Theory of Computation
Chapter 13: Approximability

Guan-Shieng Huang

Jan. 3, 2007

Decision v.s. Optimization Problems

decision problems: expect a "yes"/"no" answer

optimization problems: expect an optimal solution from all feasible solutions

When an optimization problem is proved to be NP-complete, the next step is

- to find useful heuristics
- to develop approximation algorithms
- to use randomness
- to invest on average-case analyses

Definition (optimization problem)

1. For each instance x there is a set of feasible solutions $F(x)$.
2. For each $y \in F(x)$, there is a positive integer $m(x, y)$, which measures the the cost (or benefit) of y.
3. $OPT(x) = m^*(x) = \min_{y \in F(x)} m(x, y)$ (minimization problem)
   $OPT(x) = m^*(x) = \max_{y \in F(x)} m(x, y)$ (maximization problem)

Definition (NPO)
NPO is the class of all optimization problems whose decision counterparts are in NP.

1. $y \in F(x) \Rightarrow |y| \leq |x|^k$ for some k;
2. whether $y \in F(x)$ can be determined in polynomial time;
3. $m(x, y)$ can be evaluated in poly. time.

Definition (Relative approximation)

x: an instance of an optimization problem P

y: any feasible solution of x

$$E(x, y) = \frac{|m^*(x) - m(x, y)|}{\max\{m^*(x), m(x, y)\}}$$

Remarks

1. $0 \leq E(x, y) \leq 1$;
2. $E(x, y) = 0$ when the solution is optimal;
3. $E(x, y) \to 1$ when the solution is very poor.

Definition (Performance ratio)

x: an instance of an optimization problem P

y: any feasible solution of x

$$R(x, y) = \max\left(\frac{m(x, y)}{m^*(x)}, \frac{m^*(x)}{m(x, y)}\right)$$

Remarks

1. $R(x, y) \geq 1$;
2. $R(x, y) = 1$ means that y is optimal;
3. $E(x, y) = 1 - \frac{1}{R(x,y)}$.

Definition (r-approximation)

$A(x)$: approximate solution of x for algorithm A

We say A is an r-approximation if

$$\forall_x R(x, A(x)) \leq r.$$

Remark

An r-approximation is also an r'-approximation if $r \leq r'$.

That is, the approximation becomes more difficult as r becomes smaller.

Definition (APX)

APX is the class of all NPO problems that have r-approximation algorithm for some constant r.

Definition (Polynomial-time approximation scheme)

P: NPO problem

We say A is a PTAS for P if

1. A has two parameters r and x where x's are instances of P;
2. when r is fixed to a constant with $r > 1$, $A(r, x)$ returns an r-approximate solution of x in polynomial time in $|x|$.

Remark

The time complexity of A could be

$$O(n^{\max\{\frac{1}{r-1},2\}}), O(n^5(r-1)^{-100}), O(n^5 2^{\frac{1}{r-1}})$$

where $n = |x|$. All of these are polynomial in n.

Definition (PTAS)
PTAS is the class of all NPO problems that admit a polynomial tome approximation scheme.

Definition (Fully polynomial-time approximation scheme)

1. A has two parameters r and x where x's are instances of P;
2. $A(r, x)$ returns an r-approximate solution of x in polynomial time both in $|x|$ and $\frac{1}{r-1}$
(since the approximation becomes more difficult when $r \to 1$).

Node Cover

Problem

Given a graph $G = (V, E)$, seek a smallest set of nodes $C \subseteq V$ such that for each edge E at least one of its endpoints is in C.

Greedy heuristic:

1. Let $C = \emptyset$.
2. While there are still edges left in G, choose the node in G with the largest degree, add it to C, and delete it from G.

However, the performance ratio is $\lg n$.

2-approximation algorithm

1. Let $C = \emptyset$.
2. While there are still edges left in G do
   (a) choose any edge (u, v);
   (b) add both u and v to C;
   (c) delete both u and v from G.

Theorem

This algorithm is a 2-approximation algorithm.

Proof. C contains $\frac{1}{2}|C|$ edges that share no common nodes. The optimum must contain at least one end points of these edges.

$$\therefore OPT(G) \geq \frac{1}{2}|C| \Rightarrow \frac{|C|}{OPT(G)} \leq 2.$$

Maximum Satisfiability

Problem (MAXSAT)
Given a set of clauses, find a truth assignment that satisfies the most of the clauses.

The following is a probabilistic argument that leads us to choose a good assignment.

1. If Φ has m clauses $C_1 \wedge C_2 \wedge \cdots \wedge C_m$, the expected number of satisfied clauses is

$$S(\Phi) = \sum_{i=1}^{m} \Pr[T \models C_i] \text{ where } T \text{ is a random assignment.}$$

2. However,

$$S(\Phi) = \frac{1}{2} \cdot S(\Phi|_{x_1=1}) + \frac{1}{2} \cdot S(\Phi|_{x_1=0}).$$

Hence at least one choice of $x_1 = t_1$ can make

$$S(\Phi) \leq S(\Phi|_{x_1=t_1}) \text{ where } t_i \in \{0,1\}.$$

3. We can continue this process for $i = 2, \ldots, n$, and finally

   $$S(\Phi) \leq S(\Phi|_{x_1=t_1}) \leq S(\Phi|_{x_1=t_1,x_2=t_2}) \leq \cdots \leq S(\Phi|_{x_1=t_1,\ldots,x_n=t_n}).$$

   That is, we get an assignment $\{x_1 = t_1, x_2 = t_2, \ldots, x_n = t_n\}$ that satisfies at least $S(\Phi)$ clauses.

4. If each C_i has at least k literals, we have

   $$\Pr_T[T \models C] = E[\text{C is satisfiable}] \geq 1 - \frac{1}{2^k}.$$

   $$\therefore S(\Phi) = \sum_{i=1}^{m} \Pr_T[T \models C_i] \geq m(1 - \frac{1}{2^k}).$$

   That is, we get an assignment that satisfies at least $m(1 - \frac{1}{2^k})$ clauses.

5. There are at most m clauses that can be satisfied (i.e. an upper bound for the optimum).

$$\therefore \text{performance ratio} \leq \frac{m}{m(1-\frac{1}{2^k})} = 1 + \frac{1}{2^k - 1}.$$

6. Since k is always at least 1, the above algorithm is a 2-approximation algorithm for MAXSAT.

Maximum Cut

Problem (MAX-CUT)
Given a graph $G = (V, E)$, partition V into two sets S and $V - S$ such that there are as many edges as possible between S and $V - S$.

Algorithm based on local improvement

1. Start from any partition S.
2. If the cut can be made large by
   - adding a single node to S, or by
   - removing a single node from S, then do so;

   Until no improvement is possible.

Theorem This is a 2-approximation algorithm.

Proof.

1. Decompose V into four parts: $V = V_1 \cup V_2 \cup V_3 \cup V_4$ such that our heuristic is $(V_1 \cup V_2, V_3 \cup V_4)$ where as the optimum is $(V_1 \cup V_3, V_2 \cup V_4)$.

2. Let e_{ij} be the number of edges between V_i and V_j for $1 \leq i \leq j \leq 4$.

3. Then we want to bound

$$\frac{e_{12} + e_{14} + e_{23} + e_{34}}{e_{13} + e_{14} + e_{23} + e_{24}}$$

by a constant.

4.

$$2e_{11} + e_{12} \leq e_{13} + e_{14} \Rightarrow e_{12} \leq e_{13} + e_{14};$$

$$e_{12} \leq e_{23} + e_{24};$$

$$e_{34} \leq e_{23} + e_{13};$$

$$e_{34} \leq e_{14} + e_{24}.$$

5.

$$\therefore e_{12} + e_{34} \leq e_{13} + e_{14} + e_{23} + e_{24};$$

$$e_{14} + e_{23} \leq e_{13} + e_{14} + e_{23} + e_{24}.$$

6.

$$\therefore e_{12} + e_{14} + e_{23} + e_{34} \leq 2(e_{13} + e_{14} + e_{23} + e_{24}).$$

Therefore, the performance ratio is bounded above by 2.

Traveling Salesman Problem

Theorem Unless $P = NP$, there is no constant performance ratio for TSP. (That is, TSP $\notin$ APX unless $P = NP$.)

Proof. Suppose TSP is c-approximable for some constant c. Then we can solve Hamilton Cycle in polynomial time.

1. Given any graph $G = (V, E)$, assign

$$d(i,j) = \begin{cases} 1 & \text{if } (i,j) \in E \\ c|V| & \text{if } (i,j) \notin E \end{cases}$$

2. If there is a c-approximation that can solve this instance in polynomial time, we can determine whether G has an HC in poly. time.

3. Suppose G has an HC. Then the approximation algorithm returns a solution with total distance at most $c|V|$, which

means it cannot include any $(i, j) \notin E$.

Remark There is a $\frac{3}{2}$-approximation algorithm for TSP when its distance satisfies the triangle inequality $d(i, j) + d(j, k) \leq d(i, k)$.

Knapsack

Problem Given n weights w_i, $1, \ldots, n$, a weight limit $\mathcal{W}$, and n values v_i, $i = 1, \ldots, n$, find a subset $S \subseteq \{1, 2, \ldots, n\}$ such that $\sum_{i \in S} w_i \leq \mathcal{W}$ and $\sum_{i \in S} v_i$ is maximum.

Pseudopolynomial algorithm

$V(w,i)$: the largest value from the first i items so that their total weight is $\leq w$

$$
\begin{aligned}
V(w,i) &= \max\{V(w,i-1), V(w-w_i,i-1)+v_i\} \\
V(w,0) &= 0
\end{aligned}
$$

The time complexity is $O(n\mathcal{W})$.

Another algorithm

1. Let $\mathcal{V} = \max\{v_1, v_2, \ldots, v_n\}$.
2. Define $W(i, v)$ to be the minimum weight from the first i items so that their total value is $\mathcal{V}$.
3.

$$\begin{aligned} W(i, v) &= \min\{W(i-1, v), W(i-1, v-v_i) + w_i\} \\ W(0, 0) &= 0 \\ W(0, v) &= \infty \text{ if } v > 0. \end{aligned}$$

Time complexity is $O(n^2\mathcal{V})$ since $1 \leq i \leq n$ and $0 \leq v \leq n\mathcal{V}$.

Approximation algorithm

Given $x = (w_1, \ldots, w_n, \mathcal{W}, v_1, \ldots, v_n)$, construct $x' = (w_1, \ldots, w_n, \mathcal{W}, v'_1, \ldots, v'_n)$ where $v'_i = 2^b \cdot \lfloor \frac{v_i}{2^b} \rfloor$ for some parameter b. We can find optimal solution for x' in time $O(\frac{n^2 \mathcal{V}}{2^b})$, using it as an approximate solution for x.

Theorem The above approximation algorithm is a polynomial-time approximation scheme.

(In fact, it is an FPTAS.)

Proof.

$$\sum_{i \in S} v_i \geq \sum_{i \in S'} v_i \geq \sum_{i \in S'} v'_i \geq \sum_{i \in S} v'_i \geq \sum_{i \in S} v_i - n2^b.$$

S: optimal for x; S': optimal for x'

Performance ratio

$$\frac{\sum_{i\in S} v_i}{\sum_{i\in S'} v_i} \le \frac{\sum_{i\in S} v_i}{\sum_{i\in S} v_i - n2^b} = \frac{1}{1-\frac{n2^b}{\sum_{i\in S} v_i}} \le \frac{1}{1-\frac{n2^b}{\mathcal{V}}} \le \frac{1}{1-\epsilon}$$

by setting $b = \lceil \lg \frac{\epsilon\mathcal{V}}{n} \rceil$.

Time complexity becomes $O(\frac{n^2\mathcal{V}}{2^b}) = O(\frac{n^3}{\epsilon})$.

$\therefore$ performance ratio $= \frac{1}{1-\epsilon}$, which can be arbitrarily close to 1.

Approximation Preserving Reductions

L-reduction $(A \leq_L B)$

A, B: two optimization problems

f: a function from instances of A to instances of B

g: a function from feasible solutions of $f(x)$ to feasible solutions of x

(f, g) is called an L-reduction iff

1. f and g are computable in logarithmic space;

2. there exists constant α such that

$$OPT(f(x)) \leq \alpha \cdot OPT(x)$$

for all instances x of A;

3. there exists constant β such that

$$|OPT(x) - m_A(x, g(s))| \leq \beta \cdot |OPT(f(x)) - m_B(f(x), s)|$$

where s is any feasible solution of $f(x)$.

Remark

- L-reductions are transitive. ($A \leq_L B$ and $B \leq_L C \Rightarrow A \leq_L C$.)
- If there is an L-reduction from A to B and $B \in APX$, then we have $A \in APX$.
- L-reductions are closed in APX, $PTAS$, and $FPTAS$.

AP-reduction $A \leq_{AP} B$

A, B: two optimization problems

f: a function $I_A \times (1, \infty) \to I_B$
(I_A: instances of A; I_B: instances of B)

g: a function $I_A \times F_B \times (1, \infty) \to F_A$
(F_A: feasible solutions for A; F_B: feasible solutions for B)

(R, S) is called an AP-reduction iff

1. $F_B(f(x, r)) \neq \emptyset$ if $F_A(x) \neq \emptyset$ for all $x \in I_A$ and $r > 1$;
   (x has solutions implies $f(x, r)$ has solutions)
2. $g(x, y, r) \in F_A(x)$ for any $x \in I_A$, $y \in F_B(f(x, r))$ and $r > 1$;
   (the solution for $f(x, r)$ can be sent back to be one for x by g)
3. f and g are computable in logarithmic space for any fixed rational $r > 1$;

4. there exists constant α such that $R_A(x, g(x, y, r)) \leq 1 + \alpha(r - 1)$ whenever $R_B(f(x, r), y) \leq r$ for all $x \in I_A$, $y \in F_B(f(x, r))$ and $r > 1$.
(the performance ratio for B is preserved in A by (f, g))

Theorem Let $A \in APX$. If $A \leq_L B$, then $A \leq_{AP} B$.
(That is, AP-reducibility is more general than L-reducibility.)

Theorem MAX3SAT is APX-complete under AP-reducibility.

Remarks

- APX-completeness (under AP-reductions) is built by the PCP-characterization of NP.
- L-reducibility builds MAXSNP-completeness.