Approximation Algorithms

Frédéric Giroire

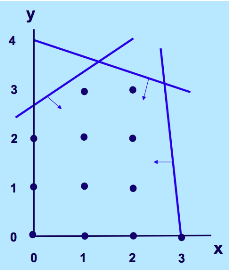
Motivation

- Goal:
  - Find "good" solutions for difficult problems (NP-hard).
  - Be able to quantify the "goodness" of the given solution.
- Presentation of a technique to get approximation algorithms: fractional relaxation of integer linear programs.

Fractional Relaxation

- Reminder:
  - Integer Linear Programs often hard to solve (NP-hard).
  - Linear Programs (with real numbers) easier to solve (polynomial-time algorithms).

- Idea:
  - 1- Relax the integrality constraints;
  - 2- Solve the (fractional) linear program and then;
  - 3- Round the solution to obtain an integral solution.



Set Cover

Definition: An approximation algorithm produces

- in polynomial time
- a feasible solution
- whose objective function value is close to the optimal OPT, by close we mean within a guaranteed factor of the optimal.

Example: a factor 2 approximation algorithm for the cardinality vertex cover problem, i.e. an algorithm that finds a cover of cost $\leq 2 \cdot OPT$ in time polynomial in $|V|$.

Set Cover

- Problem: Given a universe U of n elements, a collection of subsets of U, $\mathscr{S} = S_1, ..., S_k$, and a cost function $c : S \rightarrow Q^+$, find a minimum cost subcollection of S that covers all elements of U.
- Model numerous classical problems as special cases of set cover: vertex cover, minimum cost shortest path...
- Definition: The frequency of an element is the number of sets it is in. The frequency of the most frequent element is denoted by f.
- Various approximation algorithms for set cover achieve one of the two factors $O(\log n)$ or f.

Fractional relaxation

Write a linear program to solve vertex cover.

Var.: $x_S = 1$ if S picked in $\mathscr{C}$,
$x_S = 0$ otherwise

min $\sum_{S \in \mathscr{S}} c(S) x_S$

s. t.

$\sum_{S : e \in S} x_S \leq 1 \quad (\forall e \in U)$

$x_S \in \{0, 1\} \quad (\forall S \in \mathscr{S})$

Var.: $1 \geq x_S \geq 0$

min $\sum_{S \in \mathscr{S}} c(S) x_S$

s. t.

$\sum_{S : e \in S} x_S \leq 1 \quad (\forall e \in U)$

$x_S \geq 0 \quad (\forall S \in \mathscr{S})$

Fractional relaxation

Write a linear program to solve vertex cover.

Var.:	$x_S = 1$ if S picked in $\mathscr{C}$, $x_S = 0$ otherwise	
min	$\sum_{S \in \mathscr{S}} c(S) x_S$	
s. t.		
	$\sum_{S : e \in S} x_S \leq 1$	$(\forall e \in U)$
	$x_S \in \{0, 1\}$	$(\forall S \in \mathscr{S})$

Var.:	$1 \geq x_S \geq 0$	
min	$\sum_{S \in \mathscr{S}} c(S) x_S$	
s. t.		
	$\sum_{S : e \in S} x_S \leq 1$	$(\forall e \in U)$
	$x_S \geq 0$	$(\forall S \in \mathscr{S})$

Fractional relaxation

Write a linear program to solve vertex cover.

Var.:	$x_S = 1$ if S picked in $\mathscr{C}$, $x_S = 0$ otherwise
min	$\sum_{S \in \mathscr{S}} c(S) x_S$
s. t.	
	$\sum_{S : e \in S} x_S \leq 1$ $\quad (\forall e \in U)$ $x_S \in \{0, 1\}$ $\quad (\forall S \in \mathscr{S})$

Var.:	$1 \geq x_S \geq 0$
min	$\sum_{S \in \mathscr{S}} c(S) x_S$
s. t.	
	$\sum_{S : e \in S} x_S \leq 1$ $\quad (\forall e \in U)$ $x_S \geq 0$ $\quad (\forall S \in \mathscr{S})$

Fractional relaxation

- The (fractional) optimal solution of the relaxation is a lower bound of the optimal solution of the original integer linear program.
- Example in which a fractional set cover may be cheaper than the optimal integral set cover:
  Input: $U = \{e, f, g\}$ and the specified sets $S_1 = \{e, f\}$, $S_2 = \{f, g\}$, $S_3 = \{e, g\}$, each of unit cost.
  - An integral cover of cost 2 (must pick two of the sets).
  - A fractional cover of cost 3/2 (each set picked to the extent of 1/2).

Fractional relaxation

- The (fractional) optimal solution of the relaxation is a lower bound of the optimal solution of the original integer linear program.
- Example in which a fractional set cover may be cheaper than the optimal integral set cover:
  Input: $U = \{e, f, g\}$ and the specified sets $S_1 = \{e, f\}$, $S_2 = \{f, g\}$, $S_3 = \{e, g\}$, each of unit cost.
  - An integral cover of cost 2 (must pick two of the sets).
  - A fractional cover of cost 3/2 (each set picked to the extent of 1/2).

A simple rounding algorithm

Algorithm:

- 1- Find an optimal solution to the LP-relaxation.
- 2- (Rounding) Pick all sets S for which $x_S \geq 1/f$ in this solution.

- Theorem: The algorithm achieves an approximation factor of f for the set cover problem.
- Proof:
  - 1) All elements are covered. e is in at most f sets, thus one of this set must be picked to the extent of at least 1/f in the fractional cover.
  - 2) The rounding process increases x_S by a factor of at most f. Therefore, the cost of $\mathscr{C}$ is at most f times the cost of the fractional cover.

$$OPT_f \leq OPT \leq f \cdot OPT_f$$

- Theorem: The algorithm achieves an approximation factor of f for the set cover problem.
- Proof:
  - 1) All elements are covered. e is in at most f sets, thus one of this set must be picked to the extent of at least 1/f in the fractional cover.
  - 2) The rounding process increases x_S by a factor of at most f. Therefore, the cost of $\mathscr{C}$ is at most f times the cost of the fractional cover.

$$OPT_f \leq OPT \leq f \cdot OPT_f$$

- Theorem: The algorithm achieves an approximation factor of f for the set cover problem.
- Proof:
  - 1) All elements are covered. e is in at most f sets, thus one of this set must be picked to the extent of at least 1/f in the fractional cover.
  - 2) The rounding process increases x_S by a factor of at most f. Therefore, the cost of $\mathscr{C}$ is at most f times the cost of the fractional cover.

$$OPT_f \leq OPT \leq f \cdot OPT_f$$

Randomized rounding

- Idea: View the optimal fractional solutions as probabilities.
- Algorithm:
  - Flip coins with biases and round accordingly (S is in the cover with probability x_S).
  - Repeat the rouding $O(\log n)$ times.
- This leads to an $O(\log n)$ factor randomized approximation algorithm. That is
  - The set is covered with high probability.
  - The cover has expected cost: $O(\log n)OPT$.

Take Aways

- Fractional relaxation is a method to obtain for some problems:
  - Lower bounds on the optimal solution of an integer linear program (minimization).
    Remark: Used in Branch & Bound algorithms to cut branches.
  - Polynomial approximation algorithms (with rounding).
- Complexity:
  - Integer linear programs are often hard.
  - (Fractional) linear programs are quicker to solve (polynomial time).