More Complexity Theory via Reachability

CS254

Chris Pollett

Aug. 28, 2006.

Outline

- Maximum Flow
- Bipartite Matching
- Traveling Salesman

Motivation

- Reachability last day was a decision problem.
- Our notion of tractable versus intractable is based on decision problems.
- Many real-world problems however return values. In particular, one often has optimization problems.
- It turns out that feasibility for these problems is closely connected to feasibility of decision problems.

Maximum Flow

As an example…

- A **network** N=(V, E, s, t, c) is a graph (V,E) with two distinguished vertices s,t and with a capacity $0 \leq c(i,j)$ on edge (i,j) in the graph.
- s should be a source and t a sink.
- A **flow** f is an assignment to each edge (i,j) other than s and t of a value f(i,j) such that $0 \leq f(i,j) \leq c(i,j)$ and such that $\sum_i f(i,j) = \sum_k f(j,k)$.
- The **value** of a flow is $\sum_k f(s,k)$.
- The problem MAX FLOW is to determine the maximum possible value of a flow on a network.
- Since we are doing a maximization it is called an *optimization problem*.

An Algorithm For MAX FLOW

- Notice if f is a flow and f´ is another flow with greater value the $\Delta f = f´ - f$ is almost a flow except some f(i,j)'s might be negative.
- It is a flow in a modified network N(f) = (V, E´, s, t, c´), where

  $E´ = E - \{(i,j) \mid f(i,j) = c(i,j)\} \cup \{(i,j) \mid (j,i) \text{ in } E, f(i,j) > 0\}$,

  and $c´(i,j) = c(i,j) - f(i,j)$ for (i,j) in E

  and $c´(i,j) = f(i,j)$ for (i,j) in E´ - E.
- So telling if a flow f is a maximum is the same as deciding if there is a positive flow in N(f).
- But there will be such a flow if there is path in N(f) from s to t. i.e., an instance of REACHABILITY.
- Thus, REACHABILITY suggests an algorithm for MAX FLOW:

  (i) Start from the initially all zero flow.

  (ii) Choose a path of positive capacities from s to t, if it exists.

  If so, add to the current flow the value of the least edge on this path to make a new larger flow.

  (iii) Generate new N(f) based on this flow and repeat untit no path.

Runtime of MAX FLOW algorithm

- Each iteration of the algorithm is $O(n^2)$ -- the time to do REACHABILITY.
- There can be at most nC phases where C is the maximum value of C(i,j) for any edge.
- This is because the graph has n edges, so there are fewer than n edges leaving s. So the maximum value of the flow is nC. Since our flows have integer values, each intermediate flow must go up by at least one.
- So the total time for this algorithm is $O(n^3C)$.
- This could be bad if $C=\Omega(2^n)$.
- It turns out if one always augments by the shortest path then the algorithm is $O(n^5)$.

Some More Motivation

- Another reason to study polynomial time as our notion of tractable is its closure properties under reductions.
- That is, if we can solve one problem in polynomial time and we can show another problem polynomial time reduces to the first problem, then the second problem will also be in polynomial time.

Bipartite Matching

As an example…

- A **bipartite graph** is a triple B=(U,V, E) where U is a set of nodes called boys, V is a set of nodes called girls, |U|=|V|=n, and E $\subseteq$ U x V is a set of edges.
- A **matching** is a set M $\subseteq$ E of n edges, such that for any two edges (u,v), (u´, v´) in M, u$\neq$u´ and v$\neq$v´.
- MATCHING is the problem to determine whether or not a bipartite graph has a matching.
- Let MAX FLOW (D) be the decision problem: Does a graph have a maximum flow with value greater than D?
- We have an $O(n^5)$ algorithm for this.
- Given bipartite graph B, we can make a network N by adding vertices s, t and connecting s to each edge in U and connecting each edge in V to t. Set c(i,j) = 1 for all edges (i,j).
- Notice B has a matching iff N has a maximum flow of value $\geq$ n.
- As this reduction is easily computable this gives an $O(n^5)$ algorithm for MATCHING.

Traveling Salesman Problem

- So what is the dividing line between tractable and intractable?
- Consider the following problem given n cities and a distance function d(i,j) between pairs of cities is there a way to visit all cities with a total cost less than D?
- Obviously given a tour of each city it is easy to check if it has cost less than D.
- A particular tour is essentially a permutation and there are n! permutations. So searching all of them is intractable.
- It is unknown however if there is no better algorithm.
- This is an example of a problem in NP.
- This class is also closed under polynomial time reductions.
- One of our goals this semester will be to understand the relationship of P to NP.