and $x(v)$ that would cause $n(u) = n(v)$, we create a disjunction of $x(u)$ and $x(v)$ that is false if it implies $n(u) = n(v)$; for example, if $x(u) =$ true and $x(v) =$ false implies that $n(u) = n(v)$, then we create a clause $(\overline{x(u)} \vee x(v))$. Since G is 3-colorable, given a correct coloring of S, there exists a setting of the variables $x(v)$ that satisfies all the clauses. Since each clause has two variables, it is possible to determine in polynomial time whether the instance is satisfiable or not; we leave it as an exercise to the reader to prove this (Exercise 6.3). Obviously if we find a setting of the variables that satisfies all constraints, this implies a correct coloring of the entire graph, whereas if the constraints are not satisfiable, our current coloring of S must not have been correct.

In Section 12.4, we'll revisit the idea from this section of drawing a small random sample of a graph and using it to determine the overall solution for the maximum cut problem in dense graphs.

Exercises

5.1 In the *maximum k-cut problem*, we are given an undirected graph $G = (V, E)$, and nonnegative weights $w_{ij} \geq 0$ for all $(i, j) \in E$. The goal is to partition the vertex set V into k parts $V_1, \ldots, V_k$ so as to maximize the weight of all edges whose endpoints are in different parts (i.e., $\max_{(i,j)\in E: i\in V_a, j\in V_b, a\neq b} w_{ij}$).
Give a randomized $\frac{k-1}{k}$-approximation algorithm for the MAX k-CUT problem.

5.2 Consider the following greedy algorithm for the maximum cut problem. We suppose the vertices are numbered $1, \ldots, n$. In the first iteration, the algorithm places vertex 1 in U. In the kth iteration of the algorithm, we will place vertex k in either U or W. In order to decide which choice to make, we will look at all the edges F that have the vertex k as one endpoint and whose other endpoint is $1, \ldots, k-1$, so that $F = \{(j, k) \in E : 1 \leq j \leq k-1\}$. We choose to put vertex k in U or W depending on which of these two choices maximizes the number of edges of F being in the cut.

(a) Prove that this algorithm is a 1/2-approximation algorithm for the maximum cut problem.

(b) Prove that this algorithm is equivalent to the derandomization of the maximum cut algorithm of Section 5.1 via the method of conditional expectations.

5.3 In the *maximum directed cut problem* (sometimes called MAX DICUT), we are given as input a directed graph $G = (V, A)$. Each directed arc $(i, j) \in A$ has nonnegative weight $w_{ij} \geq 0$. The goal is to partition V into two sets U and $W = V - U$ so as to maximize the total weight of the arcs going from U to W (that is, arcs (i, j) with $i \in U$ and $j \in W$).
Give a randomized $\frac{1}{4}$-approximation algorithm for this problem.

5.4 Consider the nonlinear randomized rounding algorithm for MAX SAT as given in Section 5.6. Prove that using randomized rounding with the linear function $f(y_i) = \frac{1}{2}y_i + \frac{1}{4}$ also gives a $\frac{3}{4}$-approximation algorithm for MAX SAT.

5.5 Consider the nonlinear randomized rounding algorithm for MAX SAT as given in Section 5.6. Prove that using randomized rounding with the piecewise linear function

$$f(y_i) = \begin{cases} \frac{3}{4}y_i + \frac{1}{4} & \text{for } 0 \leq y_i \leq \frac{1}{3} \\ 1/2 & \text{for } \frac{1}{3} \leq y_i \leq \frac{2}{3} \\ \frac{3}{4}y_i & \text{for } \frac{2}{3} \leq y_i \leq 1 \end{cases}$$

also gives a $\frac{3}{4}$-approximation algorithm for MAX SAT.

5.6 Consider again the maximum directed cut problem from Exercise 5.3.

(a) Show that the following integer program models the maximum directed cut problem:

$$\begin{aligned}
\text{maximize} \quad & \sum_{(i,j)\in A} w_{ij} z_{ij} \\
\text{subject to} \quad & z_{ij} \leq x_i, && \forall (i,j) \in A, \\
& z_{ij} \leq 1 - x_j, && \forall (i,j) \in A, \\
& x_i \in \{0,1\}, && \forall i \in V, \\
& 0 \leq z_{ij} \leq 1, && \forall (i,j) \in A.
\end{aligned}$$

(b) Consider a randomized rounding algorithm for the maximum directed cut problem that solves a linear programming relaxation of the integer program and puts vertex $i \in U$ with probability $1/4 + x_i/2$. Show that this gives a randomized $1/2$-approximation algorithm for the maximum directed cut problem.

5.7 In this exercise, we consider how to derandomize the randomized rounding algorithm for the set cover problem given in Section 1.7. We would like to apply the method of conditional expectations, but we need to ensure that at the end of the process we obtain a valid set cover. Let X_j be a random variable indicating whether set S_j is included in the solution. Then if w_j is the weight of set S_j, let W be the weight of the set cover obtained by randomized rounding, so that $W = \sum_{j=1}^{m} w_j X_j$. Let Z be a random variable such that $Z = 1$ if randomized rounding does not produce a valid set cover, and $Z = 0$ if it does. Then consider applying the method of conditional expectations to the objective function $W + \lambda Z$ for some choice of $\lambda \geq 0$. Show that for the proper choice of λ, the method of conditional expectations applied to the randomized rounding algorithm yields an $O(\ln n)$-approximation algorithm for the set cover problem that always produces a set cover.

5.8 Consider a variation of the maximum satisfiability problem in which all variables occur positively in each clause, and there is an additional nonnegative weight $v_i \geq 0$ for each Boolean variable x_i. The goal is now to set the Boolean variables to maximize the total weight of the satisfied clauses plus the total weight of variables set to be false. Give an integer programming formulation for this problem, with 0-1 variables y_i to indicate whether x_i is set true. Show that a randomized rounding of the linear program in which variable x_i is set true with probability $1 - \lambda + \lambda y_i^*$ gives a $2(\sqrt{2} - 1)$-approximation algorithm for some appropriate setting of λ; note that $2(\sqrt{2} - 1) \approx 0.828$.

5.9 Recall the maximum coverage problem from Exercise 2.11; in it, we are given a set of elements E, and m subsets of elements $S_1, \ldots, S_m \subseteq E$ with a nonnegative weight $w_j \geq 0$ for each subset S_j. We would like to find a subset $S \subseteq E$ of size k that maximizes the total weight of the subsets covered by S, where S covers S_j if $S \cap S_j \neq \emptyset$.

(a) Show that the following nonlinear integer program models the maximum coverage problem:

$$\begin{aligned}
\text{maximize} \quad & \sum_{j\in[m]} w_j \left(1 - \prod_{e\in S_j} (1 - x_e)\right) \\
\text{subject to} \quad & \sum_{e\in E} x_e = k, \\
& x_e \in \{0,1\}, \qquad \forall e \in E.
\end{aligned}$$

(b) Show that the following linear program is a relaxation of the maximum coverage problem:

$$\begin{aligned}
\text{maximize} \quad & \sum_{j \in [m]} w_j z_j \\
\text{subject to} \quad & \sum_{e \in S_j} x_e \geq z_j, && \forall j \in [m], \\
& \sum_{e \in E} x_e = k, \\
& 0 \leq z_j \leq 1, && \forall j \in [m], \\
& 0 \leq x_e \leq 1, && \forall e \in E.
\end{aligned}$$

(c) Using the pipage rounding technique from Exercise 4.7, give an algorithm that deterministically rounds the optimal LP solution to an integer solution and has a performance guarantee of $1 - \frac{1}{e}$.

5.10 In the *uniform labeling problem*, we are given a graph $G = (V, E)$, costs $c_e \geq 0$ for all $e \in E$, and a set of labels L that can be assigned to the vertices of V. There is a nonnegative cost $c_v^i \geq 0$ for assigning label $i \in L$ to vertex $v \in V$, and an edge $e = (u, v)$ incurs cost c_e if u and v are assigned different labels. The goal of the problem is to assign each vertex in V a label so as to minimize the total cost.

We give an integer programming formulation of the problem. Let the variable $x_v^i \in \{0, 1\}$ be 1 if vertex v is assigned label $i \in L$, and 0 otherwise. Let the variable z_e^i be 1 if exactly one of the two endpoints of the edge e is assigned label i, and 0 otherwise. Then the integer programming formulation is as follows:

$$\begin{aligned}
\text{minimize} \quad & \frac{1}{2} \sum_{e \in E} c_e \sum_{i \in L} z_e^i + \sum_{v \in V, i \in L} c_v^i x_v^i \\
\text{subject to} \quad & \sum_{i \in L} x_v^i = 1, && \forall v \in V, \\
& z_e^i \geq x_u^i - x_v^i, && \forall (u, v) \in E, \forall i \in L, \\
& z_e^i \geq x_v^i - x_u^i, && \forall (u, v) \in E, \forall i \in L, \\
& z_e^i \in \{0, 1\}, && \forall e \in E, \forall i \in L, \\
& x_v^i \in \{0, 1\}, && \forall v \in V, \forall i \in L.
\end{aligned}$$

(a) Prove that the integer programming formulation models the uniform labeling problem.

Consider now the following algorithm. First, the algorithm solves the linear programming relaxation of the integer program above. The algorithm then proceeds in phases. In each phase, it picks a label $i \in L$ uniformly at random, and a number $\alpha \in [0, 1]$ uniformly at random. For each vertex $v \in V$ that has not yet been assigned a label, we assign it label i if $\alpha \leq x_v^i$.

(b) Suppose that vertex $v \in V$ has not yet been assigned a label. Prove that the probability that v is assigned label $i \in L$ in the next phase is exactly $x_v^i / |L|$, and the probability that it is assigned a label in the next phase is exactly $1/|L|$. Further prove that the probability that v is assigned label i by the algorithm is exactly x_v^i.

(c) We say that an edge e is *separated by a phase* if both endpoints were not assigned labels prior to the phase, and exactly one of the endpoints is assigned

a label in this phase. Prove that the probability that an edge e is separated by a phase is $\frac{1}{|L|}\sum_{i\in L} z_e^i$.

(d) Prove that the probability that the endpoints of edge e receive different labels is at most $\sum_{i\in L} z_e^i$.

(e) Prove that the algorithm is a 2-approximation algorithm for the uniform labeling problem.

5.11 Prove Lemma 5.22, and show that the integer programming solution (y^*, C^*) described at the end of Section 5.9 must be optimal for the integer program (5.3).

5.12 Using randomized rounding and First Fit, give a randomized polynomial-time algorithm for the bin-packing problem that uses $\rho \cdot \mathrm{OPT}(I) + k$ bins for some $\rho < 2$ and some small constant k. One idea is to consider the linear program from Section 4.6.

5.13 Show that the $O(\log n)$ factor in Corollary 5.30 can be replaced with $O(\log n / \log\log n)$ by using Theorem 5.23.

5.14 Show that there is a linear programming relaxation for the integer multicommodity flow problem of Section 5.10 that is equivalent to the linear program (5.10) but has a number of variables and constraints that are bounded by a polynomial in the input size of the flow problem.

Chapter Notes

The textbooks of Mitzenmacher and Upfal [226] and Motwani and Raghavan [228] give more extensive treatments of randomized algorithms.

A 1967 paper of Erdős [99] on the maximum cut problem showed that sampling a solution uniformly at random as in Theorem 5.3 gives a solution whose expected value is at least half the sum of the edge weights. This is one of the first randomized approximation algorithms of which we are aware. This algorithm can also be viewed as a randomized version of a deterministic algorithm given by Sahni and Gonzalez [257] (the deterministic algorithm of Sahni and Gonzalez is given in Exercise 5.2).

Raghavan and Thompson [247] were the first to introduce the idea of the randomized rounding of a linear programming relaxation. The result for integer multicommodity flows in Section 5.11 is from their paper.

Random sampling and randomized rounding are most easily applied to unconstrained problems, such as the maximum satisfiability problem and the maximum cut problem, in which any solution is feasible. Even problems such as the prize-collecting Steiner tree problem and the uncapacitated facility location problem can be viewed as unconstrained problems: we need merely to select a set of vertices to span or facilities to open. Randomized approximation algorithms for constrained problems exist, but are much rarer.

The results for the maximum satisfiability problem in this chapter are due to a variety of authors. The simple randomized algorithm of Section 5.1 is given by Yannakakis [293] as a randomized variant of an earlier deterministic algorithm introduced by Johnson [179]. The "biased coins" algorithm of Section 5.3 is a similar randomization of an algorithm of Lieberherr and Specker [216]. The randomized rounding, "better of two," and nonlinear randomized rounding algorithms in Sections 5.4, 5.5, and 5.6, respectively, are due to Goemans and Williamson [137].