Approximation Algorithms III

2023/11/23

詹博华（中国科学院软件研究所）

Summary

We show two more techniques for designing approximation algorithms:

- **Randomization:** applied to the MAX-3-CNF satisfiability problem.
- **(LP) Relaxation:** applied to weighted vertex cover.

MAX-3-CNF Satisfiability

- We say that a randomized algorithm has approximation ratio $\rho(n)$ if, for any input of size n, the *expected cost* C of the solution produced by the randomized algorithm is within a factor of $\rho(n)$ of the cost of the optimal solution.
- **MAX-3-CNF satisfiability:** given a 3-CNF formula ϕ, return an assignment to variables in ϕ that **maximizes** the number of clauses satisfied.

MAX-3-CNF Satisfiability

Theorem: randomly setting each variable to 1 with probability ½ and 0 with probability ½ yields a randomized 8/7-approximation algorithm.

Note: we assume each clause has exactly three distinct literals, with no clause containing both a variable and its negation.

MAX-3-CNF Satisfiability

Proof:

- Given a random assignment of variables x_i, each clause is satisfied with probability $7/8$ (each literal is satisfied with probability $1/2$, so the probability that all three literals are not satisfied has probability $1/8$).
- This shows that the expected number of clauses that are satisfied is $7m/8$, where m is the number of clauses (this makes crucial use of **linearity of expectation**,, $E[A+B] = E[A] + E[B]$, even if random variables A and B are not independent).
- Since m is an upper bound on the number of satisfied clauses, we have approximation ratio $\frac{m}{7m/8} = 8/7$.

Weighted vertex-cover

- Consider a weighted generalization of the vertex cover problem.
- Given graph $G = (V, E)$ where each vertex $v \in V$ has positive weight $w(v)$.
- Find the vertex cover with minimum total weight.
- **Approach:** compute a lower bound on minimum-weight vertex-cover using linear programming, then "round" the solution to obtain an actual vertex cover.

Conversion to Linear-Programming

- Associate a variable $x(v)$ to each vertex $v \in V$.
- $x(v) = 0$ corresponds to not picking v, and $x(v) = 1$ corresponds to picking v for the vertex cover.
- This gives rise to a **0-1 integer program** as follows.

$$\text{Minimize } \sum_{v \in V} w(v)x(v)$$

$$\text{Subject to } x(u) + x(v) \geq 1 \text{ for each } (u, v) \in E, \text{ and}$$

$$x(v) \in \{0,1\} \text{ for each } v \in V.$$

Relaxation

- This alone does not help, since we know 0-1 integer programming is also NP-complete.
- However, we can relax the problem by removing the $x(v) \in \{0,1\}$ constraint into $0 \leq x \leq 1$ (for real number x).
- This gives a **linear programming** problem:

$$\text{Minimize } \sum_{v \in V} w(v)x(v)$$

$$\text{Subject to } x(u) + x(v) \geq 1 \text{ for each } (u,v) \in E,$$

$$x(v) \leq 1 \text{ and } x(v) \geq 0 \text{ for each } v \in V.$$

Relaxation

- Since we have simply relaxed constraints in converting the 0-1 integer program to linear program, any solution to the 0-1 integer program is a solution to the linear program.
- This shows optimal solution to the linear program gives a **lower bound** for the solution of 0-1 integer program.
- The linear program can be solved using simplex algorithm or (in guaranteed polynomial time) interior-point methods.

Approximation Algorithm

- From the solution $\boldsymbol{x}$ to the linear program, we can give an approximate solution to weighted vertex-cover as follows: for any $v \in V$, add v to the vertex cover if $x(v) \geq 1/2$.

APPROX-MIN-WEIGHT-VC(G, w)

$C = \emptyset$
compute $\bar{x}$, an optimal solution to the linear program in lines (35.17)–(35.20)
for each $v \in V$
    if $\bar{x}(v) \geq 1/2$
        $C = C \cup \{v\}$
return C

Approximation Algorithm

Theorem: the above algorithm solves weighted vertex-cover with approximation ratio 2.

Proof: we need to show two parts:

1. The algorithm does give a vertex-cover.
2. The total weight returned is within a factor of 2 of the optimal solution.

Proof of two parts

First part: the solution returned is a vertex cover.

- For each edge (u, v), we have the constraint $x(u) + x(v) \geq 1$ in the linear program. Hence at least one of $x(u)$ and $x(v)$ must have value $\geq 1/2$, so at least one of u and v is placed in the vertex cover.

Second part: the total weight is within factor of 2 of the optimal.

- Starting from solution to the linear program, which gives a lower bound on the total weight. The solution returned **increases** the weight of each picked vertex by **at most a factor of 2** (from $\geq 1/2$ to 1), and **reduces** the weight of other vertices (from $\leq 1/2$ to 0).

Summary: technique of relaxation

We have given an algorithm for weighted vertex-cover with approximation ratio 2.

The basic idea is:

- **Relax** the problem into one that is easier to solve, by removing/weakening some constraints.
- Solution to the weakened problem gives a lower bound on the optimal solution. **Adjust** the solution to the weakened problem so the original constraints are satisfied.

Summary of Class

In this class, we have learned:

- Time complexity (divide and conquer, amortized analysis).
- Heaps and hash tables, sorting and searching.
- Red-black trees and B-trees.
- Dynamic programming and greedy algorithm.
- Graph algorithms: DFS, BFS, shortest-path, minimum spanning tree, network flow.
- Linear programming.
- NP-completeness and approximation algorithms.

Summary of Class

But more importantly, we should have learned:

- How to choose appropriate algorithms for a known problem.
- How to design new algorithms for new problems.
- How to prove correctness and analyze efficiency of algorithms.
- How to identify problems that are too hard (NP-complete), and what to do for these problems.

Summary of Class

Many topics in the textbook are left out:

- Randomized algorithms.
- Augmenting data structures (interval trees).
- Fibonacci heaps, van Emde Boas Trees, union-find.
- Algorithms in linear algebra and number theory.
- Fast fourier transform.
- String algorithms.
- Computational geometry.

Summary of Class

But even more topics are not in the textbook at all:

- Local and global optimization (gradient methods, simulated annealing, genetic algorithms).
- Algorithms for SAT and its generalizations.
- Convex programming.
- Concurrent data structures.
- Distributed algorithms.
- …

We are really just getting started!