Fundamentals of optimization problems

Dmitriy Serdyuk

Ferienakademie in Sarntal 2012
FAU Erlangen-Nürnberg, TU München, Uni Stuttgart

September 2012

Overview

1. Introduction
   - Optimization problems
   - PO and NPO problems
   - NP-hard optimization problems
2. Approximate solutions
   - Definitions
   - Absolute approximation
   - Relative approximation
   - Limits to approximability: The gap technique
3. Polynomial time approximation schemes
   - Definitions
   - APX versus PTAS
4. Fully polynomial time approximation schemes
   - Definitions
   - Negative results for the class FPTAS

Overview

What is optimization problem

Definition

Optimization problem $\mathcal{P}$: $(I_\mathcal{P}, SOL_\mathcal{P}, m_\mathcal{P}, goal_\mathcal{P})$ where

1. $I_\mathcal{P}$ – set of instances of the problem $\mathcal{P}$
2. $SOL_\mathcal{P}$ – is a function that associates to any input instance $x \in I_\mathcal{P}$ the set of feasible solutions of x
3. $m_\mathcal{P}$ – is a measure function that for every pair $(x, y)(x \in I_\mathcal{P}$ and $y \in SOL_\mathcal{P}(x))$ associates positive integer which is the value of the feasible solution y
4. $goal_\mathcal{P} \in \{MIN, MAX\}$ – specifies whether $\mathcal{P}$ is a maximization or a minimization problem

Optimal solutions

- $SOL^*_{\mathcal{P}}$ – set of optimal solutions of x
- $m_{\mathcal{P}}(x, y^*) = goal_{\mathcal{P}}\{v | v = m_{\mathcal{P}}(x, z) \land z \in SOL_{\mathcal{P}}(x)\}$
- The value of any optimal solution y^* of x will be denoted as $m^*_{\mathcal{P}}(x)$

Example: Minimum Vertex Cover

Given a graph $G = (V, E)$, the Minimum Vertex Cover problem is to find a vertex cover of minimum size. Formally:

- $I = \{G = (V, E) | G \text{ is a graph}\}$
- $SOL(G) = \{U \subseteq V | \forall (v_i, v_j) \in E : v_i \in U \vee v_j \in U\}$
- $m(G, U) = |U|$
- goal = MIN

Problems associated with optimization problem

Definition

- *Constructive problem* $\mathcal{P}_C$ – given an instance $x \in I_{\mathcal{P}}$, derive an optimal solution $y^* \in SOL_{\mathcal{P}}(x)$ and its measure $m^*_{\mathcal{P}}(x)$
- *Evaluation problem* $\mathcal{P}_E$ – given an instance $x \in I_{\mathcal{P}}$, derive its optimal measure $m^*_{\mathcal{P}}(x)$
- *Decision problem* $\mathcal{P}_D$ – given an instance $x \in I_{\mathcal{P}}$ and a positive integer value K, derive whether $m^*_{\mathcal{P}}(x) \geq K$ if $goal = MAX$ or $m^*_{\mathcal{P}}(x) \leq K$ if $goal = MIN$

Underlying language of $\mathcal{P}$ is

1. $\{(x, K) | x \in I \wedge m^*(x) \geq K\}$ if $goal = MAX$
2. $\{(x, K) | x \in I \wedge m^*(x) \leq K\}$ if $goal = MIN$

Examples

Minimum Vertex Cover

- Instance: Graph $G = (V, E)$, $K \in \mathbb{N}$
- Question: derive whether exists vertex cover on G of size $\leq K$

Minimum Traveling Salesperson(TSP)

- Instance: Set of cities $\{c_1, \ldots, c_n\}$, $n \times n$ matrix D of distances
- Solutions: permutations $\{c_{i_1}, \ldots, c_{i_n}\}$
- Measure: $\sum_{k=1}^{n-1} D(i_k, i_{k+1}) + D(i_n, i_1)$

Overview

Example: Minimum Path

Find minimum path between two nodes in the graph G

- Instance: Graph $G = (V, E)$, two nodes $v_s, v_d \in V$
- Solution: A path $(v_s = v_{i_1}, \ldots, v_{i_k} = v_d)$
- Measure: k

Can be solved by breadth first search

Class NPO

Definition

Problem $\mathcal{P} = (I, SOL, m, goal)$ belongs to the class NPO if:

1. I is recognizable in polynomial time
2. there exists a polynomial q such that, given an instance $x \in I$, for any y, $|y| < q(|x|)$, it is decidable in polynomial time whether $y \in SOL(x)$
3. the measure function m is computable in polynomial time

Example

Minimum Vertex Cover belongs to NPO since:

1. any graph is recognizable in polynomial time
2. size of any feasible solution y is smaller then number of vertexes, testing whether a subset $U \subseteq V$ requires testing whether any edge in E is incident to at least one node in U
3. the size of U is clearly computable in polynomial time

Corresponding decision problem

Theorem

For any optimization problem $\mathcal{P}$ in NPO, the corresponding decision problem $\mathcal{P}_D$ belongs to NP

Proof.

Obviously: solution y should be guessed □

Definition

Optimization problem $\mathcal{P}$ belongs to the class PO if it is in NPO and there exists a polynomial-time computable algorithm $\mathcal{A}$ that for any instance $x \in I$, returns an optimal solution $y \in SOL^*(x)$ together with its value $m^*(x)$

Overview

NP-hard optimization problems

Definition

An optimization problem $\mathcal{P}$ is called NP-hard, if for every decision problem $\mathcal{P}' \in NP$, $\mathcal{P}' \leq_T \mathcal{P}$, $\mathcal{P}'$ can be solved in polynomial time by an algorithm which uses an oracle that, for any instance $x \in I_{\mathcal{P}}$, returns an optimal solution y^* of x and its value $m^*_{\mathcal{P}}(x)$

When problem is NP-hard?

Theorem

Let a problem $\mathcal{P} \in$ NPO be given; if underlying language of $\mathcal{P}$ is NP-complete then $\mathcal{P}$ is NP-hard

Proof.

Clearly the solution of the decision problem could be obtained for free if an oracle would give us the solution of the constructive optimization problem □

Corollary

If NP $\neq$ P then PO $\neq$ NPO

Relations between decision, evaluation and constructive problems

Theorem

For any problem $\mathcal{P} \in NPO$, $\mathcal{P}_D \equiv_T \mathcal{P}_E \leq_T \mathcal{P}_C$

Proof.

It is clearly that $\mathcal{P}_D \leq_T \mathcal{P}_E \leq_T \mathcal{P}_C$.
The range of possible values of $m(x, y)$ is bounded by $M = 2^{p(|x|)}$ for some polynomial p. Hence, by applying binary search the evaluation problem could be solved by at most $log(M) = p(|x|)$ queries to the oracle $\mathcal{P}_D$ □

Example

Maximum Clique

- Instance: Graph $G = (V, E)$
- Solution: A clique in G
  $U \subseteq V \ \forall (v_i, v_j) \in U \times U : (v_i, v_j) \in E \lor v_i = v_j$

MaximumClique could be solved using an oracle that can solve evaluation problem $MaximumClique_E$
$MaximumClique(G)$:

1. Compute k – size of the maximum clique in the graph G
2. if $k = 1$ return any node
3. find node v for which $MaximumClique_E(G(v)) = k$
4. return $\{v\} \cup MaximumClique(G^{-}(v))$

Where $G(v)$ – a subgraph induced by v and its neighbors, $G^{-}(v)$ – a subgraph induced by neighbors of v

Constructive to decision problem reduction

Theorem

If problem $\mathcal{P} \in$ NPO and $\mathcal{P}_D$ is NP-complete, then $\mathcal{P}_C \leq_T \mathcal{P}_D$

Proof.

Let us assume, that $\mathcal{P}$ is a maximization problem.
Problem $\mathcal{P}'$ has the same definition except for the measure function $m_{\mathcal{P}'}$, which is defined as follows. Let p a polynomial, which bounds the length of the solutions of $\mathcal{P}$. Let $\lambda(y)$ denote the rank of y in the lexicographical order. Then we denote the measure function $m_{\mathcal{P}'}(x, y) = 2^{p(|x|)+1} m_{\mathcal{P}}(x, y) + \lambda(y)$.

Constructive to decision problem reduction(2)

Proof.

1. For all $y_1, y_2 \in SOL_{\mathcal{P}'}(x)$ measure functions are different $m_{\mathcal{P}}(x, y_1) \neq m_{\mathcal{P}}(x, y_2)$. Therefore exists only unique optimal solution of the problem $\mathcal{P}'$ $y^*_{\mathcal{P}'} \in SOL^*_{\mathcal{P}'}(x)$.
2. If $m_{\mathcal{P}'}(x, y_1) > m_{\mathcal{P}'}(x, y_2)$ then $m_{\mathcal{P}}(x, y_1) \geq m_{\mathcal{P}}(x, y_2)$. Therefore, $y^*_{\mathcal{P}'} \in SOL^*_{\mathcal{P}}(x)$

Optimal solution could be derived by computing remainder of the division $m^*_{(P)'}(x)$ by $2^{p(|x|)+1}$.
$\mathcal{P}_D$ is NP-complete and it can be used to solve $\mathcal{P}'_D$. □

Overview

Approximate algorithms

Definition

Given an optimization problem $\mathcal{P} = (I, SOL, m, goal)$, an algorithm $\mathcal{A}$ is an *approximation* algorithm for $\mathcal{P}$ if for any given instance $x \in I$ it returns an approximate solution, that is a feasible solution $A(x) \in SOL(x)$

Overview

Absolute approximate algorithms

Definition

Given an optimization problem $\mathcal{P} = (I, SOL, m, goal)$, for any instance x and for any feasible solution y of x, the absolute error of y with respect to x is defined as

$$D(x, y) = |m^*(x) - m(x, y)|$$

Definition

Given an optimization problem $\mathcal{P} = (I, SOL, m, goal)$ and an approximation algorithm $\mathcal{A}$ for $\mathcal{P}$ we say that $\mathcal{A}$ is an absolute approximation algorithm if there exists a constant k such that, for every instance x of $\mathcal{P}$, $D(x, \mathcal{A}(x)) \leq k$

Problem, that not allow absolute approximation algorithm, unless $P = NP$

Theorem

Unless $P = NP$, no polynomial-time absolute approximation algorithm exists for Maximum Knapsack

Proof.

Let X be a set of n items with profits $p_1, \ldots, p_n$ and weights $a_1, \ldots, a_n$, and let b be the knapsack capacity. If the problem would allow approximation algorithm with absolute error k, then consider another instance with profits multiplied by $k + 1$. The set of the feasible solutions is the same. The only solution with absolute error bounded by k can be found. Hence we can solve exactly original problem. □

Overview

Relative error

Definition

Given an optimization problem $\mathcal{P}$, for any instance x and for any feasible solution y of x, the *relative error* defined as

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

Definition

Given an optimization problem $\mathcal{P}$ and an approximation algorithm $\mathcal{A}$, we say that $\mathcal{A}$ is ϵ-approximation algorithm if the relative approximation error provided by $\mathcal{A}$

$$E(x, y) \leq \epsilon$$

Performance ratio

Definition

Given an optimization problem $\mathcal{P}$, for any instance x and for any feasible solution y of x, the *performance ratio* defined as

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

Definition

Given an optimization problem $\mathcal{P}$ and an approximation algorithm $\mathcal{A}$, we say that $\mathcal{A}$ is r-approximation algorithm if the performance ratio provided by $\mathcal{A}$

$$R(x, y) \leq r$$

Class APX

Definition

APX is the class of NPO problems such that for some $r \geq 1$ there exists a polynomial-time r-approximate algorithm.

Problem that not belongs to APX, unless $P = NP$

Theorem

If Minimum Traveling Salesperson problem belongs to APX, then P = NP

Proof.

Let us consider that exists polynomial-time r-approximate algorithm for MinTSP. For every instance of the Hamiltonian Circle decision problem we can construct the following Traveling Salesperson problem.

Problem that not belongs to APX, unless $P = NP(2)$

Proof.

Let distances on the same graph $G = (V, E)$

$$d(v_i, v_j) = \begin{cases} 1 & \text{if } (v_i, v_j) \in E \\ 1 + nr & \text{otherwise} \end{cases}$$

□

Corollary

If $P \neq NP$, then $APX \subset NPO$

Overview

The gap technique

Theorem

Let $\mathcal{P}'$ be an NP-complete decision problem and let $\mathcal{P}$ be an NPO minimization problem. Let us suppose, that there exist two polynomial-time computable functions $f : I_{\mathcal{P}'} \to I_{\mathcal{P}}$, $c : I_{\mathcal{P}'} \to \mathbb{N}$ and a constant $gap > 0$, such that for any instance x

$$m^*(f(x)) = \begin{cases} c(x) & \text{if } x \text{ is a positive instance,} \\ c(x)(1 + gap) & \text{otherwise} \end{cases}$$

Then no polynomial-time r-approximate algorithm for $\mathcal{P}$ with $r < 1 + gap$ can exist, unless $P = NP$

The gap technique(2)

Proof.

We can use approximation algorithm of $\mathcal{P}$ for solving $\mathcal{P}'$ in the following way. Let us apply approximation algorithm $\mathcal{A}$ to $f(x)$

1. if x is negative instance, $m^*(f(x)) \geq c(x)(1 + gap)$ and $m(f(x), \mathcal{A}) \geq c(x)(1 + gap)$
2. if x is positive instance, we have that

$$\frac{m(f(x), \mathcal{A})}{m^*(f(x))} \leq r < 1 + gap$$

$m^*(f(x)) = c(x)$ hence $m(f(x), \mathcal{A}) < c(x)(1 + gap)$

□

Overview

Polynomial time approximation schemes

Definition

Let $\mathcal{P}$ be an NPO problem. An algorithm $\mathcal{A}$ is said to be *polynomial time approximation scheme(PTAS)* if, for any instance x and for any rational number $r > 1$, $\mathcal{A}$ applied to (x, r) returns an r-approximate solution of x in time polynomial in $|x|$.

The running time of a PTAS may also depend on $1/(r-1)$

Definition

PTAS is the class of NPO problems that admit polynomial time approximation scheme

Overview

APX versus PTAS

Theorem

If $P \neq NP$, then Minimum Bin Packing does not belong to the class PTAS

Corollary

If $P \neq NP$, then $PTAS \subset APX$

Overview

The class FPTAS

Definition

Let $\mathcal{P}$ be an NPO problem. An algorithm $\mathcal{A}$ is said to be *fully polynomial time approximation scheme(FPTAS)* if, for any instance x and for any rational number $r > 1$, $\mathcal{A}$ applied to (x, r) returns an r-approximate solution of x in time polynomial both in $|x|$ and $1/(r-1)$.

Definition

FPTAS is the class of NPO problems that admit fully polynomial time approximation scheme

Overview

Many problems not belong to class FPTAS, unless $P = NP$

Definition

An optimization problem is *polynomially* bounded if there exists a polynomial p such that for any instance x and for any $y \in SOL(x)$, $m(x, y) \leq p(|x|)$

Theorem

No NP-hard polynomially bounded optimization problem belongs to the class FPTAS, unless $P = NP$

Many problems not belong to class FPTAS, unless $P = NP$(2)

Proof.

Suppose we have a FPTAS $\mathcal{A}$ for the problem $\mathcal{P}$ which, for any instance x and for any rational $r > 1$, runs in time bounded $q(|x|, 1/(r-1))$. Since $\mathcal{P}$ is polynomially bounded there exists polynomial p such that $m^*(x) \leq p(|x|)$. If we choose $r = 1 + 1/p(|x|)$, then $\mathcal{A}$ provides an optimal solution.

$$m(x, \mathcal{A}(x, r)) \geq m^*(x) \frac{p(|x|)}{p(|x|) + 1} > m^*(x) - 1$$

□

Many problems not belong to class FPTAS, unless $P = NP$(3)

Corollary

if $P \neq NP$, then FPTAS $\subset$ PTAS

Proof.

Maximum Independent Set restricted to planar graphs belongs to PTAS. On the other side the problem is clearly polynomially bounded. □

Conclusion

- We defined optimization problems and considered some examples
- We defined different types of approximation algorithms for optimization problems. We denoted classes of optimization problems: APX, PTAS, FPTAS.