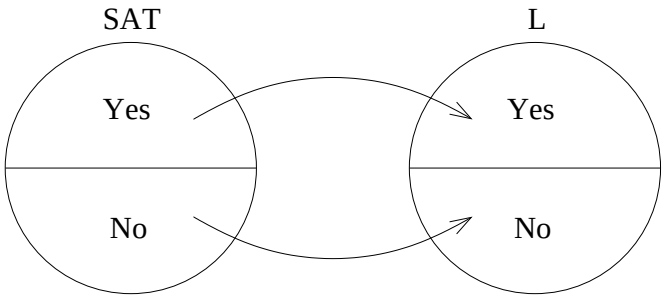
CS880: Approximations Algorithms	
Scribe: Tom Watson	**Lecturer:** Shuchi Chawla
Topic: Inapproximability	**Date:** 4/27/2007

So far in this course, we have been proving upper bounds on the approximation factors achievable for certain NP-hard problems by giving approximation algorithms for them. In this lecture, we shift gears and prove lower bounds for some of these problems, under the assumption that $P \neq NP$. For some problems, essentially matching upper and lower bounds are known, indicating that the approximability properties of these problems are well-understood. We show that (nonmetric) TSP cannot be approximated within any factor. We also show that the Edge Disjoint Paths problem is NP-hard to approximate within a factor of $O(m^{1/2-\epsilon})$ for all $\epsilon > 0$, essentially matching the $O(\sqrt{m})$ algorithm obtained in a previous lecture. Finally, we give a bootstrapping argument showing that if the Clique problem cannot be approximated within some constant factor, then it cannot be approximated within any constant factor.

27.1 Framework

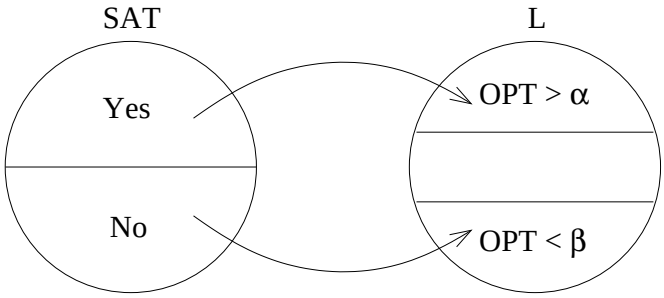
The study of approximation algorithms is mostly focused on NP-hard problems. These problems cannot be solved in polynomial time, assuming $P \neq NP$. Our general goal is to show that under the same assumption, not only can these problems not be solved exactly, but they cannot even be approximated within certain factors in polynomial time. While the complexities of solving NP-complete problems exactly are all polynomially related, we have already seen evidence that these problems can exhibit wildly different approximability properties. For example, we have seen that the Knapsack problem has an FPTAS, indicating that it has a loose grip on its intractability, in some sense. We will shortly see that TSP cannot be approximated within any factor, indicating that it has a very strong grip on its intractability.

We begin by recalling the standard method for showing that a language L is NP-hard. One begins with another NP-hard language, say SAT, and exhibits a polynomial-time mapping reduction from SAT to L. This reduction takes an instance x and produces an instance y such that $x \in$ SAT iff $y \in L$. This can be viewed pictorially as follows.



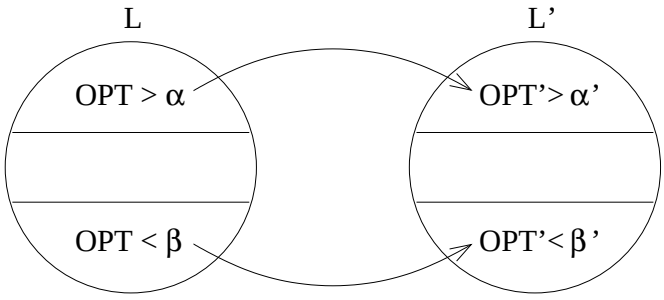
Thus if we had a polynomial-time (exact) algorithm for L then we could compose it with the

reduction to obtain a polynomial-time algorithm for SAT. For inapproximability results, L is an optimization problem, say a maximization problem, rather than a language, and we would like it to be the case that composing a polynomial-time *approximation* algorithm for L with the reduction yields a polynomial time-algorithm for SAT. This goal is achieved with a *gap-introducing reduction*, illustrated as follows.



This type of reduction maps the "yes" instances of SAT to instances of L with optimal objective value greater than α, and it maps the "no" instances of SAT to instances of L with optimal objective value less than β, for some parameters α and β. An α/β-approximation algorithm for L has the property that on instances with $OPT < \beta$ it produces a solution of value less than β (obviously) and on instances with $OPT > \alpha$ it produces a solution of value greater than $\frac{\alpha}{\alpha/\beta} = \beta$. Thus an exact algorithm for SAT can be obtained by applying the mapping reduction followed by the purported approximation algorithm, and testing whether the output has value greater than or less than β. Thus exhibiting such a gap-introducing reduction shows that it is NP-hard to approximate L within a factor of α/β.

With NP-hardness proofs, one doesn't always reduce from SAT; after proving that a problem L is NP-hard, it gets added to the arsenal of problems one can reduce from. Similarly, once we have established an inapproximability result for L, we would like to be able to establish inapproximability results for other problems by means of a reduction from L. However, in this case we don't need to do the work of introducing a gap with our reduction; we merely need to preserve the gap. This is achieved with a *gap-preserving reduction*, illustrated as follows.



It is clear from the picture that if there is a gap-introducing reduction from SAT to L as in the previous figure then composing it with this gap-preserving reduction yields a gap-introducing

reduction from SAT to L', proving that L' is NP-hard to approximate with a factor of α'/β'. We will see examples where L and L' are the same problem but $\alpha'/\beta' > \alpha/\beta$; these reductions can appropriately be called *gap-amplifying reductions*.

We have been assuming that the problems we are dealing with are maximization problems, but the same concepts apply to minimization problems. We now look at some concrete examples of how this framework is employed to prove inapproximability results.

27.2 Traveling Salesperson Problem

Our first example is for the Traveling Salesperson Problem (TSP), in which we are given a set of n nodes and a distance between each pair of nodes, and we seek a minimum-length tour that visits every node exactly once. Recall that in a previous lecture, we obtained a 3/2-approximation algorithm for Metric TSP. How much does the approximability deteriorate when we eliminate the requirement that the distances form a metric? We show that TSP in its full generality is actually inapproximable in a very strong sense.

Theorem 27.2.1 *TSP cannot be approximated within any factor unless $P = NP$.*

Proof: Recall the standard NP-hardness proof for TSP, which is a reduction from the Hamiltonian Tour problem. Given an unweighted graph G, we construct a TSP instance on the same set of nodes, where nodes adjacent in G are at distance 1 and nodes not adjacent in G are at distance ℓ for some value $\ell > 1$. Now if G has a Hamiltonian tour then the TSP instance has a tour of length n (the number of nodes) and otherwise every tour in the TSP instance has length at least $n - 1 + \ell$. Thus this reduction is, in fact, a gap-introducing reduction, proving that TSP is hard to approximate within any factor less than $(n-1+\ell)/n$. Since we can choose ℓ to be as large as we like, this proves the theorem. ■

Note that this result critically uses the fact that the distances are not required to form a metric. The largest we can make ℓ and still be guaranteed that the distances form a metric is $\ell = 2$. This implies that Metric TSP is NP-hard to approximate within any factor less than $(n - 1 + 2)/n = 1 + 1/n$. However, it is known that Metric TSP is NP-hard to approximate within a factor of $1 + \epsilon$ for some constant $\epsilon > 0$.

27.3 Edge Disjoint Paths

Our second example is for the Edge Disjoint Paths (EDP) problem, in which we are given an unweighted graph and k terminal pairs (s_i, t_i) and seek to connect s_i to t_i with edge disjoint paths for as many i as possible. In a previous lecture, we gave an $O(\sqrt{m})$-approximation algorithm for this problem, where m is the number of edges. We now show that that result is essentially tight.

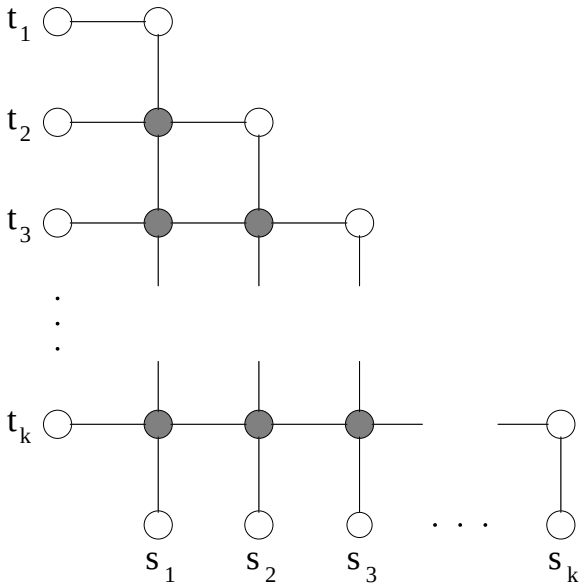
There is an NP-hardness proof for EDP where the hard EDP instances only have $k = 2$ terminal pairs. Since for such instances, the optimum objective value is either 2 or at most 1, we immediately get the following result.

Theorem 27.3.1 *For $k = 2$, EDP is NP-hard to approximate within any factor less than 2.*

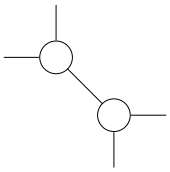
We employ a bootstrapping argument to show that Theorem 27.3.1 implies the following result.

Theorem 27.3.2 *EDP is NP-hard to approximate within a factor of* $O(m^{1/2-\epsilon})$ *for all* $\epsilon > 0$.

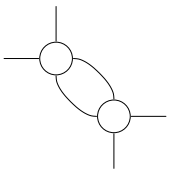
Proof: We give a gap-amplifying reduction from EDP with 2 source-sink pairs to EDP. Suppose we are given a graph H with two terminal pairs (x_1, y_1) and (x_2, y_2). We construct an EDP instance with k terminal pairs, for some k to be determined later, in a graph G having the following general structure.



If we expand each of the filled-in nodes into a single edge like this:

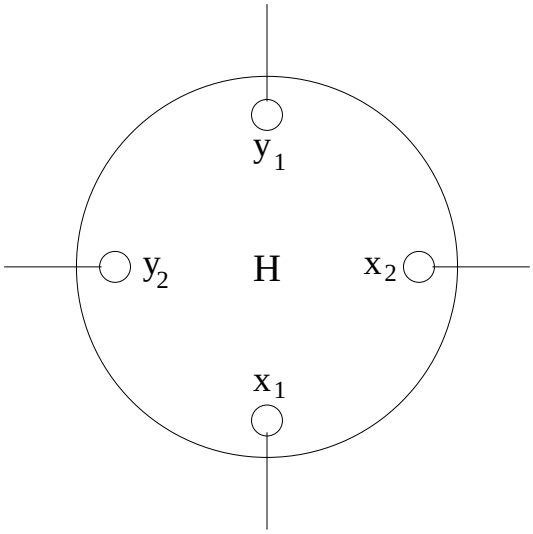


then the optimal objective value is always 1 since for every two (s_i, t_i) pairs, the unique paths connecting s_i to t_i intersect at some filled-in node and would thus have to share the edge there. If we expand each of the filled-in nodes into a pair of edges like this:



then the optimal objective value is always k, since each filled-in node can accomodate both of the (s_i, t_i) pairs whose paths intersect there.

Instead, we expand each filled-in node into a copy of H, as follows.



It follows that if H has edge disjoint paths from x_1 to y_1 and from x_2 to y_2, then each of the filled-in nodes can accommodate both (s_i, t_i) pairs that would like to use it, implying that the optimal objective value for G is k, and otherwise each filled-in node can accomodate at most one (s_i, t_i) pair, implying that the optimal objective value for G is 1. We have succeeded in amplifying the trivial factor 2 gap for H into a factor k gap for G.

Let h denote the number of edges in H. For any given $\epsilon > 0$, we can take $k = h^{1/\epsilon}$ and the reduction still runs in polynomial time. Since the number of edges in G is $m = O(k^2 h) = O(k^{2+\epsilon})$, the inapproximability factor we achieve is $k = \Omega(m^{1/(2+\epsilon)})$, which suffices to prove the theorem. ■

Interestingly, the above proof shows a large gap between the objective values in an EDP instance and the same instance of the fractional version of EDP, which is just a multicommodity flow problem. If H is connected but does not have edge disjoint paths connecting both terminal pairs, then the optimal integral objective value in G is 1, but the optimal fractional objective value is $k/2$ since we can route $1/2$ unit of each commodity without sending more than 1 unit of flow along any edge in G.

27.4 Clique

Our third example is for the Clique problem, in which we are given an unweighted graph and seek a maximum size clique in it. The best known approximation algorithm for this problem achieves an approximation guarantee of $O(n/\log^2 n)$, so even when there is a clique of size $\Omega(n)$, the algorithm only guarantees that it will find a clique of size $\Omega(\log^2 n)$. Indeed, the known inapproximability results come close to matching this bound; the best such result to date shows that for all $\epsilon > 0$, Clique cannot be approximated within a factor of $n^{1-\epsilon}$ unless $NP = ZPP$ [1].

We now prove the following result, which is another example of a bootstrapping argument.

Theorem 27.4.1 *If Clique is NP-hard to approximate within a factor of α, then it is also NP-hard to approximate with a factor of α^2.*

Proof: We give a gap-amplifying reduction from Clique to itself. Suppose we are given a graph $G = (V, E)$. We construct the graph $G' = (V', E')$ as follows. We take $V' = V \times V$, and let $\{(u, v), (w, x)\} \in E'$ iff both of the following conditions hold:

1) $\{u, w\} \in E$ or $u = w$

2) $\{v, x\} \in E$ or $v = x$.

We claim that if the maximum clique size in G is k, then the maximum clique size in G' is k^2. We first show that the maximum clique size in G' is at least k^2 by taking an arbitrary clique $Q \subseteq V$ in G and showing that $Q \times Q \subseteq V'$ is a clique in G'. Let (u, v) and (w, x) be nodes in $Q \times Q$. Since $u, w \in Q$, condition 1 follows immediately. Since $v, x \in Q$, condition 2 follows immediately. Thus $\{(u, v), (w, x)\} \in E'$ and $Q \times Q$ is a clique of size k^2 in G'.

Now we show that the maximum clique size in G' is at most k^2. Consider an arbitrary clique $S \subseteq V \times V$ in G', and let $S_\ell = \{u : (u, v) \in S \text{ for some } v \in V\}$ and $S_r = \{v : (u, v) \in S \text{ for some } u \in V\}$. Now S_ℓ is a clique in G since if $u \neq w$ are nodes in S_ℓ then there exist $v, x \in V$ such that $(u, v), (w, x) \in S$ and thus $\{(u, v), (w, x)\} \in E'$, which implies that $\{u, w\} \in E$ by condition 1. Similarly, S_r is a clique in G. We have $|S_\ell| \cdot |S_r| \geq |S|$, and since $|S_\ell| \leq k$ and $|S_r| \leq k$, we get that $|S| \leq k^2$, as desired. This finishes the proof of our claim.

It follows that if the maximum clique size in G is either at least s or less than s/α, for some value s, then the maximum clique size in G' is either at least s^2 or less than $(s/\alpha)^2 = s^2/\alpha^2$. Thus we have succeeded in amplifying a gap of α to a gap of α^2. ■

In the next lecture, we will see inapproximability results based the PCP Theorem, a deep result in complexity theory.

References

[1] J. Hastad. Clique is Hard to Approximate within n to the power 1-epsilon. In *Acta Mathematica*, 182, 1999, pp. 105-142.