CHAPTER

11

RANDOMIZED APPROXIMATION ALGORITHMS IN COMBINATORIAL OPTIMIZATION

Rajeev Motwani Joseph (Seffi) Naor Prabhakar Raghavan

Randomization has proved to be a powerful technique in finding approximate solutions to difficult problems in combinatorial optimization. In this chapter, we restrict ourselves to approximation algorithms that are efficient and provably good. The focus of this chapter is the use of randomized rounding. In this approach, one solves a relaxation of a problem in combinatorial optimization, and then uses randomization to return from the relaxation to the original optimization problem. Two kinds of relaxations of difficult combinatorial problems are considered: linear programming relaxations and semidefinite programming relaxations. A number of concrete applications are given.

11.1 INTRODUCTION

The last few years have witnessed the proliferation of randomization in approximation algorithms for difficult combinatorial problems. Probabilistic search techniques such as simulated annealing [LA87] (see also Section 12.6) have enjoyed considerable success in solving large instances of a variety of combinatorial problems. In this chapter we study approximation algorithms that are *efficient*, in that their running times are (provably) bounded by a polynomial in the size of the input; and *provably good*, in that the solution produced by the algorithm is guaranteed to be close to the optimal solution to within a specified, provable tolerance. Further, we insist that these guarantees hold for every instance of the problem being solved; the only randomness in the performance guarantee stems from the randomization in the algorithm itself, and not due to any probabilistic assumptions on the instance.

Among such provably good and efficient randomized algorithms there have been two major areas of success. The first is that of approximation algorithms for combinatorial problems such as graph coloring, multicommodity flow, and finding large cuts in a graph. Additional algorithms for the multicommodity problem and cuts are described in Chapter 5. This will be the focus of this chapter. The second area, which is covered in the chapter on approximate counting in this book (Chapter 12), is that of approximately counting the number of distinct solutions to an instance of a combinatorial problem.

All the algorithms we will discuss follow a common paradigm. They first formulate the problem as an integer program. Next, some constraints in this integer program are relaxed in order that the relaxation be efficiently solvable. Finally, randomization is used to restore the relaxed constraints.

Throughout this chapter, $\boldsymbol{E}[X]$ will denote the expectation of a random variable X, $\boldsymbol{P}[A]$ will denote the probability of an event A, and $\bar{A}$ will denote the complement of event A. We refer the reader to the book by Motwani and Raghavan [MR95] for an introduction to the area of randomized algorithms.

We will consider optimization problems where we seek to either minimize or maximize the value of an objective function $V(I)$, for a given input instance I. Let $V_A(I)$ be the value of the objective function for the solution delivered by an algorithm A for an input instance I, and let $V_*(I)$ denote the optimal value of the objective function for an instance I. For minimization problems, the performance ratio of an algorithm A is the supremum over all I of the ratio $V_A(I)/V_*(I)$; similarly, for maximization problems, the performance ratio of an algorithm A is the infimum over all I of the ratio $V_A(I)/V_*(I)$. We say that an algorithm A is an α-approximation algorithm if it has performance ratio at most α for minimization problems, and performance ratio at least α for maximization problems. In the case of randomized approximation algorithms, we replace $V_A(I)$ by $\boldsymbol{E}[V_A(I)]$ in the definition of performance ratio, where the expectation is taken over the random choices made by the algorithm. In general, the term *approximation algorithm* will denote a *polynomial-time* algorithm.

In Section 11.2 we study linear programming relaxations using an integer multicommodity flow problem, covering and packing problems, multicut problems, and the maximum satisfiability problem as illustrations. In Section 11.3 we study the application of semidefinite programming relaxations to maximum cuts in a graph and to graph coloring. We give a number of ways in which these results can be extended, and the results of some implementations, in Section 11.4.

ROUNDING LINEAR PROGRAMS

11.2

In this section we discuss approximations obtained using the linear programming relaxation of the integer program formulation of an optimization problem. A convenient abstraction for beginning this study is the *lattice-approximation problem*. After studying this problem in the abstract, we will point out its connections to linear programming relaxations and to approximation algorithms in a number of concrete settings. In the lattice approximation problem we are given an $n \times n$ matrix $\mathbf{A}$, all of whose entries are 0 or 1. In addition, we are given a column vector $\mathbf{p}$ with n entries, all of which are in the interval $[0, 1]$. We wish to find a column vector $\mathbf{q}$ with n entries, all of which are from the set $\{0, 1\}$, so as to minimize $\| \mathbf{A} \cdot (\mathbf{p} - \mathbf{q}) \|_\infty$. (Please refer to problem [LAP] in the Glossary). We think of the vector $\mathbf{q}$ as an "integer approximation" to the given real vector $\mathbf{p}$, in the sense that $\mathbf{A} \cdot \mathbf{q}$ is close to $\mathbf{A} \cdot \mathbf{p}$ in every component. Below, we apply the technique of *randomized rounding* to this problem. Following this, we give two instructive applications of the technique in detail, and finally, mention a number of other interesting applications.

One solution to this problem is *thresholding:* for all i, if $p_i \geq 0.5$ set q_i to 1, else set it to 0. Thresholding may yield a vector $\mathbf{q}$ for which $\| \mathbf{A} \cdot (\mathbf{p} - \mathbf{q}) \|_\infty$ may be as large as order of $\| \mathbf{A} \cdot \mathbf{q} \|_\infty$. Consider the following randomized rounding scheme for determining the components of $\mathbf{q}$: for each i, independently set q_i to 1 with probability p_i, and to 0 otherwise. Then, letting $\mathbf{A}_i$ denote the ith row of $\mathbf{A}$, we have $\boldsymbol{E}\big[\mathbf{A}_i \cdot \mathbf{q}\big] = \boldsymbol{E}\big[\mathbf{A}_i \cdot \mathbf{p}\big]$ by linearity of expectation (this does not require the entries of $\mathbf{A}$ to be 0-1). We now argue that for all i, $\mathbf{A}_i \cdot \mathbf{q}$ is likely to remain close to $\mathbf{A}_i \cdot \mathbf{p}$ after randomized rounding. To this end we invoke the following bounds (commonly known as *Chernoff bounds* [MR95]) on the sum of independent 0-1 (or, *Bernoulli*) random variables.

LEMMA 11.1 Let $X_1, \ldots, X_n$ be a sequence of independent Bernoulli trials such that $\boldsymbol{P}[X_i = 1] = p_i$ and $\boldsymbol{P}[X_i = 0] = 1 - p_i$. Let S be any subset of the integers $1, \ldots, n$, and let $s = |S|$. Define $Y = \sum_{i \in S} X_i$, so that $\boldsymbol{E}[Y] = \sum_{i \in S} p_i$. Then,

$$\boldsymbol{P}\Big[|Y - \boldsymbol{E}[Y]| > \sqrt{4s \ln s}\Big] \leq \frac{1}{s^2}.$$

The following alternative version of the Chernoff bound [MR95] will also be useful in the sequel:

LEMMA 11.2 Let $X_1, \ldots, X_n$ be a sequence of independent Bernoulli trials such that $\boldsymbol{P}[X_i = 1] = p_i$ and $\boldsymbol{P}[X_i = 0] = 1 - p_i$. Define $Y = \sum X_i$, so that $\boldsymbol{E}[Y] = \sum p_i$. Then, for $\beta \in [0, 1]$,

$$\boldsymbol{P}[|Y - \boldsymbol{E}[Y]| > \beta\, \boldsymbol{E}[Y]] \leq 2\exp(-0.38\beta^2\, \boldsymbol{E}[Y]).$$

By Lemma 11.1, we know that

$$\boldsymbol{P}\Big[|\mathbf{A}_i \cdot \mathbf{q} - \mathbf{A}_i \cdot \mathbf{p}| > \sqrt{4n \ln n}\Big] < \frac{1}{n^2}.$$

Now,

$$\begin{aligned}
\boldsymbol{P}\Big[\| \mathbf{A} \cdot (\mathbf{p} - \mathbf{q}) \|_\infty > \sqrt{4n \ln n}\Big] &= \boldsymbol{P}\Big[\cup_{i=1}^{n} |\mathbf{A}_i \cdot \mathbf{q} - \mathbf{A}_i \cdot \mathbf{p}| > \sqrt{4n \ln n}\Big] \\
&\leq \sum_{i=1}^{n} \boldsymbol{P}\Big[|\mathbf{A}_i \cdot \mathbf{q} - \mathbf{A}_i \cdot \mathbf{p}| > \sqrt{4n \ln n}\Big] \\
&< \frac{1}{n}.
\end{aligned}$$

We thus have:

THEOREM 11.1 For every instance $\{\mathbf{A}, \mathbf{p}\}$ of the lattice approximation problem, randomized rounding finds a solution $\mathbf{q}$ such that $\| \mathbf{A} \cdot (\mathbf{p} - \mathbf{q}) \|_\infty \leq \sqrt{4n \ln n}$, with probability at least $1 - \frac{1}{n}$.

Using slightly stronger versions of the Chernoff bound than Lemma 11.1 [Rag88, RT87, SSS95], it is possible to get slightly sharper results than that in Theorem 11.1. However, in the worst case, we know of no efficient algorithm that can find a solution $\mathbf{q}$ such that $\| \mathbf{A} \cdot (\mathbf{p} - \mathbf{q}) \|_\infty$ is $o(\sqrt{4n \ln n})$ on every instance.

The *set-balancing problem* [Spe85, Spe87] is a special case of the lattice-approximation problem, in which every entry of $\mathbf{p}$ is 0.5. (Please refer to problem [SB] in the Glossary). This problem has a rich history in combinatorics, and additionally has a direct application to a problem in integrated circuit design [DR89]. Gao and Kaufmann [GK87] show that a closely related problem arises in the solution of channel-routing problems in VLSI. Theorem 11.1 implies a solution to the set-balancing problem; directions for improving this solution will be suggested later in Section 11.4.3.

What does the lattice approximation problem have to do with linear programs and approximation algorithms? To answer this question, we define the general framework in which randomized rounding is used. Given a combinatorial optimization problem, we first formulate it as a zero-one integer linear program, if possible. We do not know of good ways of solving large zero-one linear programs in general, so we first relax the integrality constraints, and solve the resulting linear program. The linear program can be solved by any one of several algorithms with proven theoretical and/or practical efficiency. Let $\mathbf{x}$ denote the vector of variables in the integer linear program formulation, and let $\widehat{\mathbf{x}}$ denote the vector of solutions resulting from the linear program. We now apply randomized rounding to restore integrality to the variables: for each i, independently set $\mathbf{x}_i$ to 1 with probability $\widehat{\mathbf{x}}_i$, and to 0 otherwise. Let $\bar{\mathbf{x}}_i$ denote the resulting 0-1 vector of solutions to the integer program. Let $\mathbf{a}$ denote a row of the coefficient matrix of the linear program formulation. By linearity of expectation, $E[\mathbf{a} \cdot \bar{\mathbf{x}}] = \mathbf{a} \cdot \widehat{\mathbf{x}}$. This suggests that the expected value of the left-hand side of any constraint in the linear program will satisfy the bound prescribed by the right-hand side. We note that in certain instances (e.g., maximum satisfiability), better results can be obtained by rounding a variable $\mathbf{x}$ to 1 with probability $f(\widehat{\mathbf{x}})$, where f is a function that maps $\widehat{\mathbf{x}}$ to a value in $[0, 1]$.

We next illustrate the use of this technique in three settings — integer multicommodity flows, covering problems, and maximum satisfiability. These applications have been chosen because they are especially illustrative; a number of others are listed in Section 11.2.4.

11.2.1 THE INTEGER MULTICOMMODITY FLOW PROBLEM

We are given a directed network (V, E) with a set V of nodes and a set E of arcs. Let m denote $|E|$ and n denote $|V|$. Associated with each each arc $e \in E$ is a *capacity*, denoted $c(e)$. In an instance of the *multicommodity flow problem*, (see also problem [Multicom] in the Glossary), we are given a set of k triplets (s_i, t_i, d_i). In each triplet, s_i and t_i are nodes in the network, while d_i is a positive integer *demand*. A solution is a set of *flows*, denoted $f_i(e)$, where we think of $f_i(e)$ as the amount of commodity i that flows through arc e. The flows are subject to the following restrictions:

I. All the flows $f_i(e)$ should be integral.

II. The incoming and outgoing flows of each commodity at each node should obey conservation constraints.

III. Capacity constraints: for all e, $\sum_i f_i(e) \le c(e)$.

IV. Demand constraints: we must ship d_i units of commodity i from s_i to t_i. In our notation, we write $\sum_{e \in A_i} f_i(e) \ge d_i$, where A_i is the set of arcs leaving s_i.

As stated above, it is possible for the problem to be infeasible — the capacities available may not suffice to sustain the demands. Consider the following optimization version of the problem, for which we seek approximation algorithms. Given the instance, we seek to maximize the total flow of all commodities, which is clearly $\sum_i \sum_{e \in A_i} f_i(e)$. Thus, some commodities may not have their demands satisfied in the process of this maximization.

We now proceed to discuss the case $d_i = 1$ for all i; the same technique applies to the general case $d_i \ge 1$ with minor modifications. Clearly this maximization problem can be written as a 0-1 linear program. By relaxing the integrality constraint $f_i(e) \in \{0, 1\}$ for all i, e, we obtain the relaxation linear program in which $f_i(e) \in [0, 1]$ for all i, e. This linear program can be solved efficiently; in fact, one can use more efficient and direct combinatorial algorithms [KST90, PST91]; see also Chapter 5 in this book. We solve instead a slightly modified linear program, one in which we set the capacity of every arc e to $(1 - \epsilon)c(e)$ for a positive constant $\epsilon < \frac{\sqrt{5}-1}{2}$. We solve this modified linear program to obtain the fractional solutions $\widehat{f_i}(e)$, together with the total flow $\widehat{F} = \sum_i \sum_{e \in A_i} f_i(e)$. Let F denote the value of the total flow in the optimal *integral* solution. Then, $\widehat{F} \ge F(1 - \epsilon)$, and the fractional total flow in any arc is at most $(1 - \epsilon)$ times its capacity. Randomized rounding is now invoked as follows: for each commodity i, we independently perform a random walk from s_i to t_i that is guided by the fractional solutions $\widehat{f_i}(e)$. The random walk for commodity i is as follows: we begin at s_i by flipping a coin with probability of heads equal to $\sum_{e \in A_i} \widehat{f_i}(e) \le 1$. If the coin comes up heads, we proceed with the random walk as described below; if not, we assign zero flow to commodity i. We now describe the general step of the random walk: suppose that we are at a node v. Let $A(v)$ denote the set of arcs leaving v. The random walk then chooses to proceed along arc $a \in A(v)$ with probability $\frac{\widehat{f_i}(a)}{\sum_{e \in A(v)} \widehat{f_i}(e)}$. The walk for commodity i terminates on reaching t_i, which it must in at most $n - 1$ steps because the set of arcs with non-zero flow may be assumed (without loss of generality) to form a directed acyclic graph. The following lemma is easy to prove by induction:

LEMMA 11.3 The probability that the random walk for commodity i traverses any arc e is equal to $\widehat{f_i}(e)$.

We are now ready for the main result.

THEOREM 11.2 Suppose that we have an instance of multicommodity flow in which every capacity $c(e)$ is at least $5.2 \ln 4m$. Let ϵ be a positive constant less than $\frac{\sqrt{5}-1}{2}$. Then, with probability $1 - \frac{1}{m}$, the above algorithm yields a integer solution of total flow at least $F(1-\epsilon)^2$ with probability at least $1 - \frac{1}{m} - 2\exp(-0.38\epsilon^2 F)$, where F is the total flow in the optimal integer solution.

Proof. The algorithm always produces an integer solution, so it can only fail to meet the guarantees of the theorem because of the following types of failure:

I. The rounded flow violates the capacity constraint on some arc.

II. The rounded flow is of value less than $F(1-\epsilon)^2$.

We proceed to bound the probability of the first mode of failure by $\frac{1}{m}$ and the second by $2\exp(-0.38\epsilon^2 F)$. Adding together these probabilities yields the theorem.

The crucial observation is that the event "the random walk for commodity i traverses arc e" is a Bernoulli trial, for any fixed arc e. Likewise, the event "commodity i has nonzero flow in the solution" is also a Bernoulli trial. Then, after randomized rounding the flow in any arc is the sum of independent Bernoulli trials. Likewise, the total flow is a random variable that is the sum of independent Bernoulli trials. We have already observed that the expectation of the total flow after rounding is at least $F(1-\epsilon)$. Applying Lemma 11.2 with $\beta = \epsilon$ now yields the probability bound for the second mode of failure. We turn now to the flow through a fixed arc of the network, following randomized rounding. Once again, Lemma 11.2 yields that the probability that the capacity constraint of a fixed arc is violated is at most $\frac{1}{m^2}$; we omit the detailed but routine calculations. Summing this probability over all m arcs, we have the desired bound. ∎

Note that the failure probability of the algorithm can be diminished by independent repetitions. In most applications, it is the computation of the fractional solutions $\widehat{f_i}(e)$ that is computationally intensive. Randomized rounding is itself quite fast, and can be repeated many times.

11.2.2 COVERING AND PACKING PROBLEMS

Covering and packing integer programs are defined as follows. Let Z_+ denote the nonnegative integers. Let $\mathbf{A}$ be an $m \times n$ matrix over Z_+, let $\mathbf{b}$ be a vector over Z_+^m, and let $\mathbf{x}$ and $\mathbf{w}$ be vectors over Z_+^n. The *covering problem* is to minimize $\mathbf{w}^T \cdot \mathbf{x}$ subject to $\mathbf{Ax} \geq \mathbf{b}$. The *packing problem* is to maximize $\mathbf{w}^T \cdot \mathbf{x}$ subject to $\mathbf{Ax} \leq \mathbf{b}$. (Please refer to problems [PA] and [CO] in the Glossary). In a linear programming setting, covering and packing are dual problems. Randomized rounding is a very useful technique for approximating both covering and packing problems. We will present this technique in the context of an important special case of covering problems, the *set cover* problem. Please refer to Chapter 3 for an extensive discussion of this problem. In Section 11.2.2.1 we present an application of set cover to the undirected multicut problem.

Let $V = \{v_1, \ldots, v_n\}$ be a set of elements, and let $\mathcal{S} = \{S_1, \ldots, S_m\}$ be a family of subsets defined over V. There are two common ways of formulating the set cover problem, and the reader can easily verify that they are equivalent. (See also the Glossary, problem [SC]). In the first formulation, a non-negative weight function w is attached to V. The goal is to find a minimum weight set of elements $V' \subseteq V$ that intersects all subsets $S \in \mathcal{S}$. The weight of V' is defined to be the sum of the weights of the elements belonging to V'. In the second formulation, a non-negative weight function w is attached to $\mathcal{S}$, and the goal is to find a minimum weight family of subsets $\mathcal{S}' \subseteq \mathcal{S}$, such that their union is equal to V. We will henceforth use the first formulation (which is also known as the *hitting set*) of the problem.

We first write the set cover problem as an integer program. For $1 \leq i \leq n$, let x_i denote the indicator variable for element v_i.

$$\begin{aligned} &\text{Minimize } \sum_{i=1}^{n} w_i \cdot x_i \\ &\text{subject to } \sum_{i:\, v_i \in S_j} x_i \geq 1 \qquad \forall S_j \in \mathcal{S}, \\ &\qquad x_i \in \{0,1\} \qquad 1 \leq i \leq n. \end{aligned} \tag{11.1}$$

We relax this integer program and allow each variable x_i to assume values in the interval $[0,1]$. Let $\widehat{x_i}$ be the value assigned to x_i in an optimal fractional solution of the relaxed program. We now apply randomized rounding and set each variable x_i to 1 with probability $\widehat{x_i}$. Clearly, the expected weight of the elements chosen to the cover is equal to $\sum_{i=1}^{n} w_i \widehat{x_i}$.

What is the probability that a subset S_i is covered? Suppose that S_i contains elements $v_1, \ldots, v_k$; recall that $\sum_{j=1}^{k} \widehat{x_j} \geq 1$. The probability that S_i is covered is:

$$1 - \prod_{j=1}^{k} (1 - \widehat{x_j}) \geq 1 - \left(1 - \frac{1}{k}\right)^k \geq 1 - \frac{1}{e}. \tag{11.2}$$

That is, the probability that subset S_i is covered is at least a constant. To increase the probability of covering the family of subsets $\mathcal{S}$, randomized rounding can be repeatedly applied to the set of variables that were not set to 1. For example, by repeating the randomized rounding procedure $t = O(\log m)$ times, we can guarantee that the probability that a subset S_j is not covered is at most $\frac{1}{2m}$. Thus, the probability that $\mathcal{S}$ is not covered after t repetitions is at most $\frac{1}{2}$. The expected weight of the cover after t repetitions is at most $t \cdot \sum_{i=1}^{k} w_i \widehat{x_i}$, i.e., t times the weight of an optimal fractional cover. In fact, repeating the randomized rounding procedure t times is the same as scaling the original probabilities and rounding each variable x_i as follows:

$$\boldsymbol{P}[x_i = 0] = (1 - \widehat{x_i})^t.$$

We thus have:

THEOREM 11.3 For every instance $\{V, \mathcal{S}\}$ of the set cover problem, randomized rounding finds an $O(\log m)$-approximate cover, with probability at least $\frac{1}{2}$.

The result obtained in Theorem 11.3 is not the best possible. It is well known that the approximation factor obtained by the greedy algorithm can be at most $1 + \ln \Delta$, (please refer to Theorem 3.1 in Chapter 3), where Δ denotes the maximum degree of an element in V, i.e., the maximum number of subsets in $\mathcal{S}$ that can be covered by a single element in V. We will now show that the bound obtained in Theorem 11.3 can be improved. We follow the work of Bertsimas and Vohra [BV94] and Srinivasan [Sri95, Sri96].

Let A_i denote the event that subset S_i is not covered. We claim that events A_i and A_j are positively correlated. To verify this, observe that

$$\begin{aligned} \boldsymbol{P}[A_i \cap A_j] &= \prod_{\ell \in S_i - S_j} \boldsymbol{P}[x_\ell = 0] \cdot \prod_{\ell \in S_j - S_i} \boldsymbol{P}[x_\ell = 0] \cdot \prod_{\ell \in S_i \cap S_j} \boldsymbol{P}[x_\ell = 0] \\ &\geq \boldsymbol{P}[A_i] \cdot \boldsymbol{P}[A_j], \end{aligned}$$

implying also that $\boldsymbol{P}[\bar{A}_i \cap \bar{A}_j] \geq \boldsymbol{P}[\bar{A}_i] \cdot \boldsymbol{P}[\bar{A}_j]$. These inequalities are actually implied by the FKG Inequality [AS92], which also implies the more general inequality for any subset J of indices:

$$\boldsymbol{P}\left[\bigcap_{j \in J} A_j\right] \geq \prod_{j \in J} \boldsymbol{P}[A_j].$$

Suppose that randomized rounding is invoked with some scaling parameter t whose value will be determined later. We are interested in evaluating the expected weight of a solution produced by randomized rounding, given that all subsets are satisfied. Let $F = \cap_{i=1}^m \bar{A}_i$; then,

$$\begin{aligned} \boldsymbol{E}\left[\sum_{i=1}^n w_i x_i | F\right] &= \sum_{i=1}^n w_i \cdot \boldsymbol{P}[x_i = 1 | F] \\ &= \sum_{i=1}^n w_i \cdot \frac{\boldsymbol{P}[F | x_i = 1]}{\boldsymbol{P}[F]} \cdot \boldsymbol{P}[x_i = 1]. \end{aligned} \tag{11.3}$$

For each element v_i, let $N(i)$ denote the indices of the subsets in which v_i is contained, and let d_i denote the degree of v_i, i.e., $d_i = |N(i)|$. Then,

$$\frac{\boldsymbol{P}[F | x_i = 1]}{\boldsymbol{P}[F]} = \frac{\boldsymbol{P}\left[\cap_{j \notin N(i)} \bar{A}_j\right]}{\boldsymbol{P}\left[\cap_{j=1}^m \bar{A}_j\right]} \leq \frac{1}{\boldsymbol{P}\left[\cap_{j \in N(i)} \bar{A}_j\right]},$$

where the latter inequality follows from the positive correlation of the events $\{\bar{A}_i\}$. We then obtain the following inequality from (11.2) and from the positive correlation of the events $\{\bar{A}_i\}$.

$$\boldsymbol{P}\left[\cap_{j \in N(i)} \bar{A}_j\right] \geq \prod_{j \in N(i)} \boldsymbol{P}[\bar{A}_j] \geq (1 - e^{-t})^{d_i}.$$

Recall that $\Delta = \max_{i=1}^m d_i$. Substituting back in (11.3), we obtain:

$$\begin{aligned} \boldsymbol{E}\left[\sum_{i=1}^n w_i x_i | F\right] &\leq \sum_{i=1}^n w_i (1 - e^{-t})^{-d_i} \cdot (1 - (1 - \widehat{x}_i)^t) \\ &\leq (1 - e^{-t})^{-\Delta} \sum_{i=1}^n w_i (1 - (1 - t\widehat{x}_i)) \\ &= t(1 - e^{-t})^{-\Delta} \sum_{i=1}^n w_i \widehat{x}_i. \end{aligned}$$

Choosing $t = O(\log \Delta)$ we get

$$\boldsymbol{E}\left[\sum_{i=1}^{n} w_i x_i | F\right] = O\left(\log \Delta \cdot \sum_{i=1}^{n} w_i \widehat{x_i}\right).$$

This proof shows that with positive probability, randomized rounding can yield an $O(\log \Delta)$ approximation factor to the set cover problem. However, the probability of success can still be exponentially small. Thus, from the algorithmic point of view, achieving an $O(\log \Delta)$ approximation factor with a randomized algorithm still remains open. Stronger results for covering and packing problems that use the positive correlation of the events $\{A_i\}_{i=1}^{m}$ were shown by Srinivasan [Sri95, Sri96], who also showed how to de-randomize his existential proof by constructing appropriate pessimistic estimators. He obtains the following bound for covering problems:

$$1 + O\left(\max\left\{\frac{\ln\left(\frac{mB}{y^*}\right)}{B}, \sqrt{\frac{\ln\left(\frac{mB}{y^*}\right)}{B}}\right\}\right),$$

where B is defined to be the minimum entry in the vector b and $y^* = \sum_{i=1}^{n} w_i \widehat{x_i}$. This improves on the bound

$$1 + O\left(\max\left\{\frac{\ln m}{B}, \sqrt{\frac{\ln m}{B}}\right\}\right),$$

which is obtained from the (standard) application of randomized rounding to covering problems, e.g., Theorem 11.3 in the case of set cover.

For the unweighted set cover problem, Srinivasan [Sri95] (see also [Sri96]) obtains the following approximation bound:

$$\ln\left(\frac{m}{y^*}\right) + O\left(\ln\ln\left(\frac{m}{y^*}\right)\right) + O(1),$$

which is at least as good as $O(\log \Delta)$.

11.2.2.1 The undirected multicut problem

The undirected multicut problem is defined as follows. Let $G = (V, E)$ be an undirected graph, $c : E \to \mathbb{R}^+$ a capacity function, and let (s_i, t_i), $1 \le i \le k$, be k source-sink pairs. A multicut is a set of edges that separates each source from its corresponding sink, i.e, the removal of a multicut disconnects the graph into connected components such that no source-sink pair is contained in the same connected component. Finding a minimum capacity multicut is an *NP*-complete problem. Garg, Vazirani, and Yannakakis [GVY93] gave a polynomial-time algorithm that approximates the minimum capacity multicut (in an undirected graph) to within a factor of $O(\log k)$. This algorithm is described in detail in Section 5.2.2. We show a set cover formulation of the multicut problem that yields an $O(\log k)$ approximation factor via randomized rounding. We follow the work of Bertsimas and Vohra [BV94].

We first note that the minimum capacity multicut problem can be formulated as a set cover problem in a natural way. Let the elements in the set cover formulation correspond to the edges in the graph and let each source-sink path define a subset. We are looking for a minimum weight set of edges that intersects each source-sink path. However, the approximation factor obtained from this set cover formulation is quite weak, and can be as bad as $\Omega(|V|)$. We now define a different set cover formulation of the multicut problem.

Let F be the collection of all subsets of V with the property that for any $S \in F$, *at most* one of s_i and t_i belong to S, for all $1 \le i \le k$. We denote by $\delta(S)$ the set of edges for which exactly one endpoint is in S. Let $c(A)$ denote the sum of the capacities of the edges in a set A. Let a_{iS} (b_{iS}) be the indicator variable of the event $a_i \in S$ ($t_i \in S$). Let $x(S) = 1$ mean that subset S is picked, $x(S) = 0$ otherwise. We are now ready to define the following integer program:

$$\begin{aligned} \text{Minimize } & \frac{1}{2}\sum_{S\in F} c(\delta(S)) \cdot x(S) \\ \text{subject to } & \sum_{S\in F} a_{iS} \cdot x(S) \ge 1 \qquad \forall s_i,\ 1 \le i \le k, \\ & \sum_{S\in F} b_{iS} \cdot x(S) \ge 1 \qquad \forall t_i,\ 1 \le i \le k, \\ & x(S) \in \{0,1\}, \qquad \forall S \in F. \end{aligned} \tag{11.4}$$

In words, the goal here is to cover all sources and sinks in the graph by subsets belonging to F. Given any feasible multicut C, it is not hard to see that the connected components generated by removing the edges of C induce a feasible solution to the integer program. Conversely, from any feasible solution to the integer program, a feasible multicut can be computed.

Bertsimas and Vohra [BV94] show that a linear relaxation of this integer program, where each variable $x(S)$ is allowed to assume values in the range $[0, 1]$, can be computed in polynomial time, albeit the number of variables is exponential. This follows by observing that there exists a relaxed optimal solution in which the number of variables that have non-zero variables is bounded by a polynomial. Given an optimal fractional solution, we can now apply the set cover randomized rounding procedure to this solution, and obtain a feasible multicut from it. The approximation factor obtained is $O(\log k)$, since the number of elements that need to be covered is $2k$. We note that a greedy algorithm that finds an approximate multicut, and uses the above set cover formulation of the multicut problem, was given by Cheriyan and Yu [CY95].

11.2.3 THE MAXIMUM SATISFIABILITY PROBLEM

The satisfiability problem (SAT) is defined as follows: given a set of clauses in conjunctive normal form over a collection of boolean variables, we wish to decide whether there is an assignment of the variables that satisfies all the clauses. In the MAX SAT problem, we are given an instance of satisfiability and we seek an assignment that *maximizes* the number of satisfied clauses. Since SAT is known to be *NP*-hard, it immediately follows that MAX SAT is also *NP*-hard. Given an instance I, let $M_*(I)$ be the maximum number of clauses that can be satisfied, and let $M_A(I)$ be the number of clauses satisfied by an algorithm A. Recall that the performance ratio of algorithm A is the infimum (over all instances I) of $\frac{M_A(I)}{M_*(I)}$, and that for a randomized algorithm A, the quantity $M_A(I)$ may be a random variable, in which case we replace $M_A(I)$ by $\boldsymbol{E}[M_A(I)]$ in the definition of the performance ratio. Thus, our definition requires us to satisfy a number of clauses close to the best possible for the instance at hand.

We now give a simple randomized algorithm that achieves a performance ratio of $\frac{3}{4}$. We note that the techniques described here can easily be generalized to yield the same performance ratio for the weighted MAX SAT problem, where a non-negative weight is attached to each clause, and the goal is to find an assignment that maximizes the weight of the satisfied clauses.

Consider setting each variable in the instance independently to 1 with probability $\frac{1}{2}$. Clearly, a clause containing k literals is not satisfied by this process with probability $\frac{1}{2^k}$; thus, such a random assignment would work well for instances in which every clause contains many literals: if every clause were to contain k or more literals, we immediately have an α-approximation algorithm for which $\alpha \geq 1 - 2^{-k}$. This idea is implicit in early work of Johnson [Joh74]. It follows that we have a randomized $\frac{3}{4}$-approximation algorithm for instances of MAX SAT in which every clause has at least two literals. It appears that the bottleneck for achieving a performance ratio of $\frac{3}{4}$ stems from clauses consisting of a single literal. We now give a different algorithm that performs especially well when there are many clauses consisting of single literals. This algorithm is due to Goemans and Williamson [GW94a].[1] We then argue that on any instance, one of these two algorithms will yield a (randomized) $\frac{3}{4}$-approximation. Thus, given an instance, we run both algorithms and take the better of the two solutions. In Section 11.3 we will describe a general technique due to Goemans and Williamson [GW94b] that achieves an improved approximation ratio.

The idea again is to formulate the problem as an integer linear program, solve the linear programming relaxation, and then to round using the randomized rounding. To each clause C_j in the instance, we associate an indicator variable $z_j \in \{0, 1\}$ in the integer linear program that is 1 when C_j is satisfied and 0 otherwise. For each variable x_i, we use an indicator variable y_i that is 1 if the variable x_i is set TRUE, and 0 otherwise. Let S_j^+ be the set of variables that appear in the uncomplemented form in clause S_j, and S_j^- be the set of variables that appear in the complemented form in clause C_j. We may then formulate the MAX SAT problem as follows:

$$\begin{aligned} &\text{Maximize } \sum_j z_j \\ &\text{subject to } \sum_{S_j^+} y_i + \sum_{S_j^-} (1 - y_i) \geq z_j \qquad \forall j, \\ &\qquad\qquad y_i, z_j \in \{0, 1\} \qquad \forall i, j. \end{aligned} \tag{11.5}$$

We solve the linear program in which we relax the integrality constraints (11.5), allowing the y_i and the z_j to assume values in the interval $[0, 1]$. Let $\widehat{y_i}$ be the value assigned to y_i in the optimal solution to this linear program, and let $\widehat{z}_j$ be the value assigned to z_j. Clearly, $\sum_j \widehat{z}_j$ is an upper bound on the number of clauses that can be satisfied. We first show that using randomized rounding, we obtain a solution in which the expected number of clauses satisfied is at least $(1 - \frac{1}{e}) \sum_j \widehat{z}_j$.

[1]Prior to the work of Goemans and Williamson, Yannakakis [Yan92] had given a deterministic polynomial-time $\frac{3}{4}$-approximation algorithm for MAX SAT.

Randomized rounding independently sets the variable y_i to 1 (corresponding to x_i being set TRUE) with probability $\widehat{y}_i$. Where k is a positive integer, let β_k denote $1-(1-\frac{1}{k})^k$. We first show that for a clause C_j with k literals, the probability that it is satisfied by randomized rounding is at least $\beta_k\widehat{z}_j$. Because $\beta_k \geq (1-\frac{1}{e})$, the expected number of clauses satisfied by randomized rounding is at least $(1-\frac{1}{e})\sum_j \widehat{z}_j$.

LEMMA 11.4 Let C_j be a clause with k literals. The probability that it is satisfied by randomized rounding is at least $\beta_k\widehat{z}_j$.

Proof. Consider a single clause C_j. We may assume without loss of generality that all the variables contained in it appear in uncomplemented form, and that C_j is of the form $x_1 \vee \cdots \vee x_k$. By constraint (11.5) in the linear program,

$$\widehat{y}_1 + \cdots + \widehat{y}_k \geq \widehat{z}_j.$$

Clause C_j remains unsatisfied by randomized rounding only if every one of the variables y_i is rounded to 0. Since each variable is rounded independently, this occurs with probability $\prod_{i=1}^{k}(1-\widehat{y}_i)$. To show that

$$1-\prod_{i=1}^{k}(1-\widehat{y}_i) \geq \beta_k\widehat{z}_j,$$

we observe that the expression on the left is minimized when $\widehat{y}_i = \frac{\widehat{z}_j}{k}$ for all i. It now suffices to show that $1-(1-\frac{z_j}{k})^k \geq \beta_k z_j$ in $[0, 1]$; this follows from elementary calculus. ∎

THEOREM 11.4 Given an instance of MAX SAT, the expected number of clauses satisfied by linear programming and randomized rounding is at least $(1-\frac{1}{e})$ times the maximum number of clauses that can be satisfied on that instance.

We have studied two randomized algorithms MAX SAT: one that set each variable to 1 with probability $\frac{1}{2}$, and a second that used the solutions to the linear program as a basis for randomized rounding. We will now show by a simple convexity argument that on any instance, one of the algorithms is a $\frac{3}{4}$-approximation algorithm. Namely, given any instance, we run both algorithms and choose the better solution. Let n_1 denote the expected number of clauses that are satisfied when each variable is independently set to 1 with probability $\frac{1}{2}$. Let n_2 denote the expected number of clauses that are satisfied when we use the linear programming followed by randomized rounding (corresponding to Theorem 11.4). The following exercise will be used in the proof below.

EXERCISE 11.1 For $k \geq 1$, $\alpha_k = 1-\frac{1}{2^k}$ and $\beta_k = 1-(1-\frac{1}{k})^k$, show that

$$\alpha_k + \beta_k \geq \frac{3}{2}.$$

THEOREM 11.5 $\max\{n_1, n_2\} \geq \frac{3}{4}\sum_j \widehat{z}_j$.

Proof. We show that

$$\max\{n_1, n_2\} \geq \frac{n_1 + n_2}{2} \geq \frac{3}{4}\sum_j \widehat{z}_j.$$

The first inequality is immediate, we prove the second one. Let α_k denote $1 - \frac{1}{2^k}$, and let C^k denote the set of clauses that contain precisely k literals. Then,

$$n_1 = \sum_{k\geq 1}\sum_{C_j \in C^k} \alpha_k \geq \sum_{k\geq 1}\sum_{C_j \in C^k} \alpha_k \cdot \widehat{z}_j$$

since $0 \leq \widehat{z}_j \leq 1$. By Lemma 11.4,

$$n_2 \geq \sum_{k\geq 1}\sum_{C_j \in C^k} \beta_k \cdot \widehat{z}_j$$

Hence,

$$\frac{n_1 + n_2}{2} \geq \sum_{k\geq 1}\sum_{C_j \in C^k} \frac{\alpha_k + \beta_k}{2} \cdot \widehat{z}_j.$$

This inequality yields the theorem, since $\alpha_k + \beta_k \geq \frac{3}{2}$ for all $k \geq 1$ as proved in Exercise 11.1. ∎

11.2.4 RELATED WORK

There are many other interesting applications of randomized rounding; we mention a few of them. Naor and Roth [NR95] apply the technique to a file distribution problem in networks. Klein and Sairam [KS92] apply it to the computation of approximate shortest paths in a graph. Lin and Vitter [LV92] consider a family of packing problems. Kortsarz and Peleg [KP93] study the construction of a useful class of graphs known as *spanners*. The paper by Bertsimas and Vohra [BV94] is a comprehensive guide to uses of the technique in a variety of covering problems. One of the interesting ideas in their paper is to devise a function that maps each value in the linear program solution to a probability in $[0, 1]$, a generalization of the technique used by Raghavan and Thompson [RT87], and Goemans and Williamson [GW94a]; note that in the above examples, we have only used the obvious identity function. In some applications, as shown in [BV94], it helps to use other mappings from fractional solutions to probabilities. The interested reader is referred to these papers and the ones cited in them for further applications. Before proceeding to semidefinite programming, it is worth pointing out an alternative view of randomized rounding. Instead of thinking of the process as rounding a fraction $\hat{x}$ to 1 with probability $\hat{x}$, we may adopt the following equivalent view: We pick a random number y uniformly in $[0, 1]$. If $\hat{x} > y$ we round up to 1. Otherwise, we round down to 0. The effect is the same, but we now think of comparing the fraction $\hat{x}$ to a randomly chosen threshold. This view will be useful in the following section.