CS240 Algorithm Design and Analysis

Lecture 27

Approximation Algorithms

Quan Li
Fall 2023
2024.01.12

Generalized Load Balancing

Generalized Load Balancing

Input. Set of m machines M; set of n jobs J.

- Job j must run contiguously on an authorized machine in $M_j \subseteq M$.
- Job j has processing time t_j.
- Each machine can process at most one job at a time.

Def. Let J(i) be the subset of jobs assigned to machine i. The load of machine i is $L_i = \Sigma_{j \in J(i)} t_j$.

Def. The makespan is the maximum load on any machine = $\max_i L_i$.

Generalized load balancing. Assign each job to an authorized machine to minimize makespan.

Generalized Load Balancing: Integer Linear Program and Relaxation

ILP formulation. x_{ij} = time machine i spends processing job j.

$$
\begin{array}{llllll}
(IP) & \min & L & & & \\
& \text{s.t.} & \sum_i x_{ij} & = & t_j & \text{for all } j \in J \\
& & \sum_j x_{ij} & \leq & L & \text{for all } i \in M \\
& & x_{ij} & \in & \{0, t_j\} & \text{for all } j \in J \text{ and } i \in M_j \\
& & x_{ij} & = & 0 & \text{for all } j \in J \text{ and } i \notin M_j
\end{array}
$$

LP relaxation.

$$
\begin{array}{llllll}
(LP) & \min & L & & & \\
& \text{s.t.} & \sum_i x_{ij} & = & t_j & \text{for all } j \in J \\
& & \sum_j x_{ij} & \leq & L & \text{for all } i \in M \\
& & x_{ij} & \geq & 0 & \text{for all } j \in J \text{ and } i \in M_j \\
& & x_{ij} & = & 0 & \text{for all } j \in J \text{ and } i \notin M_j
\end{array}
$$

Generalized Load Balancing: Lower Bounds

Lemma 1. The optimal makespan $L^* \geq \max_j t_j$.

Pf. Some machine must process the most time-consuming job. ■

Lemma 2. Let L be the optimal value to the LP. Then, the optimal makespan $L^* \geq L$.

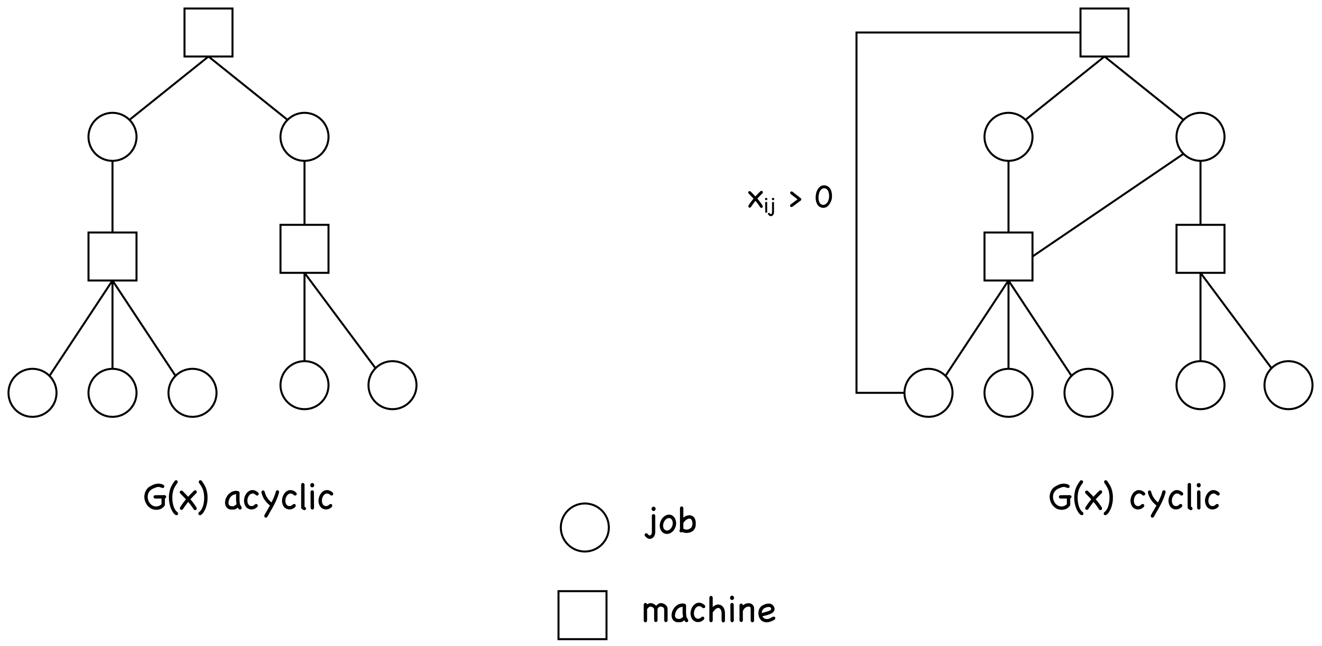
Pf. LP has fewer constraints than IP formulation. ■

Generalized Load Balancing: structure of LP solution

Lemma 3. Let x be solution to LP. Let G(x) be the graph with an edge from machine i to job j if $x_{ij} > 0$. Then G(x) is acyclic.

can transform x into another LP solution where G(x) is acyclic if LP solver doesn't return such an x

Pf. (deferred)



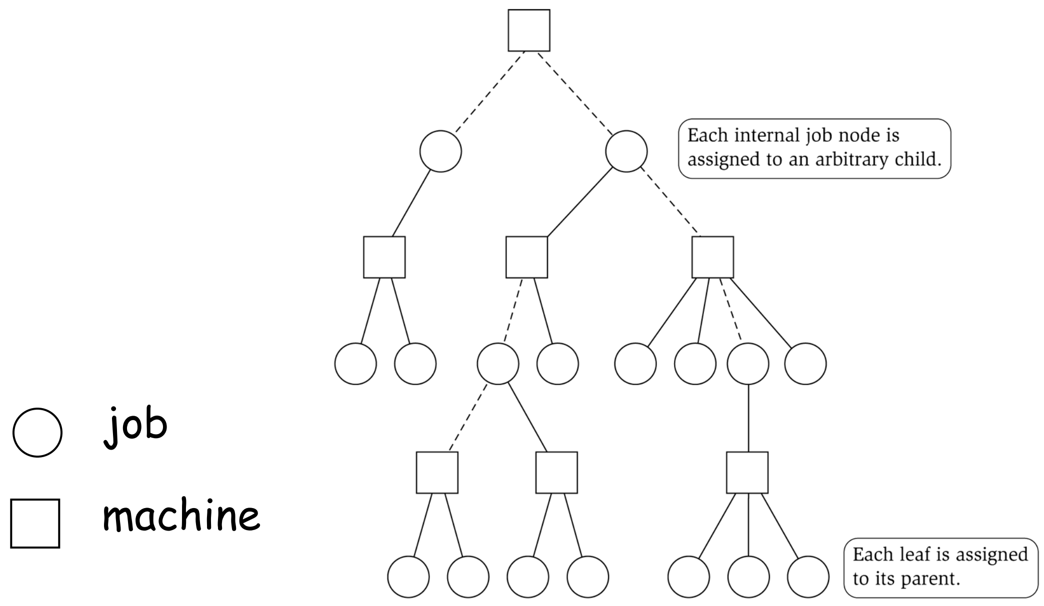
Generalized Load Balancing: Rounding

Rounded solution. Find LP solution x where G(x) is a forest. Root forest G(x) at some arbitrary machine node r.

- If job j is a leaf node, assign j to its parent machine i.
- If job j is not a leaf node, assign j to one of its children.

Lemma 4. Rounded solution only assigns jobs to authorized machines.

Pf. If job j is assigned to machine i, then $x_{ij} > 0$. LP solution can only assign positive value to authorized machines. ▪



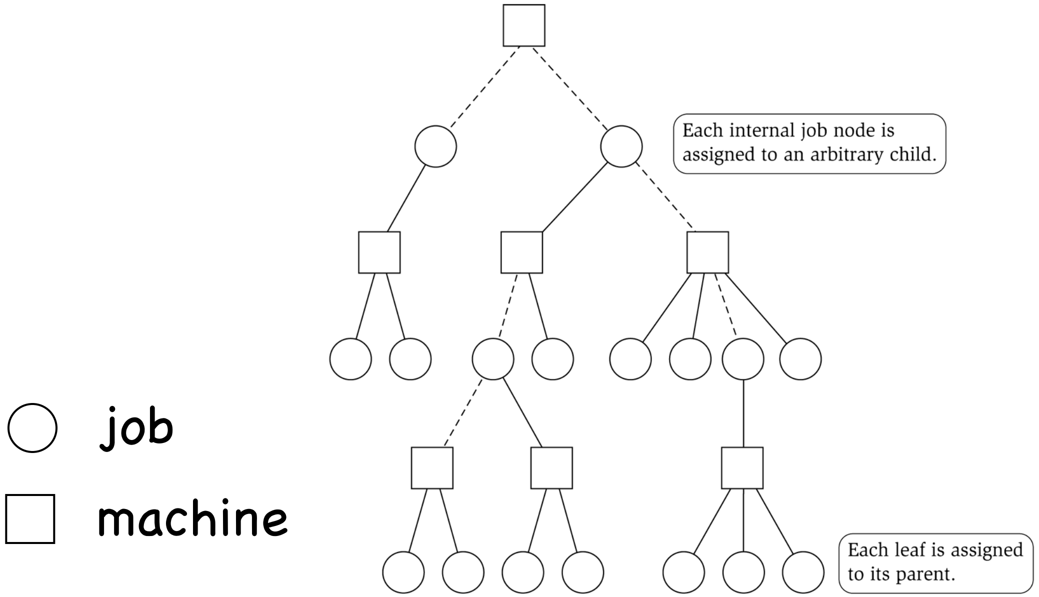
Generalized Load Balancing: Analysis

Lemma 5. If job j is a leaf node and machine i = parent(j), then $x_{ij} = t_j$.

Pf. Since j is a leaf, $x_{ij} = 0$ for all $i \neq$ parent(j). LP constraint guarantees $\Sigma_i \, x_{ij} = t_j$. ▪

Lemma 6. At most one non-leaf job is assigned to a machine.

Pf. The only possible non-leaf job assigned to machine i is parent(i). ▪



Generalized Load Balancing: Analysis

Theorem. Rounded solution is a 2-approximation.

Pf.

- Let J(i) be the jobs assigned to machine i.
- By Lemma 6, the load L_i on machine i has two components:

  - leaf nodes

$$\sum_{\substack{j \in J(i) \\ j \text{ is a leaf}}} t_j \overset{\text{Lemma 5}}{=} \sum_{\substack{j \in J(i) \\ j \text{ is a leaf}}} x_{ij} \le \sum_{j \in J} x_{ij} \overset{\text{LP}}{\le} \underbrace{L}_{\text{optimal value of LP}} \overset{\text{Lemma 2 (LP is a relaxation)}}{\le} L^*$$

  - parent(i)

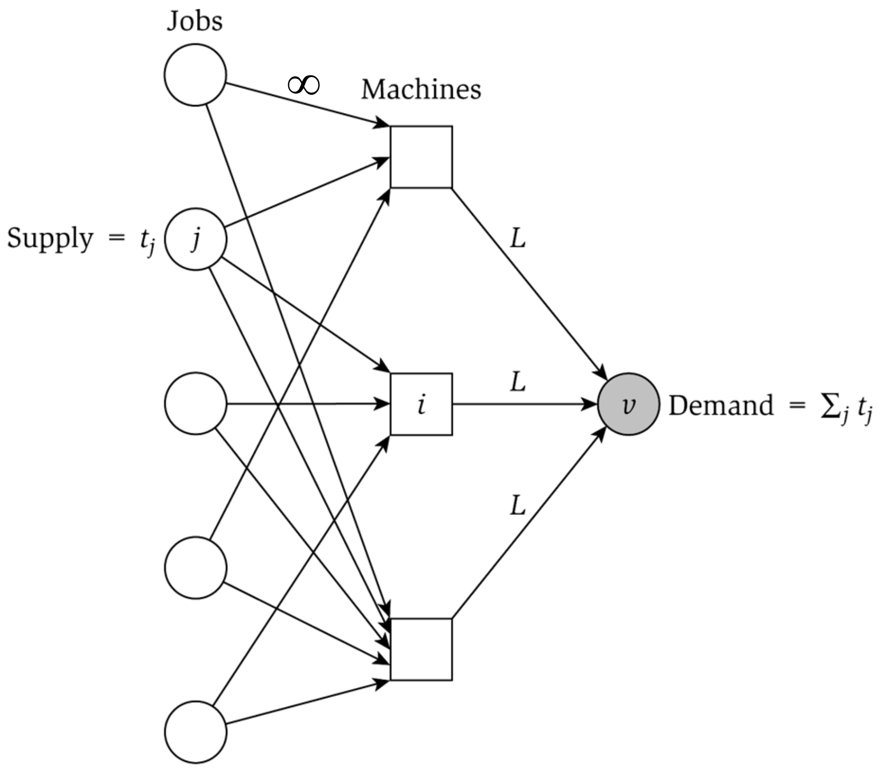
$$t_{\text{parent}(i)} \overset{\text{Lemma 1}}{\le} L^*$$

- Thus, the overall load $L_i \le 2L^*$. ■

Generalized Load Balancing: Flow Formulation

Flow formulation of LP.

$$
\begin{array}{lcll}
\sum_i x_{ij} & = & t_j & \text{for all } j \in J \\
\sum_j x_{ij} & \le & L & \text{for all } i \in M \\
x_{ij} & \ge & 0 & \text{for all } j \in J \text{ and } i \in M_j \\
x_{ij} & = & 0 & \text{for all } j \in J \text{ and } i \notin M_j
\end{array}
$$



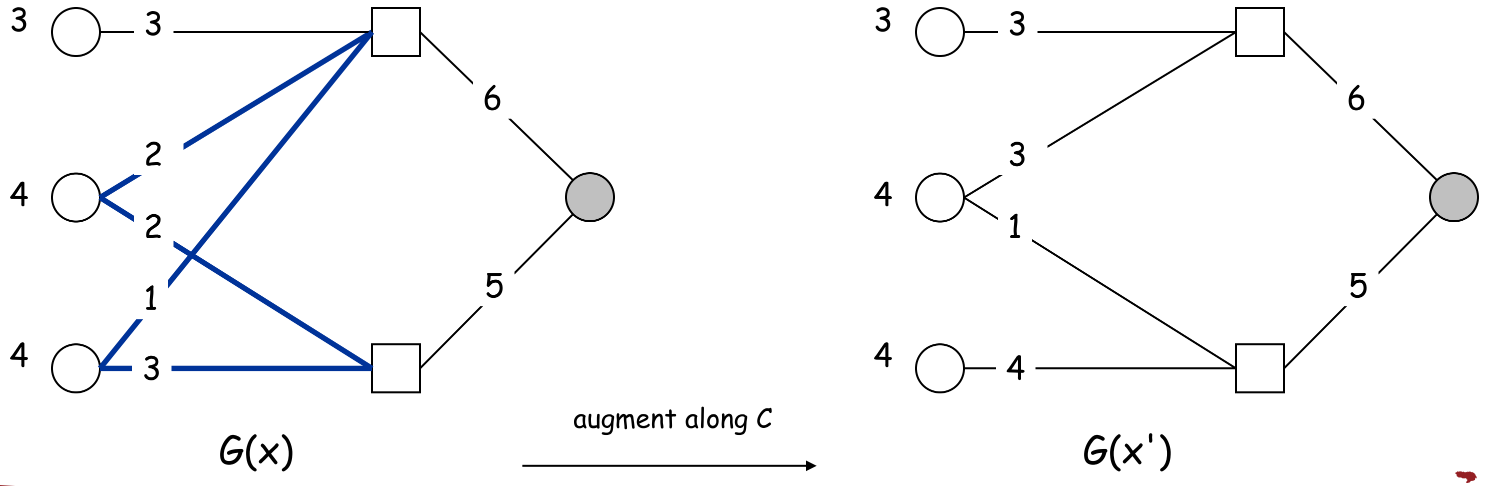
Observation. Solution to feasible flow problem with value L are in one-to-one correspondence with LP solutions of value L.

Generalized Load Balancing: Structure of Solution

Lemma 3. Let (x, L) be solution to LP. Let G(x) be the graph with an edge from machine i to job j if $x_{ij} > 0$. We can find another solution (x', L) such that G(x') is acyclic.

Pf. Let C be a cycle in G(x).

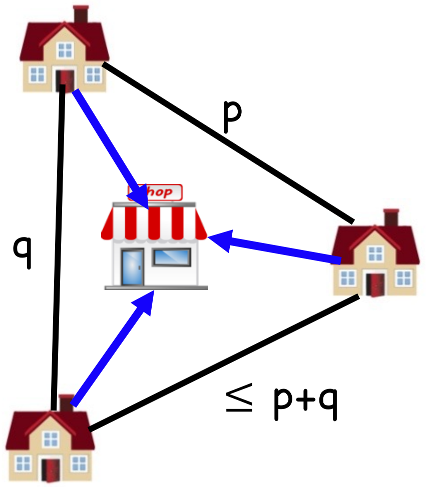
- Augment flow along the cycle C. ← flow conservation maintained
- At least one edge from C is removed (and none are added).
- Repeat until G(x') is acyclic.

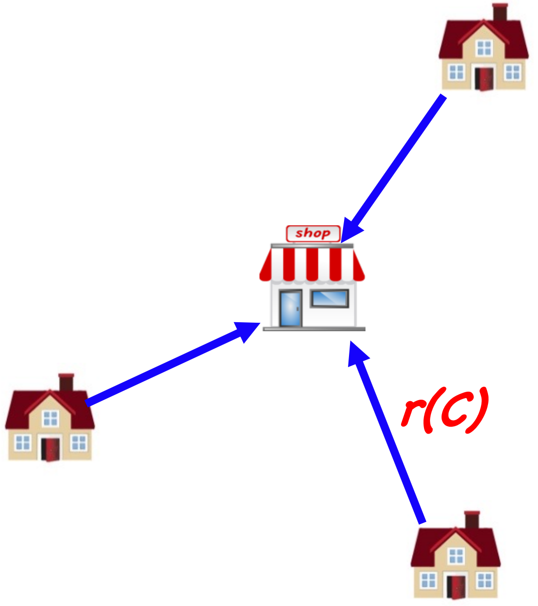


K-Center Problem

K-Center Problem

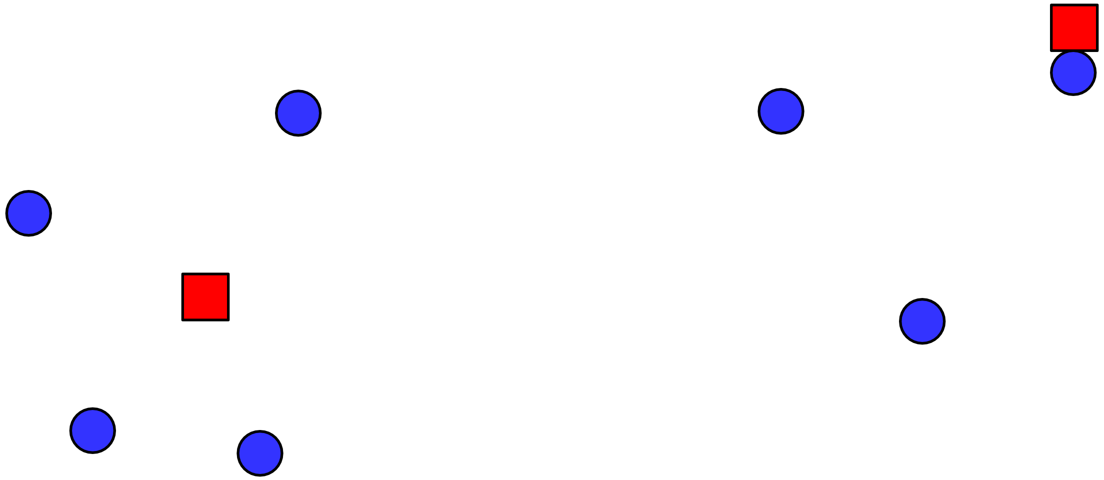
- Given a city with n sites, we want to build k centers to serve them.
  - Let S be set of sites, C be set of centers.
- Each site uses the center closest to it.
  - Distance of site s from the nearest center is $d(s, C) = min_{c \in C} d(s, c)$.
- Goal is to make sure no site is too far from its center.
  - We want to minimize the max distance that any site is from its closest center.
    - Minimize $r(C) = max_{s \in S} min_{c \in C} d(s, c)$.
  - C is called a cover of S, and r is called C's radius.
  - Where should we put centers to minimize the radius?
- Assume distances satisfy triangle inequality.





Gonzalez's Algorithm

- k-Center is NP-complete.
- We'll give a simple 2-approximation for it.
- Idea Say there's one site that's farthest away from all centers. Then it makes the radius large. We'll put a center at that site, to reduce the radius.
  - Note we allow putting center at same location as site.



Gonzalez's Algorithm

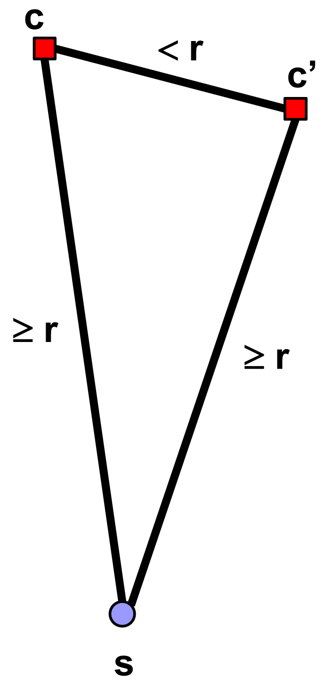
- C is set of centers, initially empty.

- repeat k times
  - choose site s with maximum d(s,C)
  - add s to C
- return C

- Note The centers are located at the sites.

Proof of Correctness

- Let C be the algorithm's output, and r be C's radius.
  - $r = max_{s \in S} min_{c \in C} d(s, c)$
- Lemma 1 For any $c, c' \in C, d(c, c') \geq r.$
- Proof Since r is the radius, there exists a point $s \in S$ at distance ≥ r from all the centers.
  - If there's no such s, then C's radius < r.
  - So s is distance ≥ r from c and c'.
  - Suppose WLOG c' is added to C after c.
  - If d(c,c')<r, then algorithm would add s to C instead of c', since s is farther.

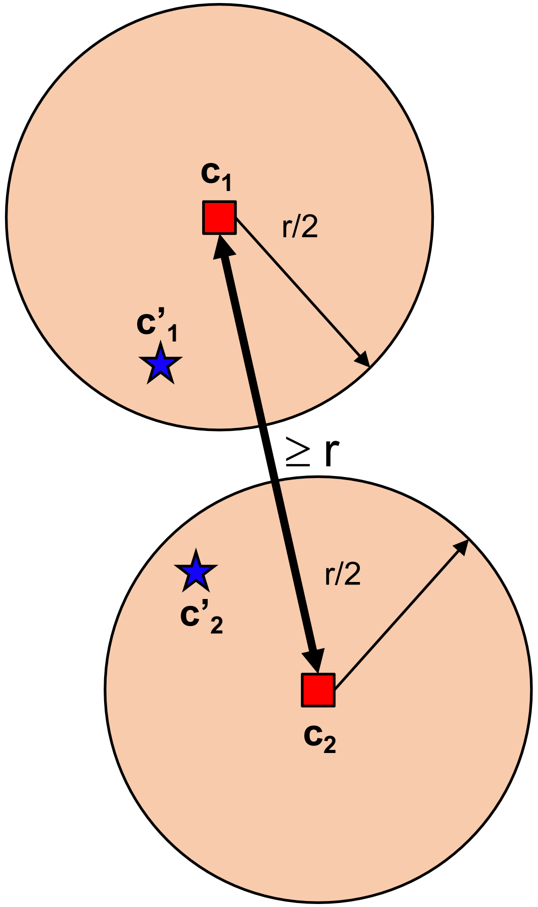


Proof of Correctness

- Cor There exist k+1 points mutually at distance $\geq$ r from each other.
  - By the lemma, the k centers are mutually $\geq$ r distance apart.
  - Also, there's an $s \in S$ at distance $\geq$ r from all the centers.
    - Otherwise, C's covering radius is < r.
  - So, the k centers plus s are the k+1 points.
- Call these k+1 points D.

Proof of Correctness

- Let C^* be an optimal cover with radius r^*.
- Lemma 2 Suppose $r > 2r^*$. Then for every $c \in D$, there exists a corresponding $c' \in C^*$. Furthermore, all these c' are unique.
- Proof Draw a circle of radius $r/2$ around each $c \in D$.
  - There must be a $c' \in C^*$ inside the circle, because
    - c is at most distance r^* away from its nearest center, since r^* is C^*'s radius.
    - $r/2 > r^*$.
  - Given $c_1, c_2 \in D$, let $c'_1, c'_2 \in C^*$ be inside c_1 and c_2's circle, resp.
  - c_1 and c_2's circles don't touch, because $d(c_1, c_2) \geq r$.
  - So $c'_1 \neq c'_2$



Proof of Correctness

- Thm Let C be the output of Gonzalez's algorithm and let C^* be an optimal k-center. Then $r(C) \leq 2r(C^*)$.
- Proof By Lemma 2, if $r(C)>2r(C^*)$, then for every $c \in D$, there is a unique $c' \in C^*$.
  - But there are k+1 points in D, by the corollary.
  - So, there are k+1 points in C^*. This is a contradiction because C^* is a k-center.