Chapter 17: An Introduction to Approximation Algorithms

(cp. Williamson, Shmoys: "The Design of Approximation Algorithms")

Approximation Algorithms

Most interesting discrete optimization problems are $\mathcal{NP}$-hard.

Thus, unless $\mathcal{P} = \mathcal{NP}$, for such problems we cannot find algorithms that simultaneaously

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

To deal with $\mathcal{NP}$-hard optimization problems, we need to relax at least one of the three requirements.

In this Chapter, we give a brief introduction to approximation algorithms.That is, we relax the first requirement, i.e., we search for algorithms that produce solutions that are "good enough".

Next Semester: Lecture "Approximation Algorithm."

We would like to have some sort of performance guarantee.

Performance Guarantee

Definition 17.1.

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 value of an optimal solution.

For an α-approximation algorithm, we call α the performance guarantee (or approximation factor/ratio) 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 value is at least half the value of the optimal value.

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

Polynomial-Time Approximation Schemes

Note: The worst-case bounds are often due to pathological cases that do not arise in practice.

Often, approximation algorithms are much better in practice than indicated by their performance guarantee.

Definition 17.2.

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

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

Examples: PTAS exist for the knapsack problem and the Euclidean TSP ($\longrightarrow$ next semester.)

The Set Cover Problem

There are several fundamental techniques used in the design and analysis of approximation algorithms ($\longrightarrow$ next semester.)

In particulare, linear programming plays an essential role!

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.

Set-Cover-IP

Formulation as integer linear program:

$$\begin{array}{rlrcll} z^*_{IP} = & \min & \sum_{j=1}^m w_j x_j & & & \\ & \text{s.t.} & \sum_{j:e_i \in S_j} x_j & \geq & 1 & \forall i = 1, \ldots, n \\ & & x_j & \in & \{0,1\} & \forall j = 1. \ldots, m \end{array} \qquad (17.1)$$

Applications:

- Development of antivirus product,
- the VERTEX COVER PROBLEM,
- ...

Linear relaxation of the set-cover-IP:

$$\begin{array}{rlrcll} z^*_{LP} = & \min & \sum_{j=1}^m w_j x_j & & & \\ & \text{s.t.} & \sum_{j:e_i \in S_j} x_j & \geq & 1 & \forall i = 1, \ldots, n \\ & & x_j & \geq & 0 & \forall j = 1. \ldots, m \end{array} \qquad (17.2)$$

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 (17.2).
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 17.3.

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

Proof: ... □

Theorem 17.4.

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.

Rounding a Dual Solution

Dual Rounding Algorithm for Set Cover:

1. Compute an optimal solution y^* to the dual of the set-cover-LP (17.2).
2. Let $I^* := \{j \mid \sum_{i:e_i \in S_j} y_i = w_j\}$;

Lemma 17.5.

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

Proof: ... □

Theorem 17.6.

The dual rounding algorithm is an f-approximation algorithm for the set cover problem.

Proof: ... □

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. Initialize: $y \equiv 0$ and $I = \emptyset$;
2. WHILE $\exists e_k \notin \bigcup_{j \in I} S_j$ DO
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 17.7.

The primal-dual algorithm is an f-approximation algorithm for the set cover problem.

Proof: as before. □

Greedy Algorithm

Idea: Iteratively, until all elements are covered, select a set that minimizes the ratio of its weight to the number of currently uncovered elements it contains.

Greedy algorithm for the set cover problem:

1. Initialize: $\mathcal{I} = \emptyset$ and $\hat{S}_j = S_j$ for all j;
2. WHILE $\mathcal{I}$ is not a cover DO
3. $I = \text{argmin}\{\frac{w_j}{|\hat{S}_j|} \mid \hat{S}_j \neq \emptyset\}$:
4. Set $\mathcal{I} = \mathcal{I} \cup \{I\}$;
5. Set $\hat{S}_j = \hat{S}_j \setminus S_I$ for all j;

Theorem 17.8.

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

Proof: ... □

No performance guarantee better than H_n possible (unless $\mathcal{P} = \mathcal{NP}$)!