Approximation & Complexity

Ioannis Psarros

National & Kapodistrian University of Athens

April 11, 2014

Definitions

Definition

Let A an optimization problem.

- *For each instance x we have a set of feasible solutions $F(x)$.*
- *For each $s \in F(x)$ we have a positive integer cost $c(s)$.*
- *The optimum cost is defined as $\mathrm{OPT}(x) = \min_{s \in F(x)} c(s)$ (or $\max_{s \in F(x)} c(s)$).*

Definitions

Definition (Minimization)

Let M an algorithm which returns $M(x) \in F(x)$. M is an ρ-approximation algorithm, where $\rho > 1$, if for all x we have,

$$\frac{c(M(x))}{\mathrm{OPT}(x)} \leq \rho.$$

Definition (Maximization)

Let M an algorithm which returns $M(x) \in F(x)$. M is an ρ-approximation algorithm, where $0 < \rho < 1$, if for all x we have,

$$\frac{c(M(x))}{\mathrm{OPT}(x)} \geq \rho.$$

MAXSAT

Definition (MAXSAT)

Given a set of m clauses in n boolean variables, find the truth assignment that satisfies the most.

Consider the following randomized algorithm:

- Set each Boolean variable to be *true* independently with probability $1/2$.
- Return the resulting truth assignment.

MAXSAT

Consider a clause c_i with k_i literals. The probability $p(c_i)$ that this clause is satisfied is $1 - \frac{1}{2^{k_i}}$.
Hence,

$$\mathbb{E}(N) = \sum_{i=1}^{m} p(c_i) \geq \frac{1}{2} m \geq \frac{1}{2} \mathrm{OPT}$$

where N denotes the number of satisfied clauses.
Can we do it deterministically?

MAXSAT

The following holds:

$$\mathbb{E}(N) = \frac{1}{2}(\mathbb{E}(N|x_1 = \mathit{true}) + \mathbb{E}(N|x_1 = \mathit{false})).$$

So, deterministically assign to the next variable the value that maximizes the expectation.

Theorem

There exists a polynomial time deterministic algorithm with approximation factor $1/2$ *for the MAXSAT problem.*

The above is a general method for derandomizing known as the method of conditional expectation.

L-reductions

Ordinary reductions are inadequate for studying approximability.

Definition

Let A and B two optimization problems. An L-reduction from A to B is a pair of functions R and S, both computed in logarithmic space, with following two additional properties:

- *If x an instance of A and $R(x)$ an instance of B then:*

  $$\mathrm{OPT}(R(x)) \leq \alpha \cdot \mathrm{OPT}(x),$$

  where $\alpha > 0$.

- *If s feasible solution of $R(x)$ then $S(s)$ is a feasible solution of x s.t.*

  $$|\mathrm{OPT}(x) - c(S(s))| \leq \beta \cdot |\mathrm{OPT}(R(x)) - c(s)|,$$

  where $\beta > 0$.

Properties

Proposition

If (R, S) is an L-reduction from problem A to problem B and (R', S') is an L-reduction from problem B to problem C, then their composition $(R \cdot R', S' \cdot S)$ is an L-reduction from A to C.

Proposition

If there is an L-reduction (R, S) from A to B with constants α, β and there is a polynomial-time $(1+\epsilon)$-approximation algorithm for B, then there is a polynomial-time $(1 \pm \alpha\beta\epsilon)$-approximation algorithm for A.

Given an instance x of A apply the $(1+\epsilon)$-approx algorithm to the instance $R(x)$ of B. We obtain solution s and we return $S(s)$.

The class SNP

Fagin's theorem states that all graph theoretic properties in NP can be expressed in existential second-order logic.

Definition

SNP Strict NP or SNP consists of all properties expressible as

$$\exists S \forall x_1 \forall x_2 \cdots \forall x_k \phi(S, P, x_1, \cdots, x_k),$$

where ϕ is a quantifier-free First-Order expression and P predicates (the input).

But SNP contains decision problems..

The class MAXSNP

Definition

Define $MAXSNP_0$ to be the class of optimization problems expressed as

$$\max_S |\{(x_1, \cdots, x_k)\} \in U^k : \phi(P_1, \cdots P_m, S, x_1, \cdots, x_k)\}|,$$

where U is a finite universe and $P_1, \cdots, P_m, S$ predicates.

Definition

MAXSNP is the class of optimization problems that are L-reducible to a problem in $MAXSNP_0$.

The class MAXSNP

Example

MAX-CUT is in MAXSNP_0 and therefore in MAXSNP. It can be written as follows:

$$\max_S |\{(x,y) : ((G(x,y) \vee G(y,x)) \wedge S(x) \wedge \neg S(y))\}|.$$

The class MAXSNP

Example

MAX2SAT is in MAXSNP_0 and therefore in MAXSNP. Let P_0, P_1, P_2 predicates s.t.:

- $P_0(x, y) \Leftrightarrow x \vee y$ is a clause.
- $P_1(x, y) \Leftrightarrow \neg x \vee y$ is a clause.
- $P_2(x, y) \Leftrightarrow \neg x \vee \neg y$ is a clause.

MAX2SAT can be written as

$$\max_S |\{(x, y) : \phi(P_0, P_1, P_2, S, x, y)\}|,$$

where ϕ is the following expression:

$$(P_0(x, y) \wedge (S(x) \vee S(y))) \vee (P_1(x, y) \wedge (\neg S(x) \vee S(y))) \vee$$

$$\vee (P_2(x, y) \wedge (\neg S(x) \vee \neg S(y))).$$

MAXSNP-Completeness

Definition

A problem in MAXSNP is MAXSNP-complete if all problems in MAXSNP L-reduce to it.

Theorem

MAX3SAT is MAXSNP-complete.

Proof.

It suffices to show that all problems in MAXSNP_0 can be L-reduced to MAX3SAT. Consider a problem $A \in \text{MAXSNP}_0$ which is defined by the expression:

$$\max_S |\{(x_1, \cdots, x_k) : \phi\}|.$$

MAXSNP-Completeness

Proof(Cont.)

- For each k-tuple $y \in U^k$ substitute for $(x_1, \cdots, x_k)$ in ϕ and obtain ϕ_y.
- ϕ_y contains atomic expressions that uses P_i and S. Evaluate atomic expressions that use P_i.
- ϕ_y now consists of atomic expressions of the form $S(y_{i_1}, \cdots, y_{i_r})$.
- k is independent of the input $\implies$ ϕ_y can be transformed into an equivalent 3CNF expression ϕ'_y of constant size.

MAXSNP-Completeness

Proof(Cont.)

Each satisfiable 3CNF expression ϕ'_y consists of at most c clauses, where c depends on ϕ. Hence,

$$\mathrm{OPT}(R(x)) \leq c \cdot m,$$

where m the number of satisfiable expressions ϕ_y.
We can also see that

$$\mathrm{OPT}(x) \geq 2^{-k} m.$$

Hence,

$$\mathrm{OPT}(x) \leq 2^k c \cdot \mathrm{OPT}(x).$$

The first condition is satisfied for $\alpha = 2^k c$.

MAXSNP-Completeness

Proof(Cont.)

Second condition is also satisfied for $\beta = 1$. We can lift the cost function for MAX3SAT s.t. the number of unsatisfied clauses equals the number of unsatisfied expressions ϕ_y. In other words,

$$|\mathrm{OPT}(x) - c(S(s))| \leq |\mathrm{OPT}(R(x)) - c(s)|.$$

□

PTAS-FPTAS

Definition

An optimization problem has a polynomial time approximation scheme (PTAS) if there exists $(1 \pm \epsilon)$*-approximation algorithm for any* $\epsilon > 0$ *and running time bounded by a polynomial in the size of the input.*

Definition

An optimization problem has a fully polynomial time approximation scheme (FPTAS) if there exists $(1 \pm \epsilon)$*-approximation algorithm for any* $\epsilon > 0$ *and running time bounded by a polynomial in the size of the input and* $1/\epsilon$*.*

Essentialy, FPTAS is the best we can hope for an NP-hard optimization problem.

FPTAS for the knapshack problem

The knapshack problem admits a pseudo-polynomial algorithm with running time $O(n^2 P)$ where P is the profit of the most valuable object.
What if P is bounded by a polynomial in n?
An FPTAS for the knapshack problem:

- Given $\epsilon > 0$, let $K = \frac{\epsilon P}{n}$.
- For each object a_i define profit $profit'(a_i) = \lfloor \frac{profit(a_i)}{K} \rfloor$.
- Using the dynamic programming algorithm, find the best solution S' for the new set of profits.

FPTAS for the knapshack problem

Lemma

Let A the output of our algorithm. Then,

$$\mathit{profit}(A) \geq (1-\epsilon)\mathrm{OPT}.$$

Proof

Let O the set that gives the optimum solution.

$$\mathit{profit}(O) - K \cdot \mathit{profit}'(O) \leq nK$$

$$\mathit{profit}(S') \geq K \cdot \mathit{profit}'(O) \geq \mathit{profit}(O) - nK = OPT - \epsilon P \geq (1-\epsilon)OPT$$

The running time is $O(n^2 \lfloor \frac{P}{K} \rfloor) = O(\frac{n^3}{\epsilon})$.

FPRAS

Definition

Consider a problem in P whose counting version f is #P-complete. An algorithm A is a fully polynomial randomized approximation scheme (FPRAS) if for each instance $x \in \Sigma^*$ and error parameter $\epsilon > 0$,

$$Pr[|A(x) - f(x)| \leq \epsilon f(x)] \geq \frac{3}{4}$$

and the running time of A is polynomial in $|x|$ and $1/\epsilon$.

Counting DNF Solutions

Problem

Let $f = C_1 \vee C_2 \vee \cdots \vee C_m$ be a formula in disjunctive normal form on n Boolean variables $x_1, \cdots, x_n$. Compute $\#f$, the number of satisfying truth assignments of f.

Let S_i be the set of truth assignments that satisfy C_i. Clearly $|S_i| = 2^{n-r_i}$ where r_i the number of literals in C_i.Let M be the multiset union of all S_i.Let $c(\tau)$ be the number of clauses that τ satisfies. Pick a satisfying truth assignment, τ , for f with probability $c(\tau)/|M|$ and define $X(\tau) = |M|/c(\tau)$.

Counting DNF Solutions

Pick at random a satisfying truth assignment, τ, for f with probability $c(\tau)/|M|$:

- First pick a clause so that the probability of picking clause C_i is $|S_i|/|M|$.
- Next, among the truth assignments satisfying the picked clause, pick one at random.

$$Pr[\tau \text{ is picked}] = \sum_{i:\tau \text{ satisfies } C_i} \frac{|S_i|}{|M|} \cdot \frac{1}{|S_i|} = \frac{c(\tau)}{|M|}$$

$$\mathbb{E}[X] = \sum_{\tau} Pr[\tau \text{ is picked}] \cdot X(\tau) = \sum_{\tau \text{ satisfies } f} \frac{c(\tau)}{|M|} \cdot \frac{|M|}{c(\tau)} = \#f.$$

Counting DNF Solutions

Luckily,

$$\frac{\sigma(X)}{\mathbb{E}[X]} \leq m - 1.$$

Sampling X polynomially many times (in n and $1/\epsilon$) and simply outputting the mean leads to an FPRAS for $\#f$.
In particular, if we set $k = 4(m-1)^2/\epsilon^2$, the following holds (by Chebyshev's inequality)

$$Pr[|X_k - \mathbb{E}[X_k]| \geq \epsilon\mathbb{E}[X_k]] \leq (\frac{\sigma(X_k)}{\epsilon\mathbb{E}[X_k]})^2 = (\frac{\sigma(X)}{\epsilon\sqrt{k}\mathbb{E}[X]})^2 \leq \frac{1}{4}.$$

Thank You!