LECTURE 9: LOCAL SEARCH, SHORTEST PATH
ANNOUNCEMENTS

- Homework 2 out – due next Friday

- Contacting the TAs: adv-algorithms-ta-fall18@googlegroups.com
LAST CLASS

- Greedy algorithm has a non-trivial approximation guarantee
- Analysis the tricky part of greedy algorithms
- Local search – start with any feasible solution, improve it
Matching: suppose we have $n$ children, $n$ gifts and “happiness” values $H_{ij}$. Assign gifts to children to maximize “total happiness”.

- Local search algorithm:
  - Start with any assignment
  - For each pair of children, see if “swapping gifts” improves total cost
Denote "locally optimal" solution by assignment \( p_1, p_2, \ldots, p_n \)

\[
H_{i, p_i} + H_{j, p_j} \geq H_{i, p_j} + H_{j, p_i}
\]

\( \forall i, j \)

(because otherwise, we would have swapped gifts of children \( i, j \)).
Theorem. Consider any solution that is “locally optimum”. Its cost is at least \((1/2) \times \text{OPT}\).

Denote “locally optimal” solution by assignment \(q_1, q_2, \ldots, q_n\).

\[
\text{value of OPT solution: } \sum_{i} H_{i, q_i}
\]

\[
\text{value of solution we found: } \sum_{i} H_{i, q_i}
\]
Let $j$ be the index that was assigned $q_i$ in our solution. I.e., $j = q_i$

\[ \sum (H_{i,j} + H_{j,q_i}) \geq \sum H_{i,q_i} + H_{j,p_i} \geq \sum H_{i,q_i} \]

\[
\text{val of our solution} + \sum H_{j,q_i} \\
\geq 2 \times \text{val. of our solution}.
\]
CLASSIC EXAMPLE — GRADIENT DESCENT

Warning: multivariate calculus coming up

Problem. Given a convex function $f$ over a domain $D$, find

$$\arg\min_{x \in D} f(x)$$

Convex

$$f(x + y) \leq f(x) + f(y)$$

$$f\left(\frac{x + y}{2}\right) \leq \frac{f(x) + f(y)}{2}$$

Continuously differentiable

$$f\left(\lambda x + (1-\lambda)y\right) \leq \lambda f(x) + (1-\lambda)f(y)$$

$$\lambda \in (0, 1)$$
Fact: If \( f \) satisfies \( f\left(\frac{x+y}{2}\right) \leq \frac{f(x) + f(y)}{2} \),

then the min of \( f \) over a convex domain \( D \) is unique.
LOCAL SEARCH

**Problem.** Given a convex function $f$ over a domain $D$, find

$$\arg\min_{x \in D} f(x)$$

- start with any pt, look at all nbrs
- if one of them has lower $f()$ value, move to that nbr, continue.
- \( f : \mathbb{R}^n \rightarrow \mathbb{R} \)

- Start with any \( x \) in \( D \).

**Algorithm:**

1. If \( \exists y \in \text{Ball}(x, \delta) \) s.t. \( f(y) < f(x) \) and \( y \in D \), move the first locally optimal point \( y \), and repeat.

**Claim:** This will always find the minimum value for \( f \).

If \( f(x^*) \leq f(x) \)

then \( f(y) < f(x) \),

\( y = \lambda x + (1-\lambda)x^* \)

\( f(y) \leq \lambda f(x) + (1-\lambda)f(x^*) \)
Problem. Given a convex function $f$ over a domain $D$, find 

$$\text{argmin}_{x \in D} f(x)$$

- What is a good direction to move?
- Need domain $D$ to be convex

This is an easy step — just move in the direction opposite the "gradient."
COMMENTS ON GRADIENT DESCENT

- How many steps to take?
- What is the step size? (should it be fixed?)

Perhaps the most used algorithm in ML.
LOCAL SEARCH SUMMARY

- Start with solution, move to “neighboring” solution by changing current solution slightly
- How neighbors are chosen determines complexity
- Not always optimal $\left( \in Hw - 2 \right)$.
- Number of steps can be LARGE if not done carefully – standard trick: always make sure you improve by significant amount
GRAPH ALGORITHMS
LECTURE 9

GRAPH BASICS

- Common data structures — adjacency lists, adjacency matrix
- Basic procedures: Breadth first search, depth first search
- **TODAY:** shortest paths — one of the first polynomial time algorithms!

(1950s)
**Shortest path problem:** given a (possibly directed) graph $G = (V, E)$, and two vertices $u, v$, find the length of the shortest path from $u$ to $v$.

- What if the graph is not weighted?
- Negative weight edges?
BFS + DISTANCE UPDATES?

\[ \forall v \in G, \text{ maintain} \]
- \[ \text{dist}[v] \rightarrow \text{initialized using BFS} \]
- See for an edge \( ij \), see if
  \[ \text{dist}[j] > \text{dist}[i] + l_{ij} \]
  we know that \( \text{dist}[j] \) is incorrect!
  \[ \text{dist}[j] = \text{dist}[i] + l_{ij} \]
DISTANCE UPDATES — FULL ALGORITHM
CORRECTNESS
DIJKSTRA'S ALGORITHM — CAN WE AVOID MULTIPLE VISITS?
DIJKSTRA’S ALGORITHM