

# Notes on Approximation Algorithms

Bhargav Samineni

## Abstract

A collection of some notes on the design and analysis of approximation algorithms and approximation techniques. Based mainly off of [1, 2].

## 1 Introduction to LP-Duality

For a minimization linear program in canonical form, the goal is to find a non-negative, rational vector  $x$  that minimizes a given linear objective function in  $x$  subject to some linear constraints on  $x$ . If there are  $n$  decision variables and  $m$  linear constraints, then  $x \in \mathbb{Q}^n$ , the coefficients of the linear objective function can be represented by a vector  $c \in \mathbb{Q}^n$ , the coefficients of the linear constraints can be represented by a matrix  $A = (a_{ij}) \in \mathbb{Q}^{m \times n}$ , and the values of the linear constraints can be represented by a vector  $b \in \mathbb{Q}^m$ .

### Definition 1.1 (Primal and Dual)

Given a linear programming problem in canonical form denoted as  $(P)$ , we can induce a problem denoted by  $(D)$  with the following form:

$$\begin{array}{ll} (P) & \text{minimize} \quad \sum_{j=1}^n c_j x_j \\ & \text{subject to} \quad \sum_{j=1}^n a_{ij} x_j \geq b_i \quad i = 1, \dots, m \\ & \quad \quad \quad x_j \geq 0 \quad j = 1, \dots, n \\ (D) & \text{maximize} \quad \sum_{i=1}^m b_i y_i \\ & \text{subject to} \quad \sum_{i=1}^m a_{ij} y_i \leq c_j \quad j = 1, \dots, n \\ & \quad \quad \quad y_i \geq 0 \quad i = 1, \dots, m \end{array}$$

where  $a_{ij}$ ,  $b_i$ , and  $c_i$  are given rational numbers and  $y_i$  corresponds to the  $i$ th inequality of  $(P)$ .

### Definition 1.2 (Primal)

The problem  $(P)$  is referred to as the *primal*.

### Definition 1.3 (Dual)

The *dual* of the primal is problem  $(D)$ .

Every feasible solution for the dual serves as a lower bound on the optimal objective function value of the primal. The reverse also holds in that every feasible solution to the primal serves as an upper bound on the optimal objective function value of the dual.

**Theorem 1.1 (Weak Duality).** *If  $x = (x_1, \dots, x_n)$  is a feasible solution to  $(P)$  and  $y = (y_1, \dots, y_m)$  is a feasible solution to  $(D)$ , then  $\sum_{j=1}^n c_j x_j \geq \sum_{i=1}^m b_i y_i$ .*

*Proof.*

$$\sum_{j=1}^n c_j x_j \geq \sum_{j=1}^n \left( \sum_{i=1}^m a_{ij} y_i \right) x_j = \sum_{i=1}^m \left( \sum_{j=1}^n a_{ij} x_j \right) y_i \geq \sum_{i=1}^m b_i y_i$$

■

As a consequence of [Theorem 1.1](#), if we find that there exist some feasible solutions to  $(P)$  and  $(D)$  that have matching objective function values, then these solutions must be optimal.

**Theorem 1.2** (Strong Duality/LP-Duality). *If  $(P)$  and  $(D)$  are both feasible, and  $x^* = (x_1^*, \dots, x_n^*)$  and  $y^* = (y_1^*, \dots, y_m^*)$  are optimal solutions to  $(P)$  and  $(D)$ , respectively, then  $\sum_{j=1}^n c_j x_j^* = \sum_{i=1}^m b_i y_i^*$ .*

As a corollary of [Theorem 1.2](#), we find that optimal solutions to  $(P)$  and  $(D)$  must satisfy a set of conditions.

**Corollary 1.1** (Complementary Slack Conditions). *Let  $x$  and  $y$  be feasible solutions to  $(P)$  and  $(D)$ , respectively. Then  $x$  and  $y$  are both optimal iff the following are satisfied:*

1. For each  $1 \leq j \leq n$ , either  $x_j = 0$  or  $\sum_{i=1}^m a_{ij} y_i = c_j$
2. For each  $1 \leq i \leq m$ , either  $y_i = 0$  or  $\sum_{j=1}^n a_{ij} x_j = b_i$ .

## 1.1 Algorithm Design Techniques

**Definition 1.4** (Extreme Point Solution)

Consider the polyhedron defining the set of feasible solutions to an LP. A feasible solution is called an *extreme point solution* if it is a vertex of the polyhedron.

**Theorem 1.3.** *For any objective function, there is an extreme point solution that is optimal.*

Combinatorial optimization problems can often be stated as integer programs (IP), which can be relaxed to an LP. For the case of NP-Hard problems, however, the polyhedron that defines the set of feasible solutions to the LP-relaxation may not have integer vertices. With the LP-relaxation, the goal is to find near optimal integral solutions.

One method to get an approximation algorithm is through *LP-rounding*. Rounding involves solving the LP-relaxation and then converting the fractional solution to an integral solution while ensuring that in the process the cost does not increase dramatically. The approximation ratio is given by comparing the cost of the integral and fractional solutions.

Another method is the *primal-dual schema*. Consider an LP-relaxation as the primal program. Under the scheme, an integral solution to the primal and a feasible solution to the dual are constructed iteratively. Any feasible solution to the dual also provides a lower bound on OPT. The approximation ratio is given by comparing the two solutions.

### 1.1.1 Integrality Gap

**Definition 1.5** (Integrality Gap)

Given an LP-relaxation for a minimization problem, let  $\text{OPT}_f(I)$  be the cost of an optimal fractional solution to an instance  $I$  of the problem. Similarly, let  $\text{OPT}(I)$  denote the cost of an optimal solution

to the original IP for an instance  $I$  of the problem. The *integrality gap* of the relaxation is

$$\sup_I \frac{\text{OPT}(I)}{\text{OPT}_f(I)}.$$

In the case of a maximization problem, the integrality gap is given by the infimum of the ratio.

When the integrality gap of an LP-relaxation is exactly 1, such a relaxation is called an *exact relaxation*. If the cost of a solution found by an algorithm is compared to the cost of an optimal fractional solution (or feasible dual solution), the best approximation factor one can hope to achieve is the integrality gap of the relaxation.

## 2 Set Cover

### Problem 2.1 (Set Cover)

Given a ground set  $U = \{e_1, \dots, e_n\}$ , a collection  $\mathcal{S} = \{S_1, \dots, S_m\}$  of subsets of  $U$ , and a cost function  $c: \mathcal{S} \rightarrow \mathbb{Q}^+$ , find a minimum cost subcollection of  $\mathcal{S}$  that covers all elements in  $U$ .

To formulate **Set Cover** as an IP, denote the binary decision variables by  $x_j$  for each  $S_j \in \mathcal{S}$ . If  $S_j$  is in the cover, then  $x_j = 1$ .

$$\begin{aligned} & \text{minimize} && \sum_{j=1}^m c(S_j)x_j \\ & \text{subject to} && \sum_{j: e_i \in S_j} x_j \geq 1 \quad i = 1, \dots, n \\ & && x_j \in \{0, 1\} \quad j = 1, \dots, m \end{aligned} \tag{1}$$

The LP-relaxation to this IP is given by letting  $x_j \geq 0$ . A solution to the LP-relaxation can be viewed as a fractional set cover.

$$\begin{aligned} & \text{minimize} && \sum_{j=1}^m c(S_j)x_j \\ & \text{subject to} && \sum_{j: e_i \in S_j} x_j \geq 1 \quad i = 1, \dots, n \\ & && x_j \geq 0 \quad j = 1, \dots, m \end{aligned} \tag{2}$$

Introducing a variable  $y_i$  for each  $e_i \in U$ , we get the dual program.

$$\begin{aligned} & \text{maximize} && \sum_{i=1}^n y_i \\ & \text{subject to} && \sum_{i: e_i \in S_j} y_i \leq c(S_j) \quad j = 1, \dots, m \\ & && y_i \geq 0 \quad i = 1, \dots, n \end{aligned} \tag{3}$$

### 2.1 Dual-Fitting Analysis

[Algorithm 1](#) defines the dual program's value for each variable as  $y_i = p(e_i)$ . However, the solution  $y$  constructed by this approach is infeasible as it may not satisfy the first constraint. To see this, note that if a set  $S_j$  is chosen by the algorithm, it distributes its cost across all its uncovered elements. If  $S_j \cap C \neq \emptyset$ , then this means that it contains some covered elements that already have some other set's cost distributed to them, which would cause  $\sum_{i: e_i \in S_j} y_i \geq c(S_j)$ . Instead, we may be able to show that some scaled solution  $y' = \frac{y}{\kappa}$  where  $\kappa > 1$  is feasible for the dual.

---

**Algorithm 1** Greedy Set Cover Algorithm

---

```
1:  $C, I \leftarrow \emptyset$ 
2: while  $C \neq U$  do
3:    $i = \operatorname{argmin}_{j: S_j \in \mathcal{S}} \frac{c(S_j)}{|S_j \setminus C|}$ 
4:    $p(e_i) = \frac{c(S_i)}{|S_i \setminus C|} \quad \forall e_i \in S_i \setminus C$ 
5:    $C \leftarrow C \cup S_i, I \leftarrow I \cup i, \mathcal{S} \leftarrow \mathcal{S} \setminus S_i$ 
6: return  $C$ 
```

---

**Lemma 2.1.** *The vector  $y' = \frac{y}{H_g}$  is a feasible solution for the dual program Eq. (3), where  $g = \max_i |S_i|$  and  $H_g$  is the  $g$ th harmonic number.*

*Proof.* Assume the algorithm runs for  $\ell$  iterations. Let  $a_{j,k}$  be the number of elements in  $S_j$  that are still uncovered at the start of the  $k$ th iteration (hence  $a_{1,j} = |S_j|, a_{\ell+1,j} = 0$ ). Let  $A_{j,k}$  denote the set of uncovered elements in  $S_j$  that are covered at the end of the  $k$ th iteration. If  $S_i$  is chosen in the  $k$ th iteration, for each covered element  $e_i \in A_{i,k}$ ,

$$y'_i = \frac{c(S_i)}{H_g a_{i,k}} \leq \frac{c(S_j)}{H_g a_{j,k}} \quad \forall S_j \in \mathcal{S}$$

since  $S_i$  minimizes the ratio  $\frac{c(S_i)}{a_{i,k}}$ . Then for any set  $S_j$ ,

$$\begin{aligned} \sum_{i: e_i \in S_j} y'_i &= \sum_{k=1}^{\ell} \sum_{i: e_i \in A_{j,k}} y'_i \leq \sum_{k=1}^{\ell} (a_{j,k} - a_{j,k+1}) \frac{c(S_j)}{H_g a_{j,k}} \\ &= \frac{c(S_j)}{H_g} \sum_{k=1}^{\ell} \frac{a_{j,k} - a_{j,k+1}}{a_{j,k}} \\ &= \frac{c(S_j)}{H_g} \sum_{k=1}^{\ell} \sum_{l=a_{j,k+1}+1}^{a_{j,k}} \frac{1}{a_{j,k}} \\ &\leq \frac{c(S_j)}{H_g} \sum_{k=1}^{\ell} \left( \frac{1}{a_{j,k}} + \frac{1}{a_{j,k}-1} + \dots + \frac{1}{a_{j,k+1}+1} \right) \\ &\leq \frac{c(S_j)}{H_g} \sum_{l=1}^{|S_j|} \frac{1}{l} = \frac{c(S_j)}{H_g} H_{|S_j|} \leq c(S_j) \end{aligned}$$

which satisfies the first constraint of Eq. (3). ■

**Theorem 2.1.** *Algorithm 1 gives a  $H_g$  approximation to Set Cover.*

*Proof.* Since the greedy algorithm distributes the cost of the sets it chooses across all the elements,  $\sum_{j \in I} c(S_j) = \sum_{i=1}^n y_i$ . By Lemma 2.1, we know that  $y'$  is a feasible solution to the dual, so using Theorem 1.1 we get that

$$\sum_{j \in I} c(S_j) = \sum_{i=1}^n y_i = H_g \sum_{i=1}^n y'_i \leq H_g \text{OPT}_f \leq H_g \text{OPT}.$$
■

### 3 Maximum Matchings

#### Definition 3.1 (Matching)

Given a graph  $G = (V, E)$ , a matching  $M \subseteq E$  is a set of edges such that no two edges in  $M$  share a common vertex.

#### Definition 3.2 (Covered/Exposed)

A node is  $M$ -covered if some edge in  $M$  is incident to it. Else it is  $M$ -exposed.

Note that a matching  $M$  covers exactly  $2|M|$  nodes, leaving  $|V| - 2|M|$  nodes exposed. A matching is *maximal* if adding any edge  $e \in E \setminus M$  to it causes it to no longer be a matching. A matching is *perfect* if it covers all the nodes. A basic decision problem is to decide if a graph has a perfect matching. A more general problem is to find a maximum matching.

#### Problem 3.1 (Maximum (Cardinality) Matching)

Given a graph  $G = (V, E)$ , find a matching  $M$  that has maximum cardinality. Equivalently, find a matching with the fewest exposed nodes.

#### 3.1 Augmenting Paths

A path  $P$  is a collection of edges  $(v_0, v_1), \dots, (v_{k-1}, v_k) = e_1, \dots, e_k$  where each  $v_i$  is distinct.

#### Definition 3.3 (Alternating Path)

Given a matching  $M$  in a graph  $G$ , a path  $P$  in  $G$  is  $M$ -alternating if it alternates between edges in  $M$  and edges in  $E \setminus M$ .

#### Definition 3.4 (Augmenting Path)

Given a matching  $M$  in a graph  $G$ , a path  $P$  in  $G$  is  $M$ -augmenting if it is  $M$ -alternating and its end nodes are distinct and  $M$ -exposed.

Augmenting paths can be used to identify a Maximum Matching in the following way.

**Theorem 3.1** (Augmenting Path Theorem). *A matching  $M$  in a graph  $G$  is maximum iff there is no  $M$ -augmenting path.*

*Proof.*  $\implies$  (by contrapositive): Suppose there exists some  $M$ -augmenting path  $P$  with end nodes  $u, v$ . Then  $M' = M \triangle P$  (i.e. the matching constructed by adding the unmatched edges and removing the matched edges of  $P$ ) covers all the nodes covered by  $M$  plus nodes  $u, v$ . Thus,  $M$  is not a maximum matching. ■

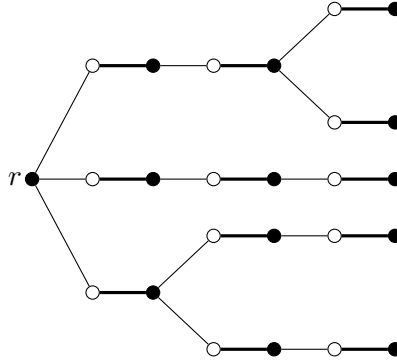
*Proof.*  $\impliedby$  (by contrapositive): Let  $M^*$  be a maximum matching. Let  $Q = M^* \triangle M$ . Then each node is incident to at most one edge in  $M^* \cap Q$  and one edge in  $M \cap Q$ . Hence,  $Q$  is an edge set of node disjoint paths and circuits where edges alternate between belonging in  $M^*$  and  $M$ . Because the edges are taken from matchings, all circuits must be of even length and contain the same number of edges from  $M^*$  and  $M$ . Therefore, since  $|M^*| > |M|$ , there must be at least one path in  $Q$  that contains more edges from  $M^*$  than  $M$ . Such a path is  $M$ -augmenting. ■

### 3.2 Alternating Trees

Let  $M$  be a current matching and  $X$  the set of  $M$ -exposed nodes.

**Definition 3.5** (Alternating Tree)

An  $M$ -alternating tree is a tree  $T$  with root node  $r \in X$  such that along every path to a node  $v$ , the path is  $M$ -alternating with  $e_i \in M$  iff  $i = 2k$  for some  $k \in \mathbb{Z}^+$ .



**Figure 1:** An example of an alternating tree. Nodes in  $A$  are white while nodes in  $B$  are black.

We can divide an alternating tree  $T = (V_T, E_T)$  into two node sets  $A$  and  $B$  such that  $A$  is the set of nodes at the other end of an odd-length  $M$ -alternating path starting at  $r$  and  $B$  is the set of nodes at the other end of an even-length  $M$ -alternating path starting at  $r$ . Such sets can be built by the following relation: starting with  $A = \emptyset$  and  $B = \{r\}$ , if  $(u, v) \in E$  such that  $u \in B$  and  $v \notin A \cup B$  and there exists an edge  $(v, w) \in M$ , then  $A \leftarrow A \cup \{v\}$  and  $B \leftarrow B \cup \{w\}$ . Note that if there exists some edge  $(i, j)$  where  $i \in B$  and  $j \notin T$  such that  $j$  is  $M$ -exposed, then the path  $P = (r, v_1), \dots, (i, j)$  is  $M$ -augmenting.

### Definition 3.6

An  $M$ -alternating tree is *maximal* if for all  $b \in B$ ,  $N(b) \subseteq V_T$  (i.e. no additional nodes can be added to the tree). It is *frustrated* if for all  $b \in B$ ,  $N(b) \subsetneq A$ .

### 3.3 Maximum Matchings in the Bipartite Case

---

**Algorithm 2** Maximum Matching in Bipartite Graphs Algorithm

**Input:** A bipartite graph  $G = (L, R, E)$

**Output:** A matching  $M$

1:  $X \leftarrow V$ 

- ▷ The set of  $M$ -exposed nodes

## 4 Exercises

### 4.1 LP-Duality

#### Exercise 4.1.1

*The dual of the dual is the primal.*

---

*Solution.*

The dual (D) is equivalent to the minimization problem

$$\begin{aligned} & \text{minimize} && \sum_{i=1}^m -b_i y_i \\ & \text{subject to} && \sum_{i=1}^m -a_{ij} y_i \geq -c_j \quad j = 1, \dots, n \\ & && y_i \geq 0 \quad i = 1, \dots, m. \end{aligned}$$

Taking the dual of this LP gives

$$\begin{aligned} & \text{maximize} && \sum_{j=1}^n -c_j x_j \\ & \text{subject to} && \sum_{j=1}^n -a_{ij} x_j \leq -b_i \quad i = 1, \dots, m \\ & && x_j \geq 0 \quad j = 1, \dots, n \end{aligned} \iff \begin{aligned} & \text{minimize} && \sum_{j=1}^n c_j x_j \\ & \text{subject to} && \sum_{j=1}^n a_{ij} x_j \geq b_i \quad i = 1, \dots, m \\ & && x_j \geq 0 \quad j = 1, \dots, n \end{aligned}$$

which is the primal (P). ■

## References

- [1] V. V. Vazirani. *Approximation Algorithms*. Springer, 2001.
- [2] D. P. Williamson and D. B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.