Inapproximability

Problems not absolutely approximable

Problems not approximable within constant ration

Maximum Clique

- No polynomial time approximation algorithm for the maximum clique problem can guarantee

  $|A(G)\text{-OPT}(G)|\leq K$

  for a fixed constant K, unless P=NP.
- Proof: Suppose A is such an algorithm. We will show that A can be used to derive a polynomial time optimization algorithm for Maximum Clique, contradicting the assumption that P≠NP.

Given as an instance any graph G, the algorithm is to

- construct a new graph G' that consists of K+1 copy copies of G with that every two vertices from the different copies are joined by an edge.
  - It is easy to see that $\text{OPT}(G')=(K+1)\text{OPT}(G)$.
- Run A on graph G' to produce a clique C' of size $A(G') \geq (K+1)\text{OPT}(G)\text{-}K$.
- Output C such that $C= C'\cap G_i$ and $|C|=\max_{1\leq i\leq K+1}|C'\cap G_i|$, where G_i is the i-th copy of G.

C is a clique for G, and

$|C| \geq [(K+1)\text{OPT}(G)\text{-}K]/(K+1)= \text{OPT}(G)\text{-}K/(K+1)$, implying that $|C| = \text{OPT}(G)$.

Minimum Steiner Tree

- No polynomial time approximation algorithm for the maximum clique problem can guarantee

  $|A(G,R)\text{-OPT}(G,R)|\leq K$

  for a fixed constant K, unless P=NP.
- Proof: Suppose A is such an algorithm. We will show that A can be used to derive a polynomial time optimization algorithm for Minimum Steiner Tree, contradicting the assumption that P≠NP.

Given as an instance any graph $G=(V, E)$ and $R\subseteq V$, the algorithm is to

- Construct a new graph by inserts K new vertices on each edge e of G, so that each e of G is chopped into K+1 *edges*.
  - $\text{OPT}(G',R)=1+(K+1)(\text{OPT}(G,R)\text{-}1)+K$,
- Run A on (G', R), producing a sterner tree T' of $A(G',R)\leq 1+(K+1)(\text{OPT}(G,R)\text{-}1)\text{-}K$ vertices.
- Obtain T by removing from T' the added vertices to recover the original edges.

T is a feasible Steiner Tree for G *and* R, and

$|T| \leq [(K+1)(\text{OPT}(G)\text{-}1)\text{-}K]/(K+1)+1= \text{OPT}(G)\text{-}K/(K+1)$, implying that $|T| = \text{OPT}(G)$.

Minimum Bin Pack

- If P≠NP, there no polynomial time approximation algorithm for Bin Pack to guarantee

  $A(G,W)/\text{OPT}(G,W)<2/3$.
- Proof: Suppose algorithm A can do so. We will show how A can be used to solve the NP-complete problem PARTITION. Given an arbitrary instance $A=\{a_1, a_2,\ldots, a_n\}$ of PARTITION, the next algorithm works for it.

- $B=(\sum_{a\in A} a)/2$
- Construct an instance $A'= \{b_1, b_2,\ldots, b_n\}$ for Bin Pack by letting each $b_i= a_i/B$.
- Call $A(A')$.
- Answer "yes" if $A(A')$ <3, and "no" otherwise.

This is correct because if A has an desired partition $\text{OPT}(A')=2$ and $A(A')<(2/3)\ \text{OPT}(A')=3$, otherwise $\text{OPT}(A')\geq 2$ and $A(A') \geq 3$.

Minimum Traveling Salesman

- No polynomial time approximation algorithm for TSP can guarantee

  $A(G, W)/\mathrm{OPT}(G, W) \le K$

  for a fixed constant K, unless P=NP.
- Proof: Suppose algorithm A can do so. We will show how A can be used to solve the NP-complete problem HC. Given an arbitrary instance graph $G=(V, E)$ of HC, the next algorithm decides whether G has an HC.

- Construct an instance of TSP by defining the weight function W as letting $W(u,v)=1$ if $\{u,v\} \in E$, and $K|V|$ otherwise.
- Call $A(G, W)$.
- Answer "yes" if $A(G, W) \le K|V|$, and "no" otherwise.

This is correct because $\mathrm{OPT}(G, W)=|V|$ if G has an HC, and $> K|V|$ otherwise.

2007/12/18 CS,OUC 5

Closing the Lecture

Summary

Comment on Assignment 2

FAQ

2007/12/18 CS,OUC 6

What have we done? A brief recall

Basic Concepts:

- Program are constrained by two factors: space and time.
- Time is the dominate factor.
- How to quantify the time so that it is an intrinsic property of an algorithm?
- What algorithm is efficient (good, practical)?

Algorithm Design:

- Reduction (transformation).
- Greedy
- Divide-and-conquer.
- Dynamic Programming.
- Data Structure.

Problem Analysis:

- Turing Reduction, Karp-Reduction
- NP-Completeness.
- NP-Hardness

Coping with NP-hard problems:

- Approximation algorithms
- Negative results.

2007/12/18 CS,OUC 7