yields 2-approximation;
  - else, s_{j_1} and s_{j_2} fall into the same cluster, but due to the nature of Gonzalez's algorithm, there exists one s_j can handle all the clusters after j_2.

Find The K Cluster Centers



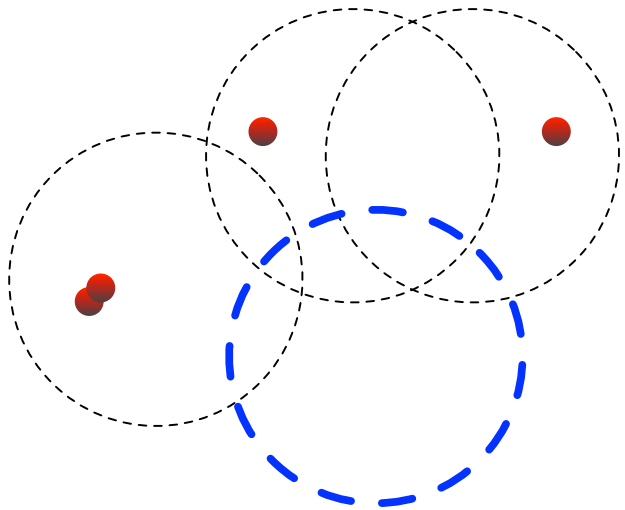
- **Lemma:** replace $S = \{s_1, \cdots, s_k\}$ by the Cartesian product $S^k = S \times \cdots \times S$, and at least one k-tuple from S^k yields 4-approximation.
- Basic idea: imagine the k hidden optimal clusters,
  - if $s_1, \cdots, s_k$ fall into different clusters, S yields 2-approximation;
  - else, s_{j_1} and s_{j_2} fall into the same cluster, but due to the nature of Gonzalez's algorithm, there exists one s_j can handle all the clusters after j_2.

Find The K Cluster Centers



- **Lemma:** replace $S = \{s_1, \cdots, s_k\}$ by the Cartesian product $S^k = S \times \cdots \times S$, and at least one k-tuple from S^k yields 4-approximation.
- Basic idea: imagine the k hidden optimal clusters,
  - if $s_1, \cdots, s_k$ fall into different clusters, S yields 2-approximation;
  - else, s_{j_1} and s_{j_2} fall into the same cluster, but due to the nature of Gonzalez's algorithm, there exists one s_j can handle all the clusters after j_2.

Find The K Cluster Centers

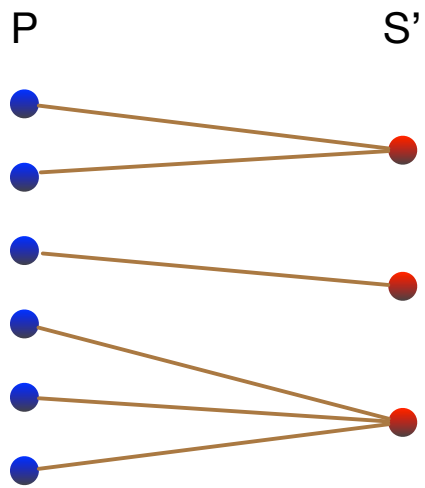


- **Lemma:** replace $S = \{s_1, \cdots, s_k\}$ by the Cartesian product $S^k = S \times \cdots \times S$, and at least one k-tuple from S^k yields 4-approximation.
- Basic idea: imagine the k hidden optimal clusters,
  - if $s_1, \cdots, s_k$ fall into different clusters, S yields 2-approximation;
  - else, s_{j_1} and s_{j_2} fall into the same cluster, but due to the nature of Gonzalez's algorithm, there exists one s_j can handle all the clusters after j_2.

Sketch of our algorithm

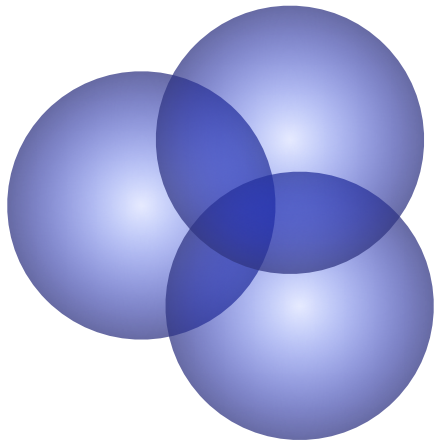
1. Run Gonzalez's algorithm to obtain $S = \{s_1, \cdots, s_k\}$.
2. Let $\mathcal{R}$ be the set of sorted nk distances from the input P to S.
3. Fix each candidate S' from S^k, binary search on $r \in \mathcal{R}$ to **check whether a balanced assignment exists for** (S', r).
4. Output the candidate with the smallest feasible r.

Find The Balanced Assignment



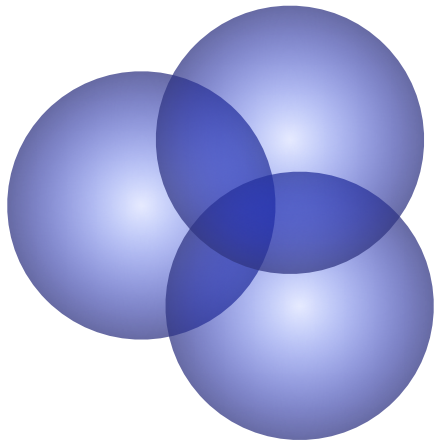
- Fix (S', r), build the bipartite graph and find the balanced assignment via max flow but with at least $O(VE) = O(n^2)$ time.

Find The Balanced Assignment: Another Idea



- When we fix (S', r), the n points are divided into at most $2^k - 1$ parts.

Find The Balanced Assignment: Another Idea



- When we fix (S', r), the n points are divided into at most $2^k - 1$ parts.
- Each part is covered by $1 \leq t \leq k$ balls.

Find The Balanced Assignment: Another Idea

$$x^{j_1}_{(j_1,j_2,\cdots,j_t)} + \cdots + x^{j_t}_{(j_1,j_2,\cdots,j_t)} = n_{(j_1,j_2,\cdots,j_t)},$$

$$L \leq \sum_{(j_1,j_2,\cdots,j_t)\in\pi_{j_l}} x^{j_l}_{(j_1,j_2,\cdots,j_t)} \leq U.$$

- Solve the system of linear equations and inequalities (SoL) with the complexity $O(k2^k)$.
  - $n_{(j_1,j_2,\cdots,j_t)}$: the number of points covered by the balls $j_1,\cdots,j_t$.
  - $x^{j_l}_{(j_1,j_2,\cdots,j_t)}$: the number of points assigned to j_l-th cluster.
  - π_{j_l}: the set of all the subsets containing j_l of $\{1,2,\cdots,k\}$.

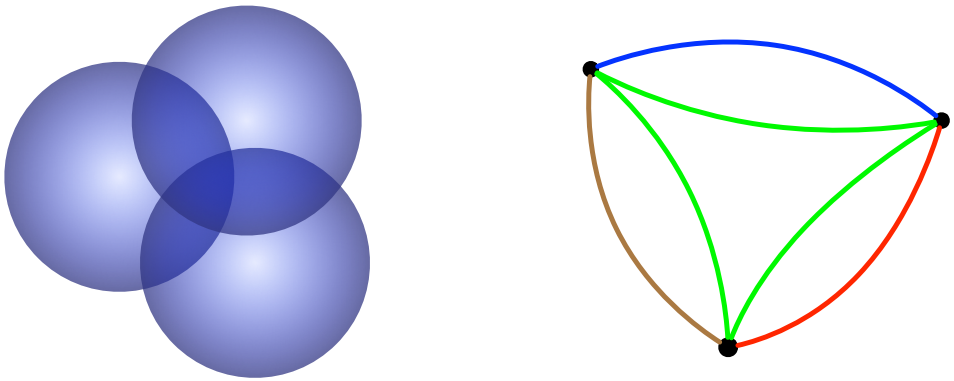
Find The Balanced Assignment: Another Idea

$$x^{j_1}_{(j_1,j_2,\cdots,j_t)} + \cdots + x^{j_t}_{(j_1,j_2,\cdots,j_t)} = n_{(j_1,j_2,\cdots,j_t)},$$

$$L \leq \sum_{(j_1,j_2,\cdots,j_t)\in\pi_{j_l}} x^{j_l}_{(j_1,j_2,\cdots,j_t)} \leq U.$$

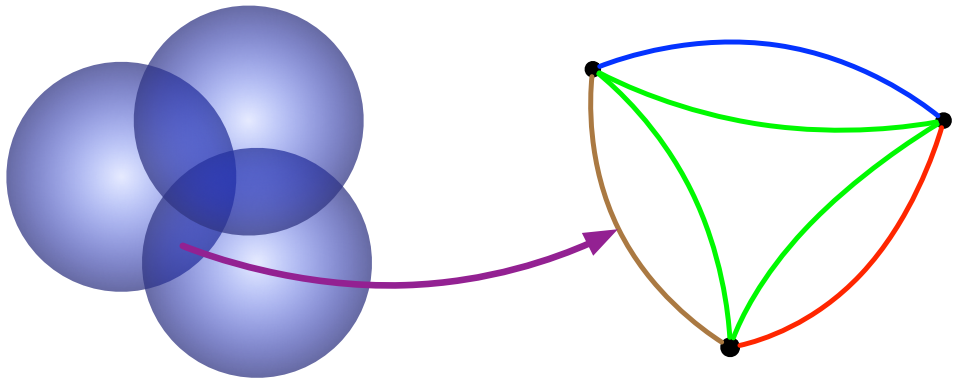
- Solve the system of linear equations and inequalities (SoL) with the complexity $O(k2^k)$.
  - $n_{(j_1,j_2,\cdots,j_t)}$: the number of points covered by the balls $j_1,\cdots,j_t$.
  - $x^{j_l}_{(j_1,j_2,\cdots,j_t)}$: the number of points assigned to j_l-th cluster.
  - π_{j_l}: the set of all the subsets containing j_l of $\{1,2,\cdots,k\}$.
- Can be solved in $O(poly(2^k))$ time, but how to transform a solution to integer?

Rounding for SOL



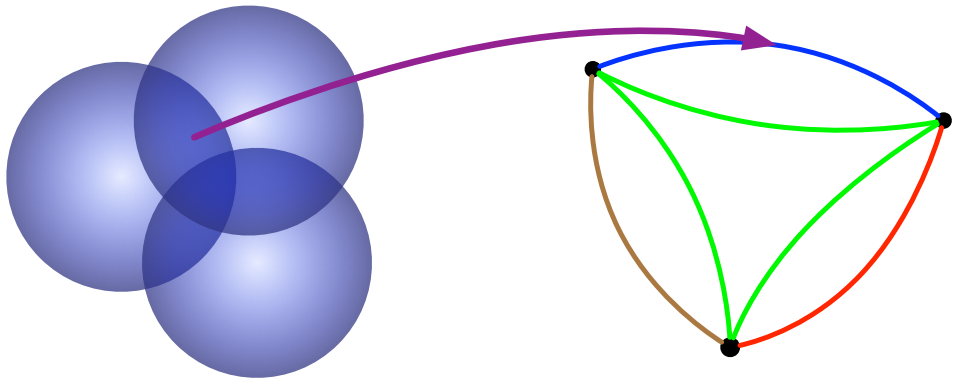
- Build a colored multigraph G:
  - each ball corresponds to one vertex;
  - each intersection of ball $j_1, \cdots, j_t$ corresponds to $\binom{t}{2}$ edges with the same color;
  - each couple of variables $(x^{j_l}_{(j_1,\cdots,j_t)}, x^{j_{l'}}_{(j_1,\cdots,j_t)})$ corresponds to a unique edge.

Rounding for SOL



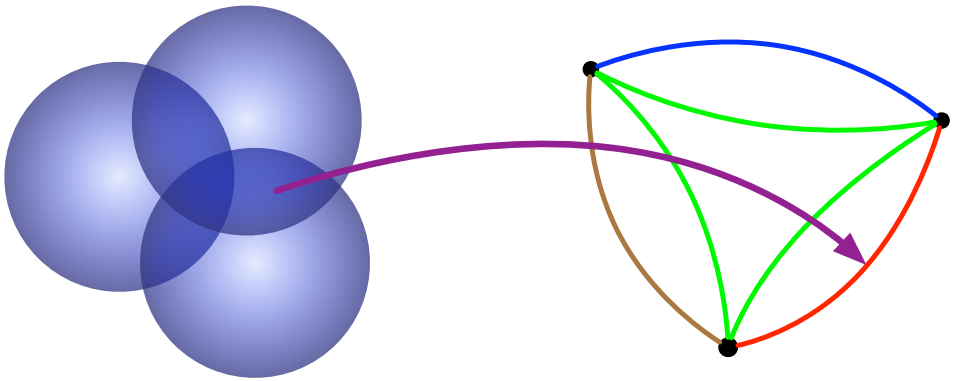
- Build a colored multigraph G:
  - each ball corresponds to one vertex;
  - each intersection of ball $j_1, \cdots, j_t$ corresponds to $\binom{t}{2}$ edges with the same color;
  - each couple of variables $(x^{j_l}_{(j_1,\cdots,j_t)}, x^{j_{l'}}_{(j_1,\cdots,j_t)})$ corresponds to a unique edge.

Rounding for SOL



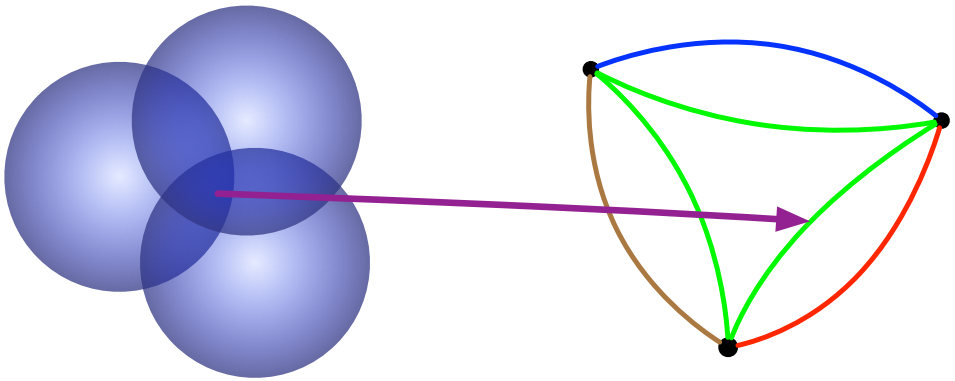
- Build a colored multigraph G:
  - each ball corresponds to one vertex;
  - each intersection of ball $j_1, \cdots, j_t$ corresponds to $\binom{t}{2}$ edges with the same color;
  - each couple of variables $(x^{j_l}_{(j_1,\cdots,j_t)}, x^{j_{l'}}_{(j_1,\cdots,j_t)})$ corresponds to a unique edge.

Rounding for SOL



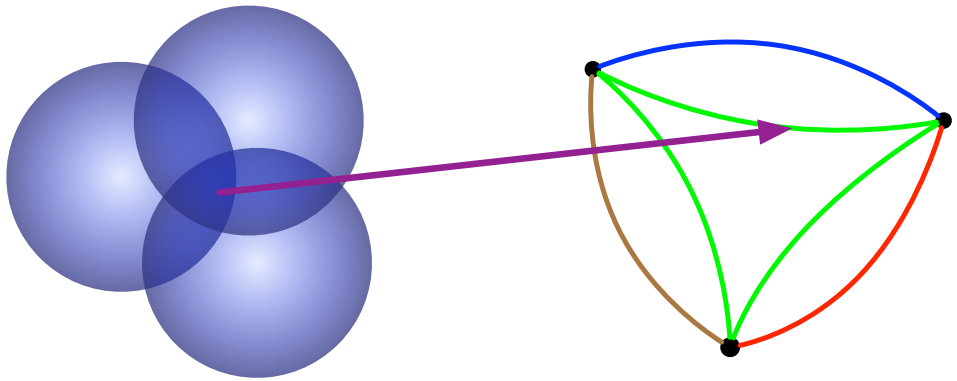
- Build a colored multigraph G:
  - each ball corresponds to one vertex;
  - each intersection of ball $j_1, \cdots, j_t$ corresponds to $\binom{t}{2}$ edges with the same color;
  - each couple of variables $(x^{j_l}_{(j_1,\cdots,j_t)}, x^{j_{l'}}_{(j_1,\cdots,j_t)})$ corresponds to a unique edge.

Rounding for SOL



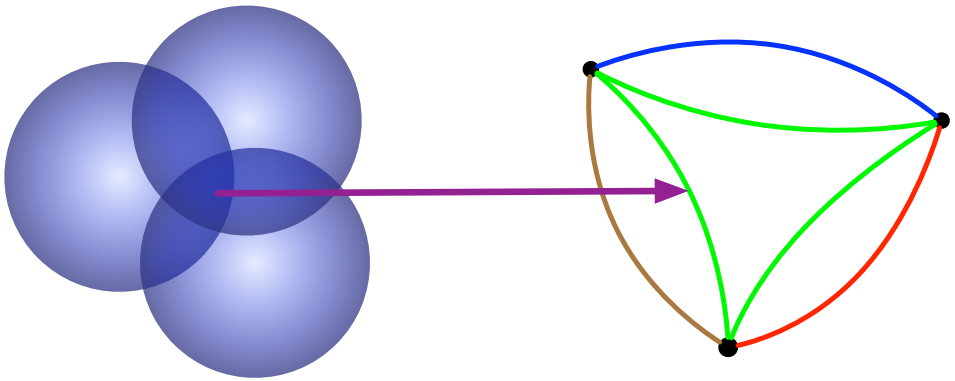
- Build a colored multigraph G:
  - each ball corresponds to one vertex;
  - each intersection of ball $j_1, \cdots, j_t$ corresponds to $\binom{t}{2}$ edges with the same color;
  - each couple of variables $(x^{j_l}_{(j_1,\cdots,j_t)}, x^{j_{l'}}_{(j_1,\cdots,j_t)})$ corresponds to a unique edge.

Rounding for SOL



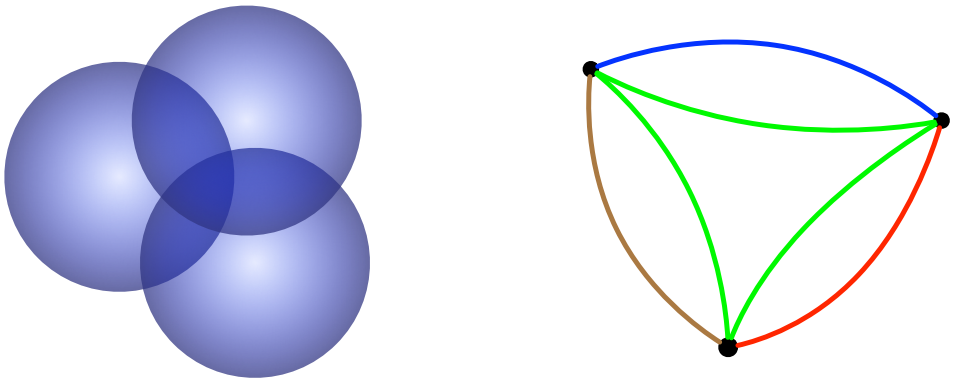
- Build a colored multigraph G:
  - each ball corresponds to one vertex;
  - each intersection of ball $j_1, \cdots, j_t$ corresponds to $\binom{t}{2}$ edges with the same color;
  - each couple of variables $(x^{j_l}_{(j_1,\cdots,j_t)}, x^{j_{l'}}_{(j_1,\cdots,j_t)})$ corresponds to a unique edge.

Rounding for SOL



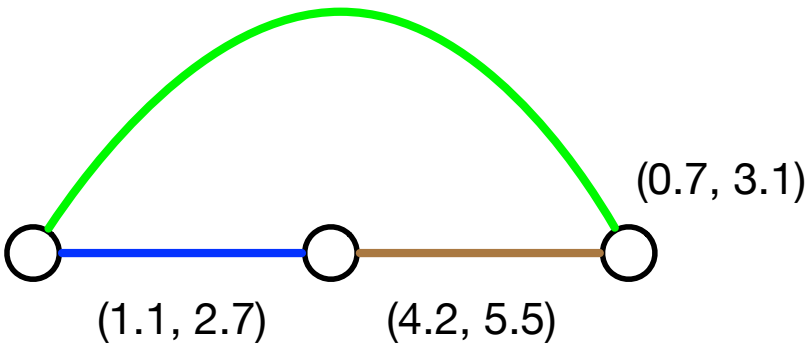
- Build a colored multigraph G:
  - each ball corresponds to one vertex;
  - each intersection of ball $j_1, \cdots, j_t$ corresponds to $\binom{t}{2}$ edges with the same color;
  - each couple of variables $(x^{j_l}_{(j_1,\cdots,j_t)}, x^{j_{l'}}_{(j_1,\cdots,j_t)})$ corresponds to a unique edge.

Rounding for SOL



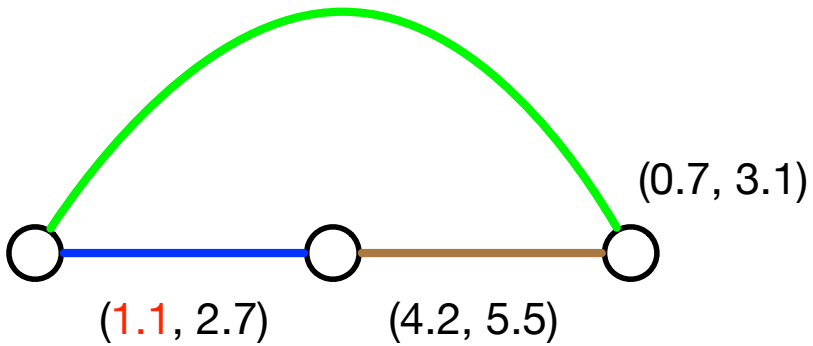
- Build a colored multigraph G:
  - each ball corresponds to one vertex;
  - each intersection of ball $j_1, \cdots, j_t$ corresponds to $\binom{t}{2}$ edges with the same color;
  - each couple of variables $(x^{j_l}_{(j_1,\cdots,j_t)}, x^{j_{l'}}_{(j_1,\cdots,j_t)})$ corresponds to a unique edge.

Rounding for SOL



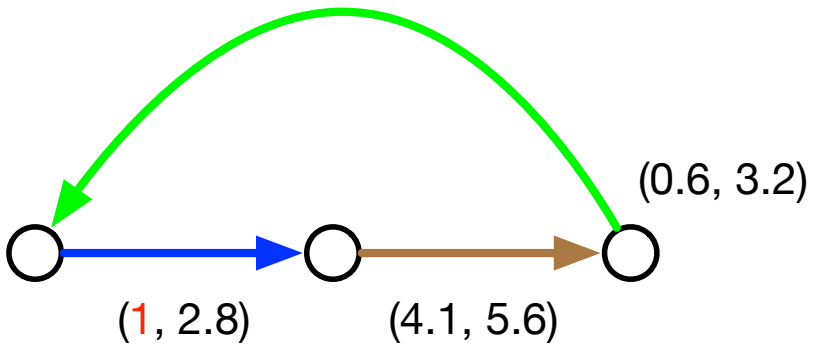
- Case 1: there exists a circle with at least two colors.

Rounding for SOL



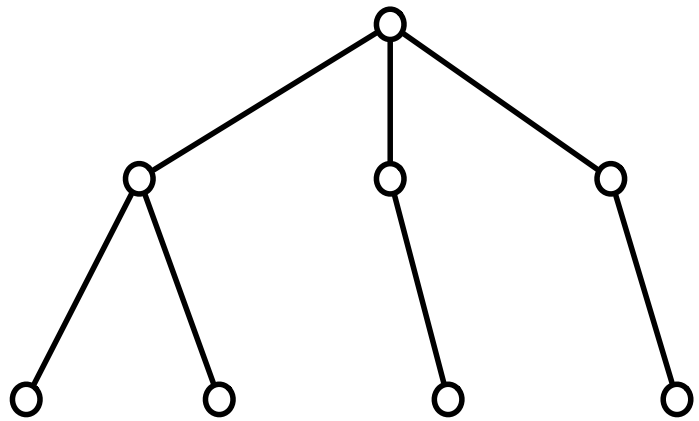
- Case 1: there exists a circle with at least two colors.

Rounding for SOL



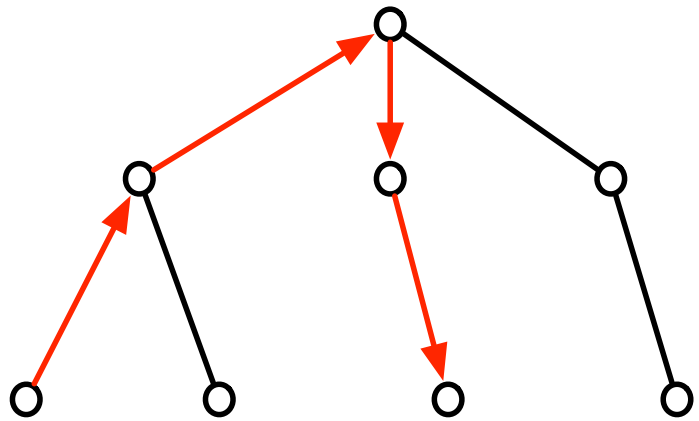
- Case 1: there exists a circle with at least two colors.

Rounding for SOL



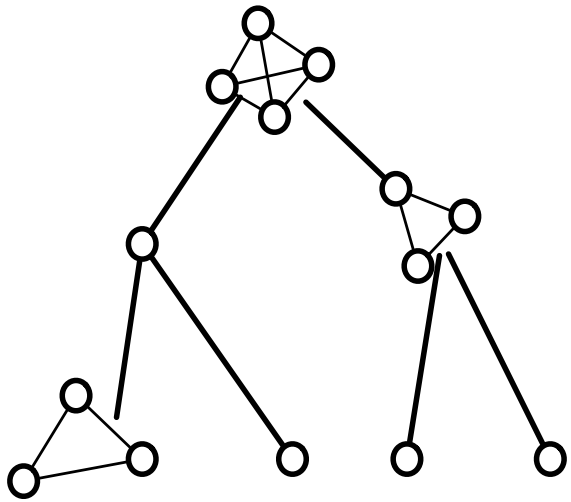
- Case 1: there exists a circle with at least two colors.
- Case 2: no circle.

Rounding for SOL



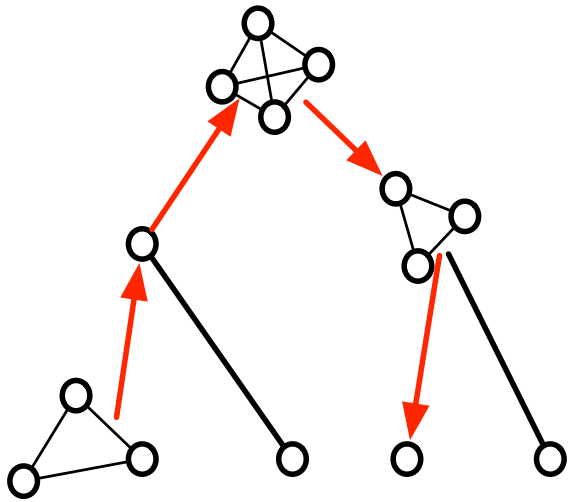
- Case 1: there exists a circle with at least two colors.
- Case 2: no circle.

Rounding for SOL



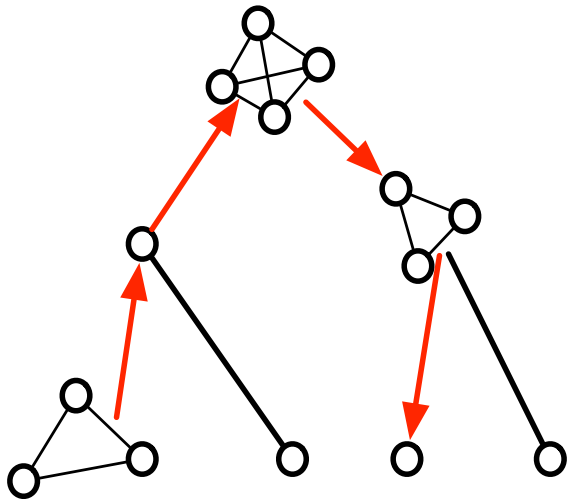
- Case 1: there exists a circle with at least two colors.
- Case 2: no circle.
- Case 3: each circle has only one color, build a pseudo tree.

Rounding for SOL



- Case 1: there exists a circle with at least two colors.
- Case 2: no circle.
- Case 3: each circle has only one color, build a pseudo tree.

Rounding for SOL



- Case 1: there exists a circle with at least two colors.
- Case 2: no circle.
- Case 3: each circle has only one color, build a pseudo tree.
- **Lemma:** Any solution of the SoL can always be rounded to an integer solution in $O(poly(2^k))$ time.

The Final Results

- **Theorem:** Our algorithm yields 4-approximation and costs $O(n(\log n + d))$ time when k is constant.
  - The time complexity is dominated by Gonzalez's algorithm, computing and sorting the nk distances, and building the SoL $\log n$ times.
  - We have the example to show that 4 is tight.
- **Corollary:** Our algorithm can be extended for any metric space, and costs $O(n(\log n + D))$ time where $O(D)$ is the running time for acquiring pairwise distance.

Thank You!

Any Question?