Chapter 1: An Introduction to Approximation Algorithms

(cp. Williamson & Shmoys, Chapter 1)

How to Tackle *NP*-Hard Optimization Problems?

Most interesting discrete optimization problems are *NP*-hard.

Thus, unless $P = NP$, for such problems we cannot find algorithms that simultaneously

1. find optimal solutions
2. in polynomial time
3. for any instance.

To deal with *NP*-hard optimization problems, we need to relax at least one of the three requirements.

In this course, we study approximation algorithms. That is, we relax the first requirement, i. e., we search for algorithms that produce solutions that are "good enough".

What is "good enough"?
We would like to have some sort of performance guarantee.

Approximation Algorithms

Definition 1.1.

An α-approximation algorithm for an optimization problem is a polynomial-time algorithm that for all instances of the problem produces a solution whose value is within a factor of α of the optimum value.

For an α-approximation algorithm, we call α the performance guarantee (or approximation ratio/factor) of the algorithm.

Convention:

- $\alpha > 1$ for minimization problems, and
- $\alpha < 1$ for maximization problems.

Thus, a $\frac{1}{2}$-approximation algorithm for a maximization problem is a polynomial-time algorithm that always returns a solution whose value is at least half the value of the optimum value.

Example:
Christofides' algorithm is a $\frac{3}{2}$-approximation algorithm for the TSP.

Polynomial-Time Approximation Schemes

Remarks:

- The worst-case bounds are often due to pathological cases that hardly arise in practice.
- Often, approximation algorithms are much better in practice than indicated by their performance guarantee.

Definition 1.2.

A polynomial-time approximation scheme (PTAS) is a family of algorithms $\{A_\varepsilon\}$, where there is an algorithm for each $\varepsilon > 0$, such that A_ε is a

- $(1+\varepsilon)$-approximation algorithm (for minimization problems), or a
- $(1-\varepsilon)$-approximation algorithm (for maximization problems).

Examples: PTASes exist for the Knapsack Problem and the Euclidean TSP.

Set Cover Problem

- There are several fundamental techniques used in the design and analysis of approximation algorithms.
- In particular, linear programming plays an essential role!
- In this introductory chapter, we illustrate some of the techniques on the Set Cover Problem.

Set Cover Problem:
Given: A set of elements $E = \{e_1, \ldots, e_n\}$, a family of subsets $\{S_1, \ldots, S_m\} \subseteq 2^E$, and a weight $w_j \geq 0$ for each $j \in \{1, \ldots, m\}$.

Task: Find $I \subseteq \{1, \ldots, m\}$ minimizing $\sum_{j \in I} w_j$ s.t. $\bigcup_{j \in I} S_j = E$.

If $w_j = 1$ for each $j \in \{1, \ldots, m\}$, the problem is called Unweighted Set Cover Problem.

IP Formulation of Set Cover

Formulation as integer linear program:

$$
\begin{aligned}
z^*_{IP} = \quad & \min \quad \sum_{j=1}^{m} w_j \cdot x_j & & & & \\
& \text{s.t.} \quad \sum_{j : e_i \in S_j} x_j & \geq & \quad 1 & \text{for all } i = 1, \ldots, n & \qquad (1.1) \\
& \qquad\qquad x_j & \in & \quad \{0,1\} & \text{for all } j = 1, \ldots, m &
\end{aligned}
$$

Linear programming relaxation:

$$
\begin{aligned}
z^*_{LP} = \quad & \min \quad \sum_{j=1}^{m} w_j \cdot x_j & & & & \\
& \text{s.t.} \quad \sum_{j : e_i \in S_j} x_j & \geq & \quad 1 & \text{for all } i = 1, \ldots, n & \qquad (1.2) \\
& \qquad\qquad x_j & \geq & \quad 0 & \text{for all } j = 1, \ldots, m &
\end{aligned}
$$

Note: $x_j \geq 0$ suffices as $x_j \leq 1$ is redundant.

A Deterministic Rounding Algorithm

For each $i = 1, \ldots, n$ let $f_i := |\{j \mid e_i \in S_j\}|$ be the number of sets containing e_i, and $f := \max_{i=1,\ldots,n} f_i$.

Deterministic Rounding Algorithm for Set Cover:

1. Compute an optimal solution x^* to the set-cover-LP (1.2).
2. For each $j \in \{1, \ldots, m\}$, set $\hat{x}_j = 1$ if $x_j^* \geq \frac{1}{f}$, and $\hat{x}_j = 0$ otherwise.

Lemma 1.3.

The collection of subsets S_j with $j \in \hat{I} := \{j \mid \hat{x}_j = 1\}$ is a set cover.

Proof: ... □

Theorem 1.4.

The rounding algorithm above is an f-approximation algorithm for the Set Cover Problem.

Proof: ... □

Note: The algorithm is a 2-approximation for the Vertex Cover Problem.

Dual Linear Program

Linear programming relaxation:

$$\begin{aligned} z_{LP}^* = \quad \min \quad & \sum_{j=1}^{m} w_j \cdot x_j & & \\ \text{s.t.} \quad & \sum_{j:e_i \in S_j} x_j \geq 1 & & \text{for all } i = 1, \ldots, n \\ & x_j \geq 0 & & \text{for all } j = 1, \ldots, m \end{aligned}$$

Dual linear program:

$$\begin{aligned} z_{LP}^* = \quad \max \quad & \sum_{i=1}^{n} y_i & & \\ \text{s.t.} \quad & \sum_{e_i \in S_j} y_i \leq w_j & & \text{for all } j = 1, \ldots, m \qquad (1.3) \\ & y_i \geq 0 & & \text{for all } i = 1, \ldots, n \end{aligned}$$

Rounding a Dual Solution

Dual Rounding Algorithm for Set Cover:

1. compute an optimal solution y^* to the dual (1.3) of the set-cover-LP;
2. let $I^* := \{j \mid \sum_{i:e_i \in S_j} y_i = w_j\}$;

Lemma 1.5.

The collection of subsets S_j with $j \in I^*$ is a set cover.

Proof: ... □

Theorem 1.6.

The dual rounding algorithm is an f-approximation algorithm for the Set Cover Problem.

Proof: ... □

Remark: Notice that $I^* \supseteq \hat{I}$ due to complementary slackness!

Primal-Dual Algorithm

Note: The two previous algorithms require solving a linear program. Special purpose algorithms are often much faster!

Idea: Construct a feasible dual solution that is "good enough".

Primal-dual algorithm for the Set Cover Problem:

1. set $y :\equiv 0$ and $I := \emptyset$;
2. while $\exists e_k \notin \bigcup_{j \in I} S_j$
3. increase y_k until $\exists j$ with $e_k \in S_j$ such that $\sum_{i:e_i \in S_j} y_i = w_j$;
4. set $I := I \cup \{j\}$;

Theorem 1.7.

The primal-dual algorithm is an f-approximation algorithm for the Set Cover Problem.

Proof: As before. □

Greedy Algorithm

Idea: Iteratively select a set minimizing the ratio of its weight to the number of currently uncovered elements it contains.

Greedy algorithm for the Set Cover Problem

1. set $I := \emptyset$ and $\hat{S}_j := S_j$ for all j;
2. while I is not a cover
3. $\quad \ell := \operatorname{argmin}\left\{ \frac{w_j}{|\hat{S}_j|} \mid \hat{S}_j \neq \emptyset \right\}$;
4. $\quad$ set $I := I \cup \{\ell\}$;
5. $\quad$ set $\hat{S}_j := \hat{S}_j \setminus S_\ell$ for all j;

Theorem 1.8.

The greedy algorithm returns a cover I with $w(I) \leq H_g \cdot z^*_{LP}$, where $g := \max_j |S_j|$ and $H_g := \sum_{k=1}^{g} \frac{1}{k} \approx \ln\ g$.

Proof: Use technique called dual fitting... □

Nonapproximability Results for Set Cover

Theorem 1.9 (Lund & Yannakakis 1994,).

If there is a $c \ln n$-approximation algorithm for the Unweighted Set Cover Problem for some constant $c < 1$, then there is an $O(n^{O(\log \log n)})$-time deterministic algorithm for each NP-complete problem.

Theorem 1.10 (Feige 1998).

There is some constant $c > 0$ such that if there is a $c \ln n$-approximation algorithm for the Unweighted Set Cover Problem, then $P = NP$.

Theorem 1.11 (Dinur & Safra 2002).

If there is an α-approximation algorithm for the Vertex Cover Problem with $\alpha < 10\sqrt{5} - 21 \approx 1.36$, then $P = NP$.

Theorem 1.12 (Khot & Regev 2008).

Assuming the Unique Games Conjecture holds, if there is an α-approximation algorithm for the Vertex Cover Problem with $\alpha < 2$, then $P = NP$.