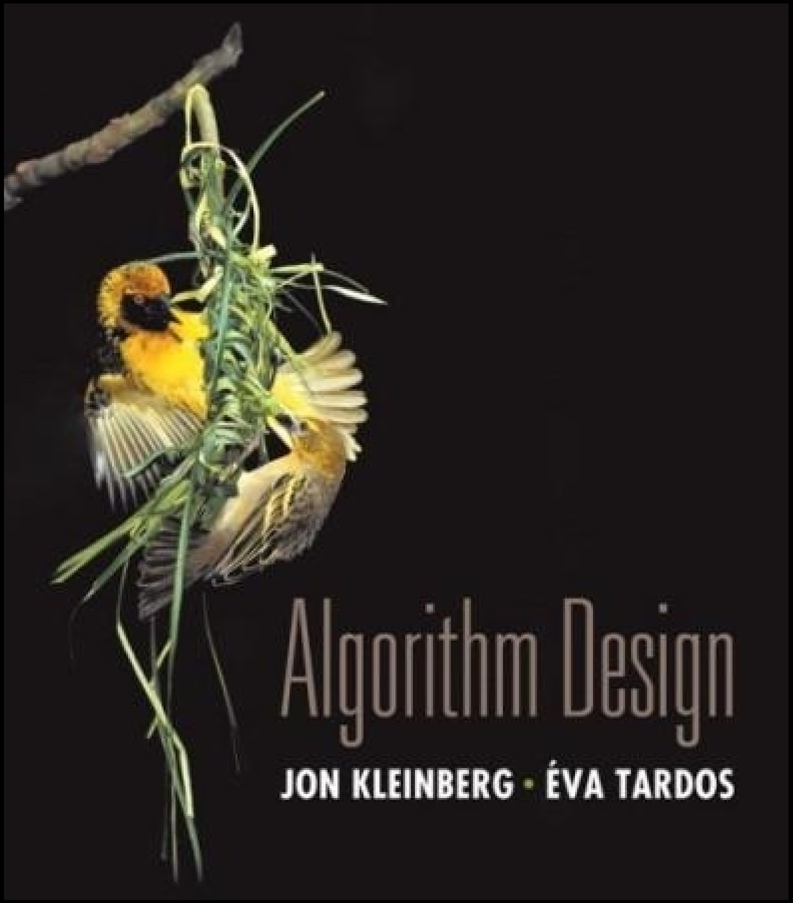


Chapter 11

Approximation Algorithms

Slides by Kevin Wayne.

Approximation Algorithms

Q. Suppose I need to solve an NP-hard problem. What should I do? A. Theory says you're unlikely to find a poly-time algorithm.

Must sacrifice one of three desired features.

- Solve problem to optimality.
- Solve problem in poly-time.
- Solve arbitrary instances of the problem.

-approximation algorithm.

- Guaranteed to run in poly-time.
- Guaranteed to solve arbitrary instance of the problem
- Guaranteed to find solution within ratio of true optimum.

Challenge. Need to prove a solution"s value is close to optimum, without even knowing what optimum value is!

11.1 Load Balancing

Load Balancing

Input. m identical machines ; n jobs, job j has processing time t_j . n Job j must run contiguously on one machine.
n A 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 = j J(i) t_j .

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

Load balancing. Assign each job to a machine to minimize makespan.

Load Balancing: List Scheduling

List-scheduling algorithm.

- Consider n jobs in some fixed order.
- Assign job j to machine whose load is smallest so far.

$LIST-SCHEDULING(m, n, t_1, t_2, \cdots, t_n)$

```
for i = 1 to m do
    L_i ← 0
    J(i) ← ∅
end for
for j = 1 to n do
    i = argmin_k L_k
    J(i) ← J(i) ∪ j
    L_i ← L_i + t_j
end for
return J(1), ··· , J(m).
```

Load Balancing: List Scheduling Analysis

Theorem.[Graham, 1966] Greedy algorithm is a 2-approximation.

- First worst-case analysis of an approximation algorithm.
- Need to compare resulting solution with optimal makespan L^*.

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

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

Lemma 2. The optimal makespan Pf.

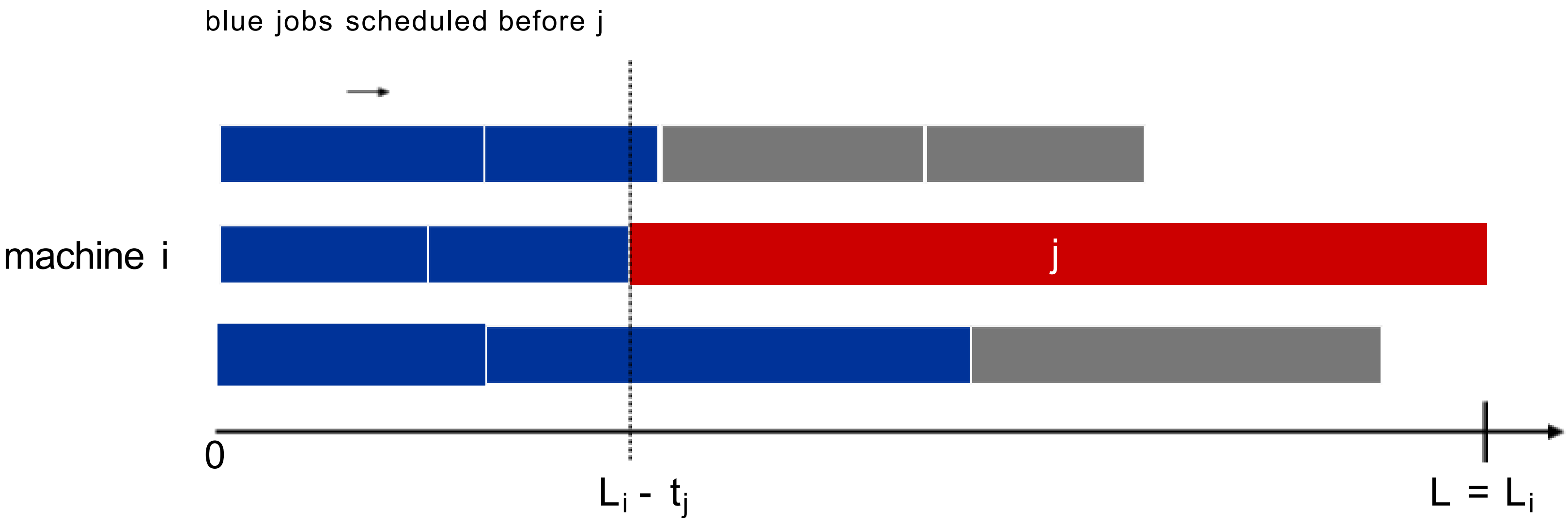
- The total processing time is $\sum_j t_j$.
- One of mmachines must do at least a 1/m fraction of total work. .

Load Balancing: List Scheduling Analysis

Theorem. Greedy algorithm is a 2-approximation.

Pf. Consider load L_i of bottleneck machine i.

- Let j be last job scheduled on machine i.
- When job j assigned to machine i, i had smallest load. Its load before assignment is $L_i - t_j$ $L_i - t_j$ L_k for all 1 k m.



Load Balancing: List Scheduling Analysis

Theorem. Greedy algorithm is a 2-approximation.

Pf. Consider load Li of bottleneck machine i.

n Let j be last job scheduled on machine i.

n When job j assigned to machine i, i had smallest load. Its load before assignment is Li - tj L i - tj L k for all 1 k m.

n Sum inequalities over all k and divide by m :

$$L_i - t_j \le \frac{1}{m}\sum_k L_k = \frac{1}{m}\sum_k t_k \le L^*$$

Lemma 2

n Now .

$$L_i = \underbrace{(L_i - t_j)}_{\le L^*} + \underbrace{t_j}_{\le L^*} \le 2L^*$$

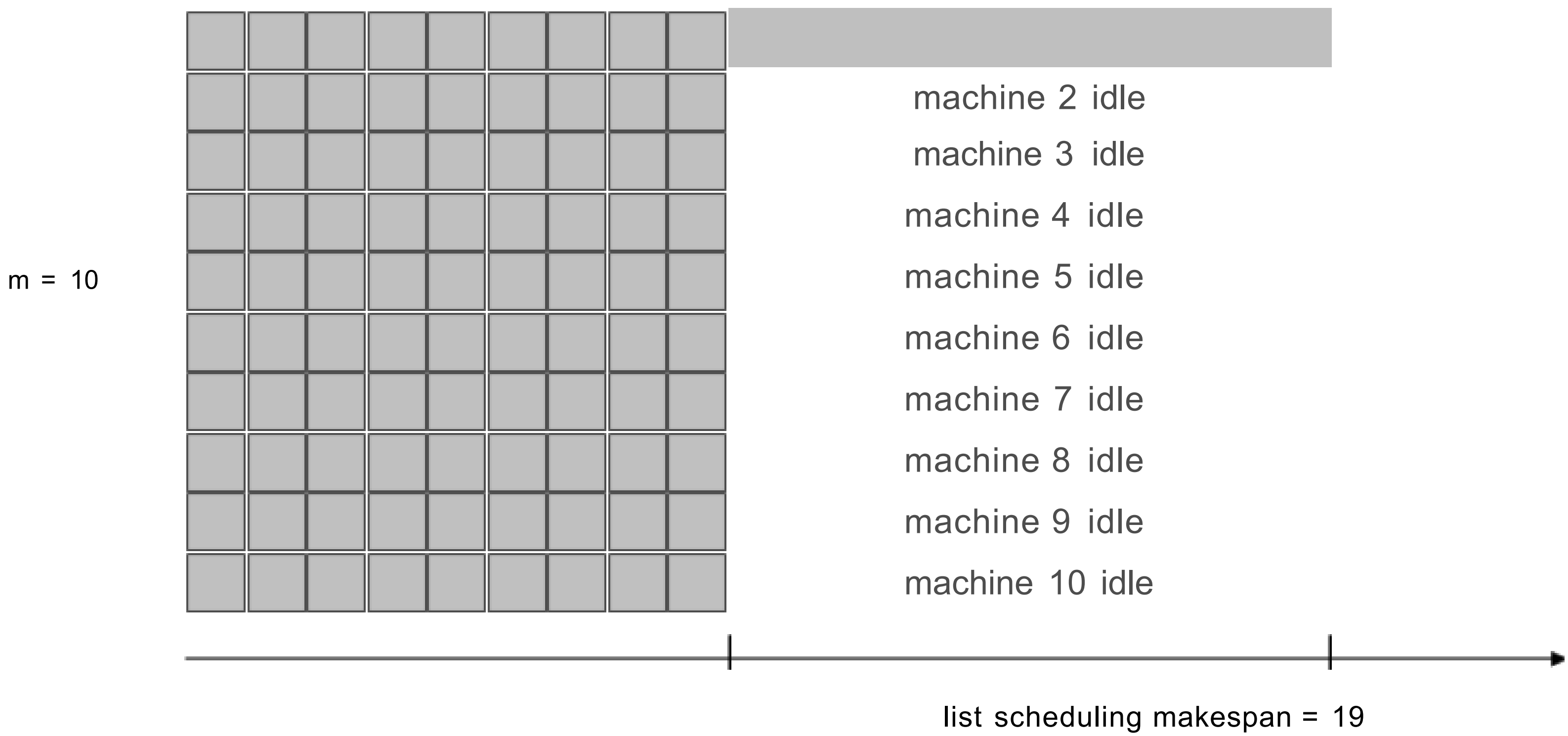
Lemma 1

Load Balancing: List Scheduling Analysis

Q. Is our analysis tight?

A. Essentially yes.

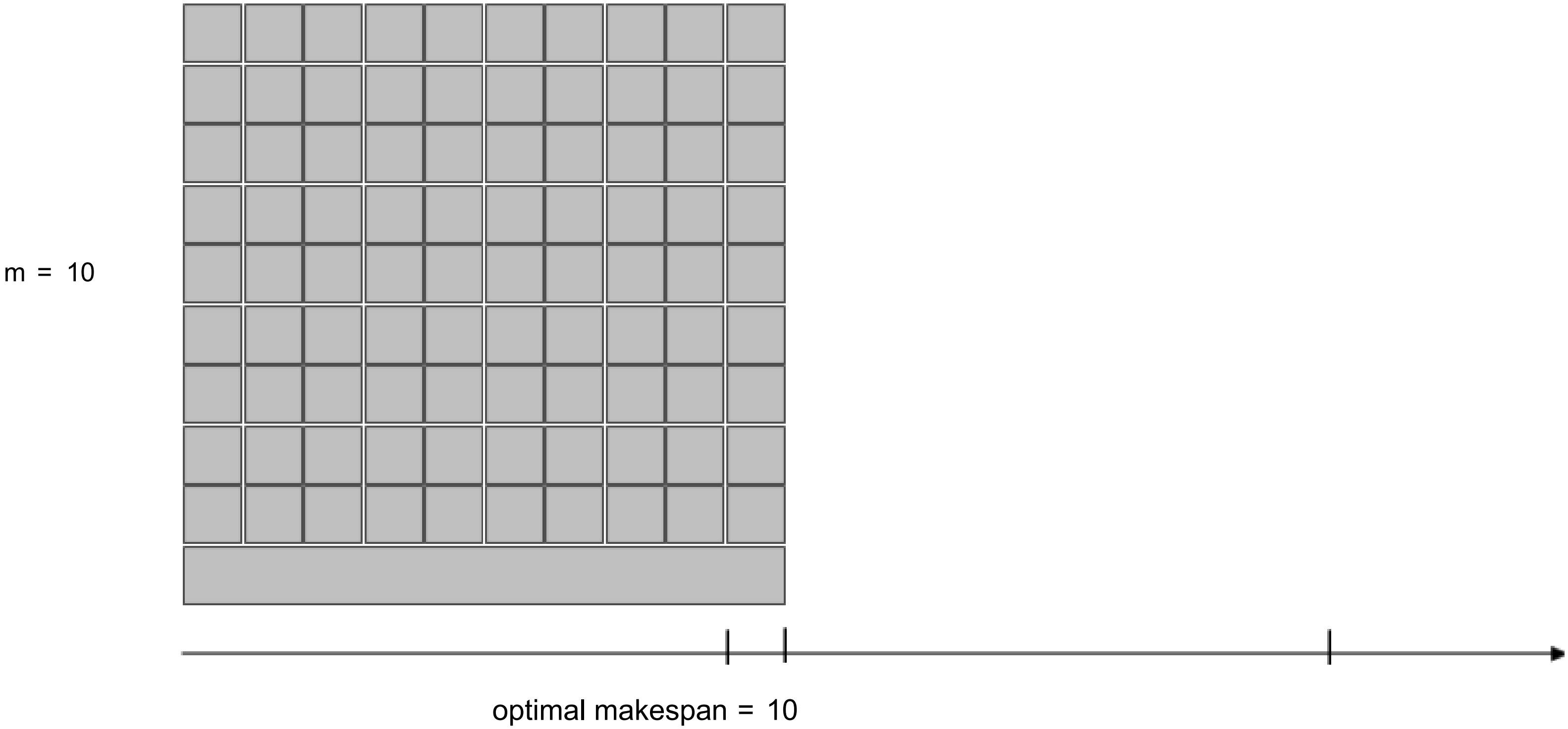
Ex: m machines, m(m- 1) jobs length 1 jobs, one job of length m



Load Balancing: List Scheduling Analysis

Q. Is our analysis tight?
A. Essentially yes.

Ex: m machines, m(m- 1) jobs length 1 jobs, one job of length m



Load Balancing: LPT Rule

Longest processing time (LPT). Sort n jobs in descending order of processing time, and then run list scheduling algorithm.

$LPT(m, n, t_1, t_2, \cdots, t_n)$

Sort jobs so that $t_1 \geq t_2 \geq \cdots \geq t_n$
for $i = 1$ to m **do**
    $L_i \leftarrow 0$
    $J(i) \leftarrow \emptyset$
end for
for $j = 1$ to n **do**
    $i = argmin_k L_k$
    $J(i) \leftarrow J(i) \cup j$
    $L_i \leftarrow L_i + t_j$
end for
return $J(1), \cdots, J(m)$.

Load Balancing: LPT Rule

Observation. If at most m jobs, then list-scheduling is optimal. Pf. Each job put on its own machine. .

Lemma 3. If there are more than m jobs, L^* 2 t_{m+1} .

Pf.

- Consider first m+1 jobst_1 , ... , t_{m+1} .
- Since the t_i 's are in descending order, each takes at least t_{m+1} time.
- There are m+1 jobs and m machines, so by pigeonhole principle, at least one machine gets two jobs. .

Theorem. LPT rule is a 3/2 approximation algorithm.

Pf. Same basic approach as for list scheduling.

[illegible]

Lemma 3

(by observation, can assume number of jobs > m)

Load Balancing: LPT Rule

Q. Is our 3/2 analysis tight?
A. No.

Theorem. [Graham, 1969] LPT rule is a 4/3-approximation. Pf. More sophisticated analysis of same algorithm.

Q. Is Graham's 4/3 analysis tight?
A. Essentially yes.

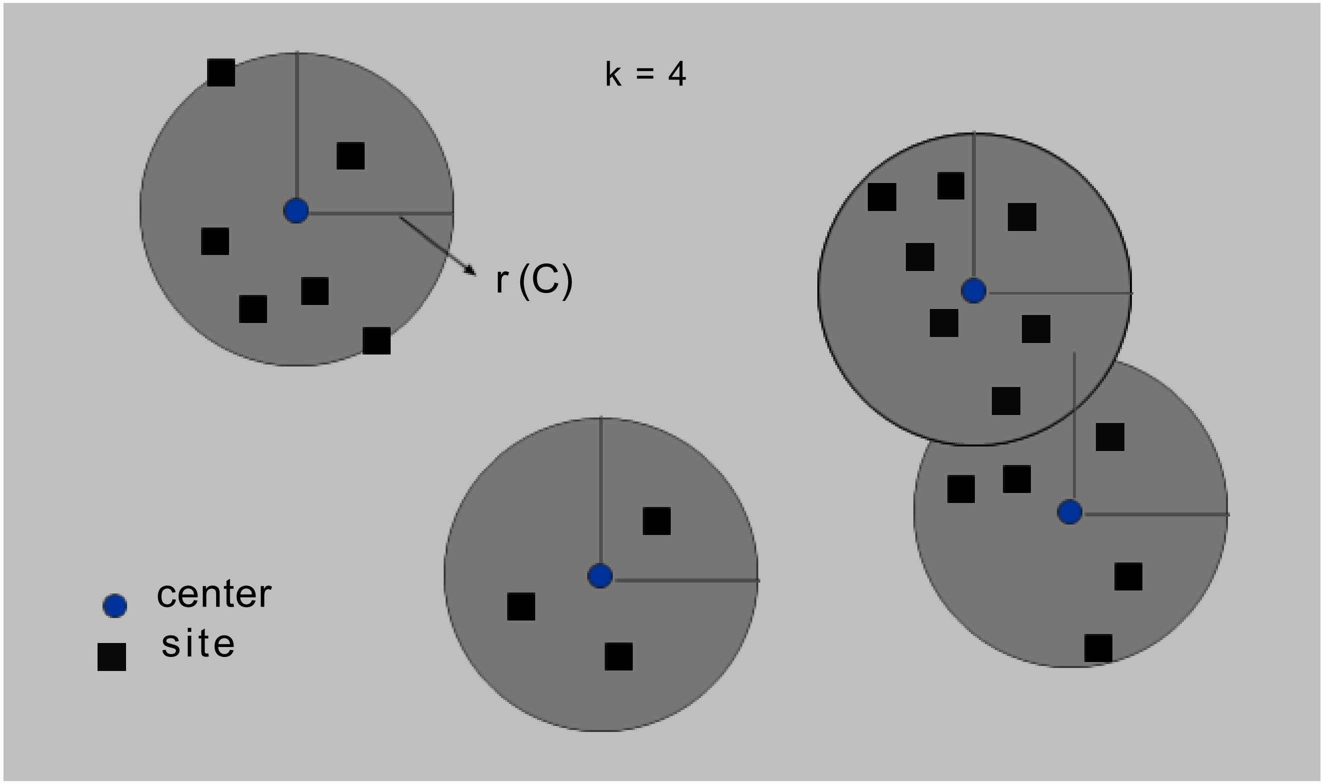
Ex: m machines, n = 2m+1 jobs, 2 jobs of length m+1, m+2, …, 2m-1 and one job of length m.

11.2 Center Selection

Center Selection Problem

Input. Set of n sites $x_1, \ldots, x_n$.

Center selection problem. Select k centers C so that maximum distance from a site to nearest center is minimized.



Center Selection Problem

Input. Set of n sites s_1 , … , s_n .

Center selection problem. Select k centers C so that maximum distance from a site to nearest center is minimized.

Notation.

- dist(x, y) = distance between x and y.
- dist(s_i, C) = min c C dist(s_i, c) = distance from s_i to closest center.
- r (C) = max$_i$ dist(s_i, C) = smallest covering radius.

Goal. Find set of centers C that minimizes r (C), subject to |C| = k.

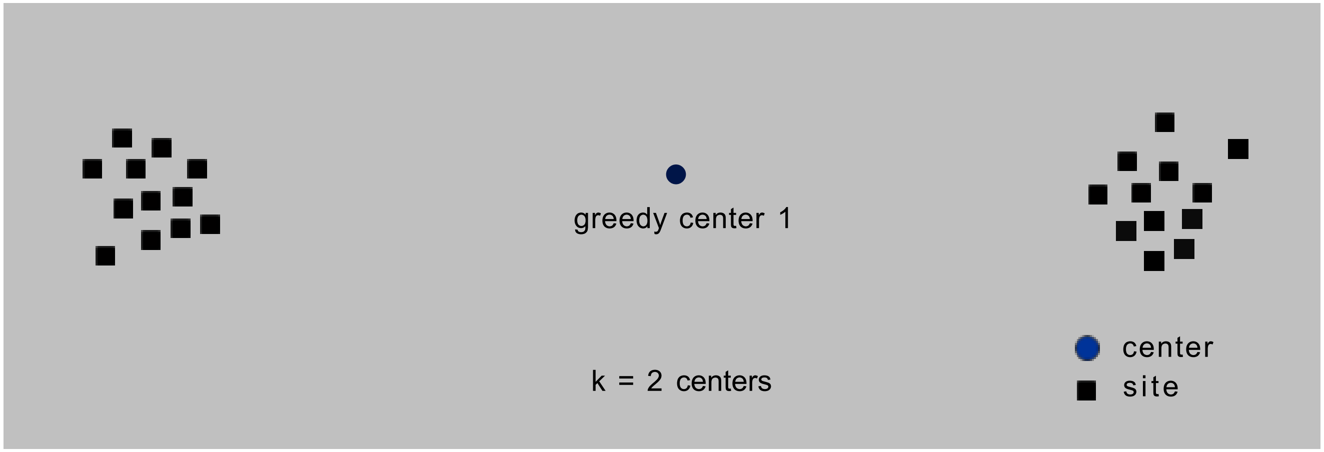
Distance function properties.

- dist(x, x) = 0 (identity)
- dist(x, y) = dist(y, x) (symmetry)
- dist(x, y) dist(x, z) + dist(z, y) (triangle inequality)

Greedy Algorithm : A False Start

Greedy algorithm. Put the first center at the best possible location for a single center, and then keep adding centers so as to reduce the covering radius each time by as much as possible.

Remark : arbitrarily bad!



Center Selection: Greedy Algorithm

Greedy algorithm. Repeatedly choose the next center to be the site

farthest from any existing center.

$GREEDY - CENTER - SELECTION(k, n, s_1, s_2, \cdots, s_n)$

$C \leftarrow \emptyset$.
for $i = 1$ to k **do**
    Select a site s_i with maximum distance $dist(s_i, C)$
    $C \leftarrow C \cup s_i$
end for
return C

Observation. Upon termination all centers in C are pairwise at least r (C) apart.

Pf. By construction of algorithm.

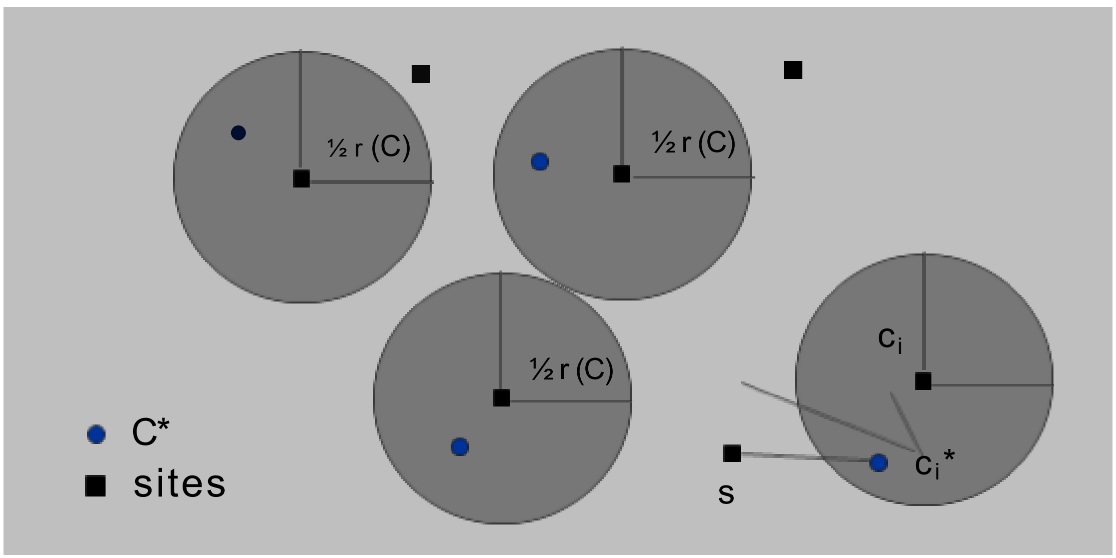
Center Selection: Analysis of Greedy Algorithm

Theorem. Let C* be an optimal set of centers. Then r (C) 2r (C*).

Pf. (by contradiction) Assume r (C*) < ½ r (C).

- For each site c_i in C, consider ball of radius ½ r (C) around it.
- Exactly one c_i^* in each ball ; let c_i be the site paired with c_i^*.
- Consider any site s and its closest center c_i^* in C*.
- dist(s, C) dist(s, c_i) dist(s, c_i^*) + dist(c_i^*, c_i) 2r (C*).
- Thus r (C) 2r (C*). .

-inequality r (C*) since c_i^* is closest center



Center Selection

Theorem. Greedy algorithm is a 2-approximation for center selection problem.

Question. Is there hope of a 3/2-approximation? 4/3?

Theorem. Unless P = NP, there no -approximation for center-selection problem for any < 2.

11.4 The Pricing Method: Vertex Cover