Approximation Algorithms
Load Balancing and k-Center
CMSC 451, Summer 2009

Some slides by Kevin Wayne.
Copyright © 2005 Pearson-Addison Wesley.
All rights reserved.

Homework 5
• Exercises 11.1, 11.4, 11.10 from the textbook.
  Additional problem.
• For 11.4: Give an LP and accompanying algorithm to approximately solve this problem. You do not need to prove that the LP is correct, but should give some explanation as to why your algorithm gives a b-approximation of the optimal solution.
• Due next Monday

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

LPT-List-Scheduling(m, n, t₁, t₂, …, tₙ) {
  Sort jobs so that t₁ ≥ t₂ ≥ … ≥ tₙ
  for i = 1 to m {
    Lᵢ ← 0
    J(i) ← φ
  }
  for j = 1 to n {
    i = argmin₁≤k≤m Lₖ
    J(i) ← J(i) ∪ {j}
    Lᵢ ← Lᵢ + tᵢ
  }
  return J(1), …, J(m)
}

jobs assigned to machine i
load on machine i
machine i has smallest load
assign job j to machine i
update load of machine i

Practice Proof: 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* ≥ 2tₘ₊₁.
Pf.

Theorem. LPT rule is a 3/2 approximation algorithm.
Pf. Same basic approach as for list scheduling.

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* ≥ 2tₘ₊₁.
Pf.

Theorem. LPT rule is a 3/2 approximation algorithm.
Pf. Same basic approach as for list scheduling.

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

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

**Input.** Set of n sites $s_1, \ldots, s_n$ and integer $k > 0$.

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

**Notation.**
- $\mathbf{dist}(x, y) =$ distance between $x$ and $y$.
- $\mathbf{dist}(s_i, C) =$ distance from $s_i$ to closest center.
- $r(C) = \max_{i} \mathbf{dist}(s_i, C) =$ smallest covering radius.

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

**Distance function properties.**
- $\mathbf{dist}(x, x) = 0$ (identity)
- $\mathbf{dist}(x, y) = \mathbf{dist}(y, x)$ (symmetry)
- $\mathbf{dist}(x, y) \leq \mathbf{dist}(x, z) + \mathbf{dist}(z, y)$ (triangle inequality)

**Some distance function possibilities.**
- Euclidean distance
- Travel time
- Edit distance

---

**k-Center Selection Example**

Ex: each site is a point in the plane, a center can be any point in the plane, $\mathbf{dist}(x, y) =$ Euclidean distance.

Remark: search can be infinite!
k-Center Selection: Greedy Algorithm Ideas

**Greedy selection outline:**
- Repeatedly add a point to the set of centers based on some selection criteria.
- Recalculate the distances from each point to its closest center.

**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!

Greedy-Center-Selection(k, n, s_1, s_2, \ldots, s_n) {
  \text{C} = \emptyset
  \text{foreach} s_i: \text{dist}(s_i, C) = \text{infinity}
  \text{Select} s_i \text{ with maximum dist}(s_i, C)
  \text{Add} s_i \text{ to C}
  \text{foreach} s_i: \text{dist}(s_i, C) = \text{distance to closest site in C}
  \text{return} C
}

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

**Pf.** By construction of algorithm.

**k-Center Selection: Analysis of Greedy Algorithm**

**Lemma.** Let s be some site covered by center c* in the optimal solution with optimal radius r, then if s is chosen as a center it covers everything c* covered within radius 2r.

**Pf.** (by picture)

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

**Pf.** (by contradiction) Assume r(C*) \neq \frac{1}{2} r(C).
- For each site c in C, consider ball of radius \frac{1}{2} r(C) around it.
- By Lemma, exactly one c* in each ball since otherwise not optimal; let c be the site paired with c*.
- Consider any site s and its closest center c* in C*.
- dist(s, C) = dist(s, c) = dist(s, c*) = dist(c*, c) = 2r(C*)
- Thus r(C) = 2r(C*)

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

**Remark.** Greedy algorithm always places centers at sites, but is still within a factor of 2 of best solution that is allowed to place centers anywhere.

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

**Theorem.** Unless P = NP, there no \(\rho\)-approximation for center-selection problem for any \(\rho < 2\).
Tight Example.

Is our algorithm analysis tight?

Tight example for k=2:

Greedy: distances = n^2
Opt: distances = n/4