Approximation Basics*

Milestones, Concepts, and Examples

Xiaofeng Gao

Department of Computer Science and Engineering
Shanghai Jiao Tong University, P.R.China

Fall 2012

* Special thanks is given to Dr. Guoqiang Li for sharing his teaching materials.

Outline

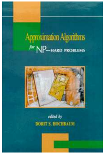
History of Approximation

1966	**Graham**: First analyzed algorithms by approximation ratio
1971	**Cook**: Gave the concepts of NP-Completeness
1972	**Karp**: Introduced plenty NP-Hard combinatorial optimization problems
1970's	Approximation became a popular research area
1979	**Garey & Johnson**: Computers and Intractability: A guide to the Theory of NP-Completeness

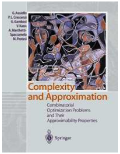
Books

CS 351 Stanford Univ

(1991-1992) Rajeev Motwani
Lecture Notes on Approximation Algorithms Volumn I

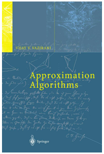


(1997) Hochbaum (Editor)
Approximation Algorithms for NP-Hard Problems

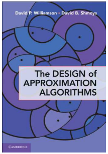


(1999) Ausiello, Crescenizi, Gambosi, etc.
Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties

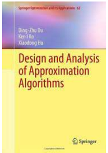
Books (2)



(2001) Vijay V. Vazirani
Approximation Algorithms



(2010) D.P. Williamson & D.B. Shmoys
The Design of Approximation Algorithms



(2012) D.Z Du, K-I. Ko & X.D. Hu
Design and Analysis of Approximation Algorithms

Hardness

There are not much hardness results until 1990s...

Theorem (PCP Theorem, ALMSS'92)

There is no PTAS for MAX-3SAT unless P = NP

ALMSS: Arora, Lund, Motwani, Sudan, and Szegedy

Conjecture (Unique Games Conjecture, Knot'02)

The Unique Game is NP-hard to approximate for any constant ratio.

Subhash Khot

Outline

NP Optimization Problem

Definition

A NP Optimization Problem P is a fourtuple $(I, sol, m, goal)$ s.t.

- I is the set of the instances of A and is recognizable in polynomial time.
- Given an instance x of I, $sol(x)$ is the set of short feasible solutions of x and $\forall x$ and $\forall y$ such that $|y| \leq p(|x|)$, it is decidable in polynomial time whether $y \in sol(x)$.
- Given an instance x and a feasible solution y of x, $m(x, y)$ is a polynomial time computable measure function providing a positive integer which is the value of y.
- $goal \in \{\max, \min\}$ denotes maximization or minimization.

An Example of NP Optimization Problem

Example

Given a graph $G = (V, E)$, the Minimum Vertex Cover problem (MVC) is to find a vertex cover of minimum size, that is, a minimum node subset $U \subseteq V$ such that, for each edge $(v_i, v_j) \in E$, either $v_i \in U$ or $v_j \in U$.

Justification → MVC is an NP Optimization Problem

- $I = \{G = (V, E) | G \text{ is a graph}\}$; *poly-time decidable*
- $sol(G) = \{U \subseteq V | \forall (v_i, v_j) \in E [v_i \in U \vee v_j \in U]\}$; *short feasible solution set and poly-time decidable*
- $m(G, U) = |U|$; *poly-time computable function*
- $goal = \min$.

NPO Class

Definition (NPO Class)

The class NPO is the set of all NP optimization problems.

Definition (Goal of NPO Problem)

The goal of an NPO problem with respect to an instance x is to find an *optimum solution*, that is, a feasible solution y such that $m(x,y) = goal\{m(x,y') : y' \in sol(x)\}$.

What is Approximation Algorithm?

Definition

Given an NP optimization problem $A = (I, sol, m, goal)$, an algorithm A is an approximation algorithm for P if, for any given instance $x \in I$, it returns an approximate solution, that is a feasible solution $A(x) \in sol(x)$ with guaranteed quality.

- Guaranteed quality is the difference between approximation and heuristics.
- Approximation for PO, NPO and NP-hard Optimization.
- Decision, Optimization, and Constructive Problems.

r-Approximation

Definition (Approximation Ratio)

Let P be an NPO problem. Given an instance x and a feasible solution y of x, we define the performance ratio of y with respect to x as

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

Definition (r-Approximation)

Given an optimization problem P and an approximation algorithm A for P, A is said to be an r-approximation for P if, given any input instance x of P, the performance ratio of the approximate solution $A(x)$ is bounded by r, say, $R(x, A(x)) \leq r$.

APX Class

Definition (F-APX)

Given a class of functions F, an NPO problem P belongs to the class F-APX if an r-approximation polynomial time algorithm A for P exists, for some function $r \in F$.

Example

- F is constant functions $\rightarrow P \in$ APX.
- F is $O(\log n)$ functions $\rightarrow P \in$ log-APX.
- F is $O(n^k)$ functions (polynomials) $\rightarrow p \in$ poly-APX.
- F is $O(2^{n^k})$ functions $\rightarrow P \in$ exp-APX.

Special Case

Definition (Polynomial Time Approximation Scheme → PTAS)

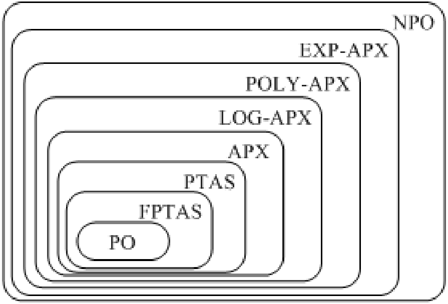
An NPO problem P belongs to the class PTAS if an algorithm A exists such that, for any rational value $\epsilon > 0$, when applied A to input (x, ϵ), it returns an $(1 + \epsilon)$-approximate solution of x in time polynomial in $|x|$.

Definition (Fully PTAS → FPTAS)

An NPO problem P belongs to the class FPTAS if an algorithm A exists such that, for any rational value $\epsilon > 0$, when applied A to input (x, ϵ), it returns a $(1 + \epsilon)$-approximate solution of x in time polynomial both in $|x|$ and in $\frac{1}{\epsilon}$.

Approximation Class Inclusion

If $P \neq NP$, then FPTAS $\subseteq$ PTAS $\subseteq$ APX $\subseteq$ Log-APX $\subseteq$ Poly-APX $\subseteq$ Exp-APX $\subseteq$ NPO



- Constant-Factor Approximation (APX)
  - Reduce App. Ratio
  - Reduce Time Complexity
- PTAS ($(1+\epsilon)$-Appx)
  - Test Existence
  - Reduce Time Complexity

Outline

Set Cover Problem

Problem

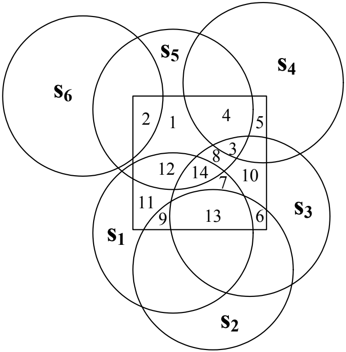
Instance: *Given a ground set of elements* $E = \{e_1, e_2, \cdots, e_n\}$, *subsets of elements* $\mathcal{S} = \{S_1, \cdots, S_m\}$ *where each* $S_j \subseteq E$, *and a nonnegative weight* $w_j \geq 0$ *for each subset* S_j.

Solution: *A subset* $I \subseteq \{1, 2, \cdots, m\}$ *such that* $\bigcup_{j \in I} S_j = E$.

Measure: $\sum_{j \in I} w_j$.

If $w_j = 1$ for each subset S_j, then it is unweighted set cover.

Application in Networking



Definition (Sensor Coverage)

Given a target region with sensor set $\mathcal{S} = \{S_1, ..., S_k\}$, find a minimum subset $\mathcal{R}$ of $\mathcal{S}$ to cover all the target region.

$S_1 = \{7, 9, 11, 12, 13, 14\}$,
$S_2 = \{6, 9, 13\}$,
$S_3 = \{3, 6, 7, 8, 10, 13, 14\}$,
$S_4 = \{3, 4, 5\}$,
$S_5 = \{2, 1, 3, 4, 8, 12, 14\}$ and
$S_6 = \{2\}$.
Find $\{S_i\}$ to cover $T = \{1, \cdots, 14\}$.

The Goal of Greedy Algorithm

- Given:
  - An instance of the problem specifies a set of items
- Goal:
  - Determine a subset of the items that satisfies the problem constraints
  - Maximize or minimize the measure function
- Steps:
  - Sort the items according to some criterion
  - Incrementally build the solution starting from the empty set
  - Consider items one at a time, and maintain a set of “selected" items
  - Terminate when break the problem constraints

Greedy Algorithm of Set Cover

Algorithm 1 Greedy Set Cover

Input: $E, \mathcal{S}, W$.

Output: Subset $I \subseteq \{1, 2, \cdots, m\}$ such that $\bigcup_{j \in I} S_j = E$.

```
I = ∅;
∀j : Ŝ_j = S_j;          ▷ Ŝ_j: compute average remaining weight
while E ≠ ∅ do
    i = arg min_{j: Ŝ_j ≠ ∅} w_j / |Ŝ_j|;
    I = I ∪ {i};
    E = E \ S_i;
    ∀j : Ŝ_j = Ŝ_j \ S_i;
end while
Return I.
```

Time Complexity

Theorem

Greedy Set Cover has time complexity $O(m^2)$.

Proof.

(1) There cannot be more than m round.

(2) In each round we compute $O(m)$ ratios, each in constant time.

Thus the total running time is $O(m^2)$. □

Preliminaries

Definition (Harmonic Number)

$$H_k = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{k}. \qquad H_k \approx \ln k$$

Theorem

Given positive numbers $a_1, \cdots, a_k$ and $b_1, \cdots, b_k$, then

$$\min_{i=1,\cdots,k} \frac{a_i}{b_i} \le \frac{\sum_{i=1}^k a_i}{\sum_{i=1}^k b_i} \le \max_{i=1,\cdots,k} \frac{a_i}{b_i}.$$

Approximation Ratio

Theorem

Greedy Set Cover is an H_n-approximation.

Proof.

(1) Consider the weight of each element.

(2) Consider the weight of each iteration.

(3) Combination and Relaxation.

□

Notations

OPT: The weight of the optimal solution.

O: The indices of sets in an optimal solution.

n_k: The number of elements that remain uncovered at the start of the kth iteration.

ℓ: Algorithm terminates in ℓ iterations. $n_1 = n$, $n_{\ell+1} = 0$.

I_k: Indices of sets chosen in iterations 1 to $k - 1$.

$\widehat{S}_j$: The set of uncovered elements in S_j at the start of the kth iteration. $\widehat{S}_j = S_j - \bigcup_{p \in I_k} S_p$

One Iteration

Lemma

For the set j chosen in the kth iteration, $w_j \leq \frac{n_k - n_{k+1}}{n_k} OPT$.

Proof.

$$\min_{j:\widehat{S}_j \neq \varnothing} \frac{w_j}{|\widehat{S}_j|} \leq \frac{\sum_{i \in O} w_i}{\sum_{i \in O} |\widehat{S}_i|} = \frac{OPT}{\sum_{i \in O} |\widehat{S}_i|} \leq \frac{OPT}{n_k}.$$

Let j be the chosen set, if we add S_j into our solution, then there will be $|\widehat{S}_j|$ fewer uncovered elements, so $n_{k+1} = n_k - |\widehat{S}_j|$. Thus

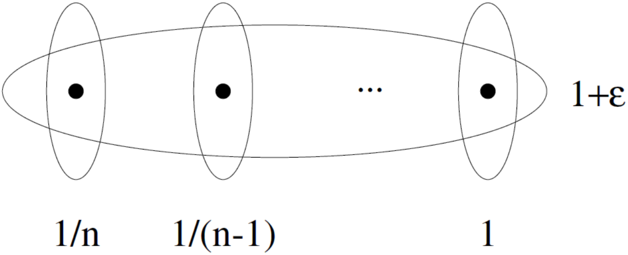
$$w_j \leq \frac{|\widehat{S}_j| OPT}{n_k} = \frac{n_k - n_{k+1}}{n_k} OPT.$$

□

Merging Together

$$
\begin{aligned}
\sum_{j \in I} w_j &\leq \sum_{k=1}^{\ell} \frac{n_k - n_{k+1}}{n_k} OPT \\
&\leq OPT \cdot \sum_{k=1}^{\ell} \left(\frac{1}{n_k} + \frac{1}{n_k - 1} + \cdots + \frac{1}{n_{k+1} + 1} \right) \\
&= OPT \cdot \sum_{i=1}^{n} \frac{1}{i} \\
&= H_n \cdot OPT.
\end{aligned}
$$

Tight Example



(1) Greedy solution: $\frac{1}{n} + \frac{1}{n-1} + \cdots + 1 = H_n$;

(2) OPT solution: $1 + \epsilon$.

The Goal

- Since a LP can be solved in polynomial time, given a hard combinatorial optimization problem P, we don't expect to a LP formulation s.t. for any instance $x \in P$, the number of constraints of the LP is polynomial in size of x (this would imply P=NP!!)
- LP can be used as a computational step in the design of approximation algorithm.
  - Integer Linear Programming (ILP)
  - Primal-Dual Algorithm

The Steps of Programming and Rounding

Given: An instance of the problem specifies a set of items

Goal: Maximize or minimize the measure function

Steps:

- Construct an Integer Program (IP) for discrete optimization problem $\rightarrow$ OPT solution.
- Relax IP to Linear Program (LP) $\rightarrow$ A polynomial-solvable solution (may not be feasible).
  - Every feasible solution for IP is feasible for LP;
  - The value of any feasible solution for IP has the same value in LP.
- Round LP solution to a feasible solution.

Integer Program

Define $x_j = \begin{cases} 1 & \text{If we select the index of } S_j \text{ into } I. \\ 0 & \text{otherwise.} \end{cases}$

The Integer Program $IP(S)$ can be formulated as:

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{m} w_j x_j \\ \text{subject to} \quad & \sum_{j: e_i \in S_j} x_j \geq 1, \quad i = 1, \cdots, n \\ & x_j \in \{0, 1\}, \quad j = 1, \cdots, m \end{aligned}$$

Relaxation

The relaxed Linear Program $LP(S)$ can be formulated as:

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{m} w_j x_j \\ \text{subject to} \quad & \sum_{j:e_i \in S_j} x_j \geq 1, \quad i = 1, \cdots, n \\ & x_j \geq 0, \quad j = 1, \cdots, m \end{aligned}$$

Define Z_{IP}^* as the optimum value of $IP(S)$, Z_{LP}^* the optimum value of $LP(S)$, then we have

$$Z_{LP}^* \leq Z_{IP}^* = OPT.$$

Deterministic Rounding

Let $f = \max_{i=1,\cdots,n} f_i, \ (f_i = |\{j : e_i \in S_j\}|)$.

x^* the optimal solution of $LP(S)$.

Algorithm 2 Deterministic Rounding for $LP(S)$

Input: x^*, f.

Output: $\widehat{x}$ for $IP(S)$.

1: **for** $j = 1$ to n **do**

2: $$\widehat{x_j} = \begin{cases} 1 & \text{If } x_j^* \geq \frac{1}{f}. \\ 0 & \text{otherwise.} \end{cases}$$

3: **end for**

4: Return $\widehat{x}$.

Feasible Solution

Lemma

LP rounding outputs a feasible solution for $IP(S)$.

Proof.

We call e_i is covered if $\exists j$, s.t. $e_i \in S_j$ and $x_j = 1$ in $\widehat{x}$.

Since x^* is feasible to $LP(S)$, $\sum_{j: e_i \in S_j} x_j^* \geq 1$ for e_i.

By the definition of f and f_i, there are $f_i \leq f$ terms in the sum, so at lease one term must be at least $\frac{1}{f}$.

Thus $\exists j$ such that $e_i \in S_j$ and $x_j^* \geq \frac{1}{f}$. $x_j = 1$ in $\widehat{x}$. □

Approximation Ratio

Theorem

LP(S)+rounding is an f-approximation for Set Cover problem.

Proof.

$f \cdot x_j^* \geq 1$ for each $j \in I$ and $fw_j x_j^* \geq 0$ for $j = 1, \cdots, m$.

$$\begin{aligned} \sum_{j \in I} w_j &\leq \sum_{j=1}^{m} w_j \cdot (f \cdot x_j^*) \\ &= f \sum_{j=1}^{m} w_j x_j^* \\ &= f \cdot Z_{LP}^* \\ &\leq f \cdot OPT. \end{aligned}$$