Chapter 5

Approximation through Randomization

RANDOMIZATION IS one of the most interesting and useful tools in designing efficient algorithms. Randomized algorithms, indeed, have been proposed for many problems arising in different areas: taking into account the scope of this book, however, we will limit ourselves to considering randomized approximation algorithms for NP-hard optimization problems.

As it can be observed also in the case of the example given in Sect. 2.6, where a randomized algorithm for MAXIMUM CUT was given and analyzed, a remarkable property of randomized algorithms is their structural simplicity. For some problems, in fact, it happens that the only known efficient deterministic algorithms are quite involved, while it is possible to introduce a randomized efficient algorithm which is much easier to code. This happens also in the case of approximation algorithms, where we are interested in achieving good approximate solutions in polynomial time. For example, in this chapter we will describe a simple randomized approximation algorithm for the weighted version of MINIMUM VERTEX COVER, which achieves an expected performance ratio comparable to that of the best deterministic algorithms based on linear programming techniques.

Randomized algorithms can sometimes be even more efficient than deterministic ones in terms of the quality of the returned solution. This will be shown in the case of the weighted versions of MAXIMUM SATISFIABILITY and MAXIMUM CUT: indeed, we will present a randomized 4/3-approximation algorithm for the former problem and a randomized 1.139-approximation algorithm for the latter one.

On the other hand, the main drawback of the randomized approach is that we may only derive statistical properties of the solution returned (in particular, with respect to its expected value): this means that, even if we prove that an algorithm returns solutions of expected good quality, we may, nevertheless, get poor approximate solutions in some cases. However, it is sometimes possible to overcome this drawback by *derandomizing* the algorithm, that is, by transforming the given randomized algorithm into a deterministic one, which always returns in polynomial time a solution whose performance ratio is no more than the expected performance ratio of the solution computed by the randomized algorithm. As we will see at the end of this chapter, this can be done by applying a general technique, called the method of conditional probabilities.

5.1 Randomized algorithms for weighted vertex cover

In this section we present a randomized approximation algorithm for the weighted version of MINIMUM VERTEX COVER. This algorithm achieves an approximate solution whose expected measure is at most twice the optimum measure. Deterministic algorithms for this problem that find approximate solutions whose performance ratio is at most 2 (or even slightly better, i.e., at most $2 - \frac{\log\log n}{2\log n}$) are known (see, for example, Sect. 2.4). In spite of this fact, even though the randomized algorithm does not improve the quality of approximation, it presents a remarkable simplicity, when compared with its deterministic counterparts.

The randomized algorithm (see Program 5.1) exploits the following idea: while there are edges which are not covered, randomly choose a vertex which is an endpoint of an uncovered edge and add this vertex to the vertex cover. The selection of vertices is done by flipping a biased coin that favors the choice of vertices with small weight. Note that, while the choice of an edge is assumed to be done deterministically and how it is performed will not be significant in order to analyze the algorithm behavior, the selection of an endpoint is assumed to be done randomly under a given probability distribution. Note also that if the graph is unweighted (that is, each vertex has weight 1), then an endpoint is chosen with probability 1/2.

Clearly, Program 5.1 runs in polynomial time. To analyze the performance ratio of the solution obtained, let $m_{RWVC}(x)$ be the random variable denoting the value of the solution found by the algorithm on instance x.

Theorem 5.1 ▶ *Given an instance x of the weighted version of* MINIMUM VERTEX COVER, *the expected measure of the solution returned by Program 5.1*

Program 5.1: Random Weighted Vertex Cover

```
input Graph G = (V, E), weight function w : V → ℕ;
output Vertex cover U;
begin
  U := ∅;
  while E ≠ ∅ do
  begin
    Select an edge e = (v, t) ∈ E;
    Randomly choose x from {v, t} with Pr{x = v} = w(t)/(w(v)+w(t));
    U := U ∪ {x};
    E := E − {e | x is an endpoint of e}
  end;
  return U
end.
```

satisfies the following inequality:

$$\mathrm{E}[m_{RWVC}(x)] \leq 2m^*(x).$$

PROOF

Let U be the vertex cover found by the algorithm with input the instance x formed by the graph $G = (V,E)$ and the weight function w, and let U^* be an optimum vertex cover for the same instance. Given any $v \in V$, we define a random variable X_v as follows:

$$X_v = \begin{cases} w(v) & \text{if } v \in U, \\ 0 & \text{otherwise.} \end{cases}$$

Since

$$\mathrm{E}[m_{RWVC}(x)] = \mathrm{E}[\sum_{v \in U} X_v] = \sum_{v \in U} \mathrm{E}[X_v] \leq \sum_{v \in V} \mathrm{E}[X_v]$$

and

$$\sum_{v \in U^*} \mathrm{E}[X_v] = \mathrm{E}[\sum_{v \in U^*} X_v] \leq \mathrm{E}[\sum_{v \in U^*} w(v)] = m^*(x),$$

in order to prove the theorem, it suffices to show that

$$\sum_{v \in V} \mathrm{E}[X_v] \leq 2 \sum_{v \in U^*} \mathrm{E}[X_v]. \tag{5.1}$$

Given an edge (v,t) selected by the algorithm at the first step of the loop, we say that (v,t) *picks* vertex v if v is randomly chosen at the next step. We also denote as $N(v)$ the set of vertices adjacent to v, i.e., $N(v) = \{u \mid u \in V \wedge (v,u) \in E\}$.

Let us now define the random variable $X_{(v,t),v}$ as

$$X_{(v,t),v} = \begin{cases} w(v) & \text{if } (v,t) \text{ picks } v, \\ 0 & \text{otherwise.} \end{cases}$$

Note that if $X_{(v,t),v} = w(v)$, then $X_{(v,t'),v} = 0$ for each $t' \in N(v)$ with $t' \neq t$. This implies that

$$X_v = \sum_{t \in N(v)} X_{(v,t),v}$$

and, by the linearity of expectation, that

$$\mathrm{E}[X_v] = \sum_{t \in N(v)} \mathrm{E}[X_{(v,t),v}].$$

Moreover, $\mathrm{E}[X_{(v,t),v}] = \mathrm{E}[X_{(v,t),t}]$: in fact, we have that

$$\begin{aligned} \mathrm{E}[X_{(v,t),v}] &= w(v)\,\Pr\{(v,t) \text{ picks } v\} \\ &= w(v)\,\Pr\{(v,t) \text{ is chosen}\}\frac{w(t)}{w(v)+w(t)} \\ &= w(t)\,\Pr\{(v,t) \text{ is chosen}\}\frac{w(v)}{w(v)+w(t)} \\ &= w(t)\,\Pr\{(v,t) \text{ picks } t\} \\ &= \mathrm{E}[X_{(v,t),t}]. \end{aligned}$$

Let us now notice that

$$\sum_{v \in U^*} \mathrm{E}[X_v] = \sum_{v \in U^*} \sum_{t \in N(v)} \mathrm{E}[X_{(v,t),v}] \tag{5.2}$$

and that

$$\sum_{v \notin U^*} \mathrm{E}[X_v] = \sum_{v \notin U^*} \sum_{t \in N(v)} \mathrm{E}[X_{(v,t),v}] = \sum_{v \notin U^*} \sum_{t \in N(v)} \mathrm{E}[X_{(v,t),t}]. \tag{5.3}$$

Observe also that, since, for any $v \notin U^*$, each vertex $t \in N(v)$ must be in U^*, then, for each term $\mathrm{E}[X_{(v,t),t}]$ in Eq. (5.3), an equal term $\mathrm{E}[X_{(v,t),v}]$ must appear in Eq. (5.2). This implies that

$$\sum_{v \notin U^*} \mathrm{E}[X_v] \leq \sum_{v \in U^*} \mathrm{E}[X_v]$$

and, as an immediate consequence, that

$$\sum_{v \in V} \mathrm{E}[X_v] = \sum_{v \in U^*} \mathrm{E}[X_v] + \sum_{v \notin U^*} \mathrm{E}[X_v] \leq 2 \sum_{v \in U^*} \mathrm{E}[X_v].$$

QED The theorem thus follows.

5.2 Randomized algorithms for weighted satisfiability

In Sect. 3.1 we presented Program 3.1, a 2-approximate greedy algorithm for MAXIMUM SATISFIABILITY. In this section, we will concentrate on MAXIMUM WEIGHTED SATISFIABILITY, whose input is given by a set of clauses C and a weight function w, which associates to each clause a positive weight, and whose goal is to find a truth assignment that maximizes the sum of the weights of the satisfied clauses.

It can be easily shown that Program 3.1 can be extended to the weighted case preserving the performance ratio (see Exercise 3.2). In this section we present two different randomized algorithms that can be combined in order to achieve an expected performance ratio equal to $4/3$. As we will see in the last section of this chapter, it is possible to derandomize these two algorithms and, hence, to obtain a deterministic algorithm with the same performance ratio for every instance.

The first of the two randomized algorithms is Program 2.10, which can, clearly, be applied also to the case in which the clauses are weighted. It is possible to modify the proof of Theorem 2.19 to show that if k is the minimum number of literals in a clause, then the expected performance ratio of the algorithm is at most $2^k/(2^k-1)$, which, in particular, is equal to 2 when $k=1$ and at most $4/3$ for $k \geq 2$ (see Exercise 5.3).

In the following, we will denote as $m_{RWS}(x)$ the random variable denoting the value of the solution found by Program 2.10 on instance x.

5.2.1 A new randomized approximation algorithm

In Program 2.10 the truth value of every variable is independently and randomly chosen with probability $1/2$. Let us now consider a generalization of that algorithm, which independently assigns the value TRUE to variable x_i, for $i=1,2,\ldots,n$, with probability p_i, where p_i is suitably chosen.

Let $m_{GRWS}(x)$ be the random variable denoting the value of the solution found by this generalization on instance x. It is then easy to see that, for any an instance x, the following equality holds:

$$\mathrm{E}[m_{GRWS}(x)] = \sum_{c \in C} w(c)\left(1 - \prod_{i \in V_c^+}(1-p_i)\prod_{i \in V_c^-} p_i\right)$$

where V_c^+ (respectively, V_c^-) denotes the set of indices of the variables appearing positive (respectively, negative) in clause c.

In the following, we show that it is possible to compute in polynomial time suitable values p_i such that $\mathrm{E}[m_{GRWS}(x)]$ is at most $4/3$ when $k \leq 2$ and is at most $e/(e-1)$ for $k \geq 3$. In order to reach this aim,

Program 5.2: General Random Weighted Satisfiability

input Instance x, i.e., set C of clauses on set of variables V, function $w : C \mapsto \mathbb{N}$;
output Truth assignment $f : V \mapsto \{\text{TRUE}, \text{FALSE}\}$;
begin
  Find the optimum value (y^*, z^*) of *LP-SAT*(x);
  for each variable v_i **do**
  begin
    $p_i := g(y_i^*)$ (for a suitable function g);
    $f(v_i)$:= TRUE with probability p_i
  end;
  return f
end.

we first represent each instance of MAXIMUM WEIGHTED SATISFIABILITY as an integer linear program. Namely, given an instance x of MAXIMUM WEIGHTED SATISFIABILITY formed by a set $C = \{c_1, c_2, \ldots, c_t\}$ of clauses defined over the Boolean variables $v_1, \ldots, v_n$ and a weight function w, we define the following integer liner program *IP-SAT*(x):

$$\begin{aligned} \texttt{maximize} \quad & \sum_{c_j \in C} w(c_j) z_j & \\ \texttt{subject to} \quad & \sum_{i \in V_{c_j}^+} y_i + \sum_{i \in V_{c_j}^-} (1 - y_i) \geq z_j & \forall c_j \in C \\ & y_i \in \{0,1\} & 1 \leq i \leq n \\ & z_j \in \{0,1\} & 1 \leq j \leq t. \end{aligned}$$

Observe that we may define a one-to-one correspondence between feasible solutions of x and feasible solutions of *IP-SAT*(x) as follows:

- $y_i = 1$ if and only if variable x_i is true;
- $z_j = 1$ if and only if clause C_j is satisfied.

Let *LP-SAT*(x) be the linear program obtained by relaxing the integrality constraints of *IP-SAT*(x) and let $(y^* = (y_1^*, \ldots, y_n^*), z^* = (z_1^*, \ldots, z_t^*))$ be an optimal solution of *LP-SAT*(x): clearly, $m^*_{LP-SAT}(x) \geq m^*_{IP-SAT}(x)$.

Given an instance x of MAXIMUM WEIGHTED SATISFIABILITY, Program 5.2 first solves *LP-SAT*(x) obtaining an optimal solution (y^*, z^*). Then, given a function g to be specified later, it computes probabilities $p_i = g(y_i^*)$, for $i = 1, \ldots, n$ and assigns the truth values according to these probabilities. If the function g can be computed in polynomial time, then Program 5.2 clearly runs in polynomial time.

The performance ratio of the returned solution depends on the choice of the function g. Let us suppose that there exists a real number α, with $0 < \alpha < 1$, such that

$$(1 - \prod_{i \in V_{c_j}^+} (1 - p_i) \prod_{i \in V_{c_j}^-} p_i) \geq \alpha z_j^*,$$

for each clause c_j. Since

$$\sum_{j=1}^{t} w(c_j) z_j^* = m_{LP-SAT}^*(x)$$

and

$$\mathrm{E}[m_{GRWS}(x)] = \sum_{c_j \in C} w(c_j)(1 - \prod_{i \in V_{c_j}^+} (1 - p_i) \prod_{i \in V_{c_j}^-} p_i),$$

the solution returned by Program 5.2 has expected performance ratio at most $1/\alpha$.

A first interesting choice of the function g consists in setting $g(y_i) = y_i^*$, for $i = 1, 2, \ldots, n$: in other words, each variable v_i is independently set to TRUE with probability y_i^*.

◀ Lemma 5.2

Given an instance x of MAXIMUM WEIGHTED SATISFIABILITY, *let (y^*, z^*) be an optimal solution of* LP-SAT(x). *Then, for any clause c_j in x with k literals, we have*

$$(1 - \prod_{i \in V_{c_j}^+} (1 - y_i^*) \prod_{i \in V_{c_j}^-} y_i^*) \geq \alpha_k z_j^*$$

where

$$\alpha_k = 1 - \left(1 - \frac{1}{k}\right)^k.$$

PROOF

Without loss of generality, we assume that every variable in clause c_j is positive (i.e., $c_j = v_{j_1} \vee \ldots \vee v_{j_k}$). The lemma is proved by showing that

$$1 - \prod_{i=1}^{k} (1 - y_{j_i}^*) \geq \alpha_k z_j^*.$$

To this aim, recall that, given a set of nonnegative numbers $\{a_1, \ldots, a_k\}$, we have that

$$\frac{a_1 + \cdots + a_k}{k} \geq \sqrt[k]{a_1 \cdots a_k}.$$

By applying the above inequality to the set $\{1-y^*_{j_1},\ldots,1-y^*_{j_k}\}$ and recalling that $\sum_{i=1}^{k} y^*_{j_i} \geq z^*_j$ we obtain

$$\begin{aligned} 1-\prod_{i=1}^{k}(1-y^*_{j_i}) &\geq 1-\left(\frac{\sum_{i=1}^{k}(1-y^*_{j_i})}{k}\right)^k \geq 1-\left(1-\frac{\sum_{i=1}^{k} y^*_{j_i}}{k}\right)^k \\ &\geq 1-\left(1-\frac{z^*_j}{k}\right)^k \geq \alpha_k z^*_j, \end{aligned}$$

where the last inequality is due to the fact that

$$f(z^*_j) = 1-\left(1-\frac{z^*_j}{k}\right)^k$$

is a concave function in the interval of interest, i.e., $[0,1]$, and $f(z^*_j) \geq \alpha_k z^*_j$ at the extremal points of the interval. The lemma is thus proved. QED

Since α_k is a decreasing function with respect to k, given an instance of MAXIMUM WEIGHTED SATISFIABILITY such that each clause has at most k literals, the previous lemma implies that choosing g as the identity function in Program 5.2 yields a randomized algorithm whose expected performance ratio is at most $1/\alpha_k$.

In particular, if $k \leq 2$, then the expected performance ratio is bounded by $4/3$, while if $k \geq 3$, since $\lim_{k\to\infty}(1-(1/k))^k = 1/e$, the expected performance ratio is at most $e/(e-1) \approx 1.582$.

5.2.2 A 4/3-approximation randomized algorithm

In this section we show that an appropriate combination of the two randomized approximation algorithm described above allows to obtain a $4/3$-approximation randomized algorithm.

First note that Program 2.10 has expected performance ratio bounded by $4/3$ if we deal with clauses with *at least* 2 literals. On the other hand, Program 5.2 with g equal to the identity function has the same expected performance ratio if we deal with clauses with *at most* 2 literals.

We can then derive a new algorithm, which simply chooses the best truth assignment returned by the previous two algorithms. The expected performance ratio of this new algorithm is analyzed in the following lemma.

Lemma 5.3 ▶ *Given an instance x of* MAXIMUM WEIGHTED SATISFIABILITY, *let W_1 be the expected measure of the solution returned by Program 2.10 on input*

x and let W_2 be the expected measure of the solution returned by Program 5.2, with g equal to the identity function, on input x. Then the following inequality holds:

$$\max(W_1, W_2) \geq \frac{3}{4} m^*(x).$$

PROOF

Since $\max(W_1, W_2) \geq (W_1 + W_2)/2$ and $m^*_{LP-SAT}(x) \geq m^*(x)$, it is sufficient to show that $(W_1 + W_2)/2 \geq 3m^*_{LP-SAT}(x)/4$. Let us denote by C^k the set of clauses with exactly k literals. From the proof of Theorem 2.19 (see also Exercise 5.3), it follows that each clause $c_j \in C^k$ is satisfied by the truth assignment returned by Program 2.10 with probability $1 - 1/2^k$. Hence,

$$W_1 \geq \sum_{k \geq 1} \sum_{c_j \in C^k} \gamma_k w(c_j) \geq \sum_{k \geq 1} \sum_{c_j \in C^k} \gamma_k w(c_j) z^*_j \tag{5.4}$$

where

$$\gamma_k = \left(1 - \frac{1}{2^k}\right)$$

and the last inequality is due to the fact that $0 \leq z^*_j \leq 1$. Moreover, by Lemma 5.2, we have that

$$W_2 \geq \sum_{k \geq 1} \sum_{c_j \in C^k} \alpha_k w(c_j) z^*_j \tag{5.5}$$

where

$$\alpha_k = 1 - (1 - \frac{1}{k})^k.$$

By summing Eqs. (5.4) and (5.5), we obtain

$$\frac{W_1 + W_2}{2} \geq \sum_{k \geq 1} \sum_{c_j \in C^k} \frac{\gamma_k + \alpha_k}{2} w(c_j) z^*_j.$$

Notice that $\gamma_1 + \alpha_1 = \gamma_2 + \alpha_2 = 3/2$. Moreover, for $k \geq 3$, we have that

$$\gamma_k + \alpha_k \geq 7/8 + 1 - \frac{1}{e} \geq 3/2.$$

Hence, it follows that

$$\frac{W_1 + W_2}{2} \geq \sum_{k \geq 1} \sum_{c_j \in C^k} \frac{3}{4} w(c_j) z^*_j = \frac{3}{4} m^*_{LP-SAT}(x)$$

and the lemma is proved.

QED

Note that it is not necessary to separately apply Programs 2.10 and 5.2 and then choose the best between the two returned solutions. Indeed, it is possible to obtain the same expected performance ratio by randomly choosing one of the two algorithms with probability $1/2$.

The proof of the following theorem easily follows from the previous lemma and is, hence, omitted.

Theorem 5.4 ▶ *There exists a randomized algorithm for* MAXIMUM WEIGHTED SATISFIABILITY *whose expected performance ratio is at most* $4/3$.

5.3 Algorithms based on semidefinite programming

In the last section we have seen that it is possible to design good randomized approximation algorithms for MAXIMUM WEIGHTED SATISFIABILITY by first relaxing the integrality constraint of an integer program and, subsequently, probabilistically rounding the optimal solution of the linear programming relaxation. This technique can be fruitfully applied to a limited number of cases. However, the underlying idea of relaxing and rounding is extremely powerful and it can be applied to other significant problems if a suitable relaxation can be found.

In this section we present a randomized approximation algorithm for the weighted version of MAXIMUM CUT, called MAXIMUM WEIGHTED CUT: given a graph $G = (V, E)$ and a weight function $w : E \mapsto \mathbb{N}$, we want to find a partition (V_1, V_2) of V such that the total weight of the corresponding cut, i.e., the set of edges with an endpoint in V_1 and the other endpoint in V_2, is maximized. We now present a randomized algorithm based on a *semidefinite* relaxation of an integer quadratic formulation of the problem, which returns a solution whose expected performance ratio is at most 1.139.

Let us first express an instance x of MAXIMUM WEIGHTED CUT as an integer quadratic program *IQP-CUT*(x). To this aim, let us associate to each pair $v_i, v_j \in V$ a value w_{ij} defined as $w_{ij} = w(v_i, v_j)$ if $(v_i, v_j) \in E$, $w_{ij} = 0$ otherwise. The integer quadratic program *IQP-CUT*(x) is then defined as

$$\begin{aligned} \texttt{maximize} \quad & \frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{j-1} w_{ij}(1 - y_i y_j) \\ \texttt{subject to} \quad & y_i \in \{-1, 1\} \qquad\qquad 1 \leq i \leq n, \end{aligned}$$

where n denotes the number of vertices of the graph. Observe that an assignment of values to variables y_i naturally corresponds to a partition

Program 5.3: Random Weighted Cut

input Instance x, i.e., graph $G = (V, E)$ and weight function w;
output Partition $\{V_1, V_2\}$ of V;
begin
  Find an optimal solution $(\mathbf{y}_1^*, \ldots, \mathbf{y}_n^*)$ of *QP-CUT*(x);
  Randomly choose a vector $\mathbf{r} \in S_2$ according to the uniform distribution;
  $V_1 := \{v_i \in V \mid \mathbf{y}_i^* \cdot \mathbf{r} \geq 0\}$;
  $V_2 := V - V_1$;
  return $\{V_1, V_2\}$
end.

(V_1, V_2) of V with cut weight equal to $\frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{j-1} w_{ij}(1 - y_i y_j)$. Indeed, let us consider two adjacent vertices v_i and v_j: if either $v_i, v_j \in V_1$ or $v_i, v_j \in V_2$ (that is, $y_i = y_j$), then $1 - y_i y_j = 0$; on the other hand, if v_i and v_j do not belong to the same set (that is, $y_i \neq y_j$), then $\frac{1}{2}(1 - y_i y_j) = 1$.

Notice that each variable y_i can be considered as a vector of unit norm in the 1-dimensional space. Let us now relax *IQP-CUT*(x) by substituting each y_i with a 2-dimensional vector $\mathbf{y}_i$ of unit norm (that is, a vector belonging to the unit 2-dimensional sphere S_2). The relaxation *QP-CUT*(x) is then defined as

$$\begin{aligned} \texttt{maximize} \quad & \frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{j-1} w_{ij}(1 - \mathbf{y}_i \cdot \mathbf{y}_j) \\ \texttt{subject to} \quad & \mathbf{y}_i \in S_2 \qquad\qquad 1 \leq i \leq n, \end{aligned}$$

where $\mathbf{y}_i \cdot \mathbf{y}_j$ denotes the inner product of vectors $\mathbf{y}_i$ and $\mathbf{y}_j$ (that is, $\mathbf{y}_i \cdot \mathbf{y}_j = y_{i,1}y_{j,1} + y_{i,2}y_{j,2}$).

QP-CUT(x) is clearly a relaxation of *IQP-CUT*(x). Indeed, given a feasible solution $Y = (y_1, \ldots, y_n)$ of *IQP-CUT*(x), we can obtain the following feasible solution of *QP-CUT*(x): $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)$ where for all $\mathbf{y}_i$, $\mathbf{y}_i = (y_i, 0)$. Clearly, the measures of Y and $\mathbf{Y}$ coincide.

Let us now consider a randomized approximation algorithm for MAXIMUM WEIGHTED CUT, which, given an instance x, behaves as follows (see Program 5.3): it first finds an optimal solution $(\mathbf{y}_1^*, \ldots, \mathbf{y}_n^*)$ of *QP-CUT*(x), and then computes an approximate solution of MAXIMUM WEIGHTED CUT by randomly choosing a 2-dimensional vector $\mathbf{r} \in S_2$ and putting each vertex v_i in V_1 or in V_2 depending on whether the corresponding vector $\mathbf{y}_i^*$ is above or below the line normal to r. An example of how the algorithm decides in which set a vertex has to be put is shown in Fig. 5.1: in this case, we have that v_2, v_4, v_5, and v_7 are included in V_1 while v_1, v_3, and v_6 are included in V_2.

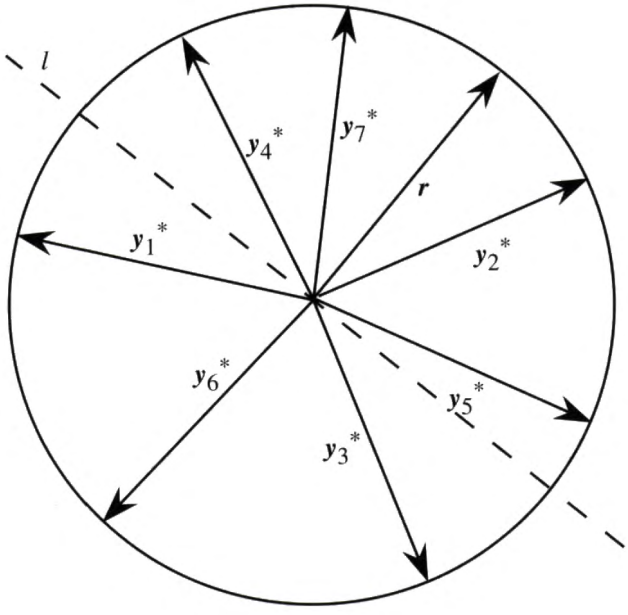

Figure 5.1
Finding a cut by separating vectors on the unit sphere

Let us now show that the expected weight of the cut returned by the algorithm is at least 0.87856 times the optimal measure, that is, the expected performance ratio of the algorithm is at most 1.139.

Lemma 5.5 ▶ *Given an instance x of* MAXIMUM WEIGHTED CUT, *let $m_{RWC}(x)$ be the measure of the solution returned by Program 5.3. Then, the following equality holds:*

$$\mathrm{E}[m_{RWC}(x)] = \frac{1}{\pi} \sum_{j=1}^{n} \sum_{i=1}^{j-1} w_{ij} \arccos(\mathbf{y}_i^* \cdot \mathbf{y}_j^*).$$

PROOF Let us first define the function *sgn* as

$$sgn(x) = \begin{cases} 1 & \text{if } x \geq 0, \\ -1 & \text{otherwise.} \end{cases}$$

Observe that the expected value $\mathrm{E}[m_{RWC}(x)]$ clearly verifies the following equality:

$$\mathrm{E}[m_{RWC}(x)] = \sum_{j=1}^{n} \sum_{i=1}^{j-1} w_{ij} \Pr\{sgn(\mathbf{y}_i^* \cdot \mathbf{r}) \neq sgn(\mathbf{y}_j^* \cdot \mathbf{r})\}$$

where $\mathbf{r}$ is a randomly and uniformly chosen vector in S_2. Therefore, to prove the lemma it is sufficient to show that

$$\Pr\{sgn(\mathbf{y}_i^* \cdot \mathbf{r}) \neq sgn(\mathbf{y}_j^* \cdot \mathbf{r})\} = \frac{\arccos(\mathbf{y}_i^* \cdot \mathbf{y}_j^*)}{\pi}. \tag{5.6}$$

Note that $sgn(\mathbf{y}_i^* \cdot \mathbf{r}) \neq sgn(\mathbf{y}_j^* \cdot \mathbf{r})$ if and only if the random line l normal to $\mathbf{r}$ separates $\mathbf{y}_i^*$ and $\mathbf{y}_j^*$. The random choice of $\mathbf{r}$ implies that l has two opposite intersecting points s and t with S_2 that are uniformly distributed. Moreover, $\mathbf{y}_i^*$ and $\mathbf{y}_j^*$ are separated by l if and only if either s or t lies on the shorter arc of S_2 between $\mathbf{y}_i^*$ and $\mathbf{y}_j^*$ (see Fig. 5.2). The probability that either s or t lies on this arc is

$$\frac{\arccos(\mathbf{y}_i^* \cdot \mathbf{y}_j^*)}{2\pi} + \frac{\arccos(\mathbf{y}_i^* \cdot \mathbf{y}_j^*)}{2\pi} = \frac{\arccos(\mathbf{y}_i^* \cdot \mathbf{y}_j^*)}{\pi}.$$

Hence, Eq. (5.6) follows and the lemma is proved. QED

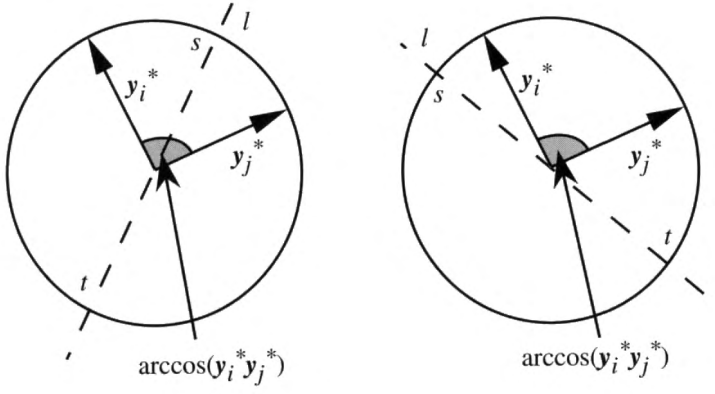

Figure 5.2
The probability of separating two vectors

◀ Theorem 5.6

For any instance of MAXIMUM WEIGHTED CUT, *Program 5.3 returns a solution whose expected measure is at least* 0.8785 *times the optimum measure.*

PROOF

Let us define

$$\beta = \min_{0<\alpha\leq\pi} \frac{2\alpha}{\pi(1-\cos\alpha)}.$$

Given an instance x of MAXIMUM WEIGHTED CUT with optimal measure $m^*(x)$, let $\mathbf{y}_1^*, \ldots, \mathbf{y}_n^*$ be an optimal solution of *QP-CUT*(x) with measure

$$m^*_{QP-CUT}(x) = \frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{j-1} w_{ij}(1-\mathbf{y}_i^* \cdot \mathbf{y}_j^*).$$

If we consider the change of variables $\mathbf{y}_i^* \cdot \mathbf{y}_j^* = \cos\alpha_{ij}$, we have, by definition of β,

$$\beta \leq \frac{2\alpha_{ij}}{\pi(1-\cos\alpha_{ij})} = \frac{2}{\pi}\frac{\arccos(\mathbf{y}_i^* \cdot \mathbf{y}_j^*)}{1-(\mathbf{y}_i^* \cdot \mathbf{y}_j^*)}$$

or, equivalently,

$$\frac{\arccos(\mathbf{y}_i^* \cdot \mathbf{y}_j^*)}{\pi} \geq \frac{\beta}{2}(1 - (\mathbf{y}_i^* \cdot \mathbf{y}_j^*)).$$

Since *QP-CUT*(x) is a relaxation of *IQP-CUT*(x), we have that

$$\begin{aligned} \mathrm{E}[m_{RWC}(x)] &\geq \frac{1}{2}\beta \sum_{j=1}^{n} \sum_{i=1}^{j-1} w_{ij}(1 - \mathbf{y}_i^* \cdot \mathbf{y}_j^*) \\ &= \beta m_{QP-CUT}^*(x) \geq m_{IQP-CUT}^*(x) = \beta m^*(x), \end{aligned}$$

where $m_{RWC}(x)$ is the measure of the solution returned by Program 5.3.

Since it is possible to show that $\beta > 0.8785$ (see Exercise 5.10), the QED Lemma is thus proved.

Regarding the time complexity of Program 5.3, it is clear that the algorithm runs in polynomial time if and only if it is possible to solve *QP-CUT*(x) in polynomial time. Unfortunately, it is not known whether this is possible. However, the definition of *QP-CUT*(x) can be slightly modified in order to make it efficiently solvable: the modification simply consists in considering variables $\mathbf{y}_i$ as vectors in the n-dimensional space instead that in the 2-dimensional one. In particular, the n-dimensional version of *QP-CUT*(x) is defined as

$$\begin{aligned} \texttt{maximize} \quad & \frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{j-1} w_{ij}(1 - \mathbf{y}_i \cdot \mathbf{y}_j) \\ \texttt{subject to} \quad & \mathbf{y}_i \in S_n \qquad\qquad 1 \leq i \leq n, \end{aligned}$$

where S_n denotes the n-dimensional unit sphere. Observe that, clearly, the above analysis of the expected performance ratio of Program 5.3 can still be carried out if we refer to this new version of *QP-CUT*(x).

In order to justify this modification, we need some definitions and results from linear algebra. First of all, we say that a $n \times n$ matrix M is *positive semidefinite* if, for every vector $x \in R^n$, $x^T M x \geq 0$. It is known that a $n \times n$ symmetric matrix M is positive semidefinite if and only if there exists a matrix P such that $M = P^T P$, where P is an $m \times n$ matrix for some $m \leq n$. Moreover, if M is positive semidefinite, then matrix P can be computed in polynomial time (see Bibliographical notes).

Observe now that, given n vectors $\mathbf{y}_1, \ldots, \mathbf{y}_n \in S_n$, the matrix M defined as $M_{i,j} = \mathbf{y}_i \cdot \mathbf{y}_j$ is positive semidefinite. On the other hand, from the above properties of positive semidefinite matrices, it follows that, given a $n \times n$ positive semidefinite matrix M such that $M_{i,i} = 1$ for $i = 1, \ldots, n$, it is

possible to compute, in polynomial time, a set of n vectors $\mathbf{y}_1, \ldots, \mathbf{y}_n \in S_n$ such that $M_{i,j} = \mathbf{y}_i \cdot \mathbf{y}_j$.

In other words, *QP-CUT*(x) is equivalent to the following *semidefinite* program *SD-CUT*(x):

$$\begin{array}{ll} \texttt{maximize} & \frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{j-1} w_{ij}(1 - M_{i,j}) \\ \texttt{subject to} & M \text{ is positive semidefinite} \\ & M_{i,i} = 1 \qquad\qquad 1 \leq i \leq n. \end{array}$$

It can be proven that, for any instance x of MAXIMUM WEIGHTED CUT, if $m^*_{SD-CUT}(x)$ is the optimal value of *SD-CUT*(x), then, for any $\varepsilon > 0$, it is possible to find a solution with measure greater than $m^*_{SD-CUT}(x) - \varepsilon$ in time polynomial both in $|x|$ and in $\log(1/\varepsilon)$ (see Bibliographical notes). It is also possible to verify that solving *SD-CUT*(x) with $\varepsilon = 10^{-5}$ does not affect the previously obtained performance ratio of 0.8785. Therefore, the following theorem holds.

Program 5.3, where the optimal solution of QP-CUT(x) *is obtained by solving the equivalent program* SD-CUT(x), *runs in polynomial time.* ◀ Theorem 5.7

As a consequence of Theorems 5.6 and 5.7, it thus follows that MAXIMUM WEIGHTED CUT admits a polynomial-time randomized algorithm whose expected performance ratio is at most 1.139.

5.3.1 Improved algorithms for weighted 2-satisfiability

The approach based on semidefinite programming can be applied to other problems and, in particular, to satisfiability problems. Let us consider, for example, MAXIMUM WEIGHTED 2-SATISFIABILITY, that is, the weighted satisfiability problem in which every clause has at most two literals.

Given an instance x of MAXIMUM WEIGHTED 2-SATISFIABILITY with n variables $v_1, \ldots, v_n$, let us define the following integer quadratic program *IQP-SAT*(x):

$$\begin{array}{ll} \texttt{maximize} & \sum_{j=0}^{n}\sum_{i=0}^{j-1}[a_{ij}(1 - y_i y_j) + b_{ij}(1 + y_i y_j)] \\ \texttt{subject to} & y_i \in \{-1, 1\} \qquad\qquad i = 0, 1, \ldots, n, \end{array}$$

where a_{ij} and b_{ij} are non-negative coefficients that will be specified later, y_i is a variable associated with v_i, for $i = 1, \ldots, n$, and y_0 denotes the boolean value TRUE, that is, v_i is TRUE if and only if $y_i = y_0$, for $i = 1, \ldots, n$.

In order to define the values of the coefficients a_{ij} and b_{ij}, let us define the value $t(c_j)$ of a clause c_j as follows:

$$t(c_j) = \begin{cases} 1 & \text{if } c_j \text{ is satisfied,} \\ 0 & \text{if } c_j \text{ is not satisfied.} \end{cases}$$

According to the previous definitions, it results that if c_j is a unit clause, then

$$t(c_j) = \frac{1 + y_i y_0}{2}$$

if $c_j = v_i$, and

$$t(c_j) = \frac{1 - y_i y_0}{2}$$

otherwise. If c_j contains two literals, then its value can be inductively computed: for example, if $c_j = v_i \vee v_k$, then

$$\begin{aligned} t(c_j) &= 1 - t(\overline{v}_i \wedge \overline{v}_k) = 1 - t(\overline{v}_i)t(\overline{v}_k) = 1 - \frac{1 - y_i y_0}{2}\frac{1 - y_k y_0}{2} \\ &= \frac{1}{4}(3 + y_i y_0 + y_k y_0 - y_i y_k y_0^2) \\ &= \frac{1}{4}[(1 + y_i y_0) + (1 + y_k y_0) + (1 - y_i y_k)] \end{aligned}$$

(the cost of the other possible clauses with two literals can be computed in a similar way).

Hence, it is possible, for any instance x of MAXIMUM WEIGHTED 2-SATISFIABILITY, to compute suitable values of a_{ij} and b_{ij} such that the resulting program *IQP-SAT*(x) is an equivalent formulation of instance x.

Program *IQP-SAT*(x) can be relaxed using the same approach used for MAXIMUM WEIGHTED CUT. By introducing unit norm $(n+1)$-dimensional vectors $\mathbf{y}_i$, for $i = 0, 1, \ldots, n$, we can indeed obtain the semidefinite relaxation of *IQP-SAT*(x) and, then, prove the following result (see Exercise 5.11).

Theorem 5.8 ▶ *There exists a randomized polynomial-time algorithm for* MAXIMUM WEIGHTED 2-SATISFIABILITY *whose expected performance ratio is at most* 1.139.

5.4 The method of the conditional probabilities

In this section we will see that a randomized approximation algorithm $\mathcal{A}$ can sometimes be *derandomized*, that is, a deterministic algorithm can be

derived whose running time is comparable to $\mathcal{A}$'s running time and whose performance ratio is no more than the expected performance ratio of $\mathcal{A}$. In particular, we will briefly describe a general technique known as the *method of conditional probabilities* and we will show how it can be applied to derandomize Program 2.10, when applied to MAXIMUM WEIGHTED SATISFIABILITY.

The method of conditional probabilities is based on viewing the behavior of a randomized approximation algorithm on a given input as a computation tree. To this aim, we assume, without loss of generality, that $\mathcal{A}$, on input x, independently performs $r(|x|)$ random choices each with exactly two possible outcomes, denoted by 0 and 1. According to this hypothesis, we can then define, for any input x, a complete binary tree of height $r(|x|)$ in which each node of level i is associated with the i-th random choice of $\mathcal{A}$ with input x, for $i = 1, \ldots, r(|x|)$: the left subtree of the node corresponds to outcome 0, while the right subtree corresponds to outcome 1. In this way, each path from the root to a leaf of this tree corresponds to a possible computation of $\mathcal{A}$ with input x.

Notice that, to each node u of level i, it is possible to associate a binary string $\sigma(u)$ of length $i-1$ representing the random choices performed so far. Moreover, we can associate to each leaf l a value m_l, which is the measure of the solution returned by the corresponding computation, and to each inner node u the average measure $\mathrm{E}(u)$ of the values of all leaves in the subtree rooted at u. Clearly, $\mathrm{E}(u)$ is the expected measure of the solution returned by $\mathcal{A}$ with input x, assumed that the outcomes of the first $|\sigma(u)|$ random choices are consistent with $\sigma(u)$. It is easy to show that, for any inner node u, if v and w are the two children of u, then either $\mathrm{E}(v) \geq \mathrm{E}(u)$ or $\mathrm{E}(w) \geq \mathrm{E}(u)$.

The derandomization is then based on the following observation: if r is the root of the computation tree, then there must exists a path from r to a leaf l such that $m_l \geq \mathrm{E}(r)$, that is, the measure of the solution returned by the corresponding computation is at least equal to the expected measure of the solution returned by $\mathcal{A}$ with input x. This path can be deterministically derived if, in order to choose which of the children v and w to proceed from a node u, we are able to efficiently determine which of $\mathrm{E}(v)$ and $\mathrm{E}(w)$ is greater.

In the following we will show how this approach can be applied to derandomize Program 2.10 in order to obtain a deterministic 2-approximation algorithm for the weighted version of MAXIMUM SATISFIABILITY.

Given an instance x of MAXIMUM WEIGHTED SATISFIABILITY, let $v_1, \ldots, v_n$ be the Boolean variables in x, which can be considered as $\{0,1\}$-variables, where the boolean values TRUE and FALSE are represented by 1 and 0, respectively. The deterministic algorithm consists of n iterations

corresponding to the n random choices performed by Program 2.10. At the i-th iteration, the value of variable v_i is determined as follows: let $\overline{v}_1, \ldots, \overline{v}_{i-1}$ be the values of variables $v_1, \ldots, v_{i-1}$ determined so far, and let

$$m_{RWS}(x \mid \overline{v}_1, \ldots, \overline{v}_{i-1})$$

be the random variable denoting the measure of the solution found by Program 2.10 when applied to the instance obtained from x by assuming that the value of variables $v_1, \ldots, v_{i-1}$ is $\overline{v}_1, \ldots, \overline{v}_{i-1}$ and applying Program 2.10 to determine the values of variables $v_i, \ldots, v_n$.

Given $\overline{v}_1, \ldots, \overline{v}_{i-1}$, the value of v_i is determined by computing

$$\mathrm{E}[m_{RWS}(x \mid \overline{v}_1, \ldots, \overline{v}_{i-1}, 0)]$$

and

$$\mathrm{E}[m_{RWS}(x \mid \overline{v}_1, \ldots, \overline{v}_{i-1}, 1)].$$

If $\mathrm{E}[m_{RWS}(x \mid \overline{v}_1, \ldots, v_{i-1}, 0)] \leq \mathrm{E}[m_{RWS}(x \mid \overline{v}_1, \ldots, \overline{v}_{i-1}, 1)]$ then the value of v_i is set to 1, otherwise it is set to 0. After n iterations, a truth assignment $\overline{v}_1, \ldots, v_n$ has been obtained with value

$$m_{\mathcal{A}}(x) = \mathrm{E}[m_{RWS}(x \mid \overline{v}_1, \ldots, v_n)].$$

We first show that the computation of $\mathrm{E}[m_{RWS}(x \mid \overline{v}_1, \ldots, \overline{v}_{i-1}, 0)]$ and $\mathrm{E}[m_{RWS}(x \mid \overline{v}_1, \ldots, \overline{v}_{i-1}, 1)]$ can be performed in deterministic polynomial time and, then, that $m_{\mathcal{A}}(x)$ is at least one half of the optimal measure. We will show how to compute $\mathrm{E}[m_{RWS}(x \mid \overline{v}_1, \ldots, \overline{v}_{i-1}, 1)]$ in polynomial time: the computation of $\mathrm{E}[m_{RWS}(x \mid \overline{v}_1, \ldots, \overline{v}_{i-1}, 0)]$ is analogous and it is omitted.

Assume that x contains t clauses $c_1, \ldots, c_t$. We have

$$\mathrm{E}[m_{RWS}(x \mid \overline{v}_1, \ldots, \overline{v}_{i-1}, 1)] = \sum_{j=1}^{t} w(c_j) \Pr\{c_j \text{ is satisfied} \mid \overline{v}_1, \ldots, v_{i-1}, 1\}$$

where

$$\Pr\{c_j \text{ is satisfied} \mid \overline{v}_1, \ldots, \overline{v}_{i-1}, 1\}$$

denotes the probability that a random truth assignment of variables $v_{i+1}, \ldots, v_n$ satisfies clause c_j given that $\overline{v}_1, \ldots, \overline{v}_{i-1}, 1$ are the truth assignments of variables $v_1, \ldots, v_{i-1}, v_i$, respectively.

Let W_i be the sum of the weights of the clauses that are satisfied by values $\overline{v}_1, \ldots, \overline{v}_{i-1}$ of variables $v_1, \ldots, v_{i-1}$ and let $C^-(i)$ be the set of clauses that are not satisfied by $v_1, \ldots, v_{i-1}$ and could be satisfied by a suitable assignment of values to variables $v_i, \ldots, v_n$.

Let c_j be a clause in $C^-(i)$. If v_i occurs positive in c_j then

$$\Pr\{c_j \text{ is satisfied} \mid v_1, \ldots, v_{i-1}, 1\} = 1.$$

If x_i occurs negative or does not occur in c_j, let d_j be the number of variables occurring in c_j that are different from $v_1, \ldots, v_i$. The probability that a random assignment of values to variables $v_{i+1}, \ldots, v_n$ satisfies clause c_j is

$$\Pr\{c_j \text{ is satisfied} \mid v_1, \ldots, v_{j-1}, 1\} = 1 - \frac{1}{2^{d_j}}.$$

Summing over all the clauses we have that

$$\mathrm{E}[m_{RWS}(x \mid \bar{v}_1, \ldots, \bar{v}_{i-1}, 1)] = W_i + \sum_{\substack{c_j \in C^-(i) \text{ s.t.} v_i \\ \text{occurs positive}}} 1 + \sum_{\substack{c_j \in C^-(i) \text{ s.t.} v_i \\ \text{occurs negative}}} (1 - \frac{1}{2^{d_j}}).$$

It is clear that the above computation can be performed in polynomial time.

In order to analyze the quality of the obtained solution observe that the chosen value $\bar{v}_i$, for $i = 1, \ldots, n$, satisfies

$$\mathrm{E}[m_{RWS}(x \mid \bar{v}_1, \ldots, \bar{v}_i)] \geq \mathrm{E}[m_{RWS}(x \mid \bar{v}_1, \ldots, \bar{v}_{i-1})].$$

Hence we have

$$\begin{aligned} \mathrm{E}[m_{RWS}(x)] &\leq \mathrm{E}[m_{RWS}(x \mid \bar{v}_1)] \\ &\leq \mathrm{E}[m_{RWS}(x \mid \bar{v}_1, \bar{v}_2)] \\ &\leq \ldots \\ &\leq \mathrm{E}[m_{RWS}(x \mid \bar{v}_1, \ldots, \bar{v}_n)] = m_{\mathcal{A}}(x). \end{aligned}$$

Since we have seen in Sect. 5.2 that $\mathrm{E}[m_{RWS}(x)] \geq m^*(x)/2$, it derives that $m_{\mathcal{A}}(C)$ is at least one half of the optimal measure.

The method of conditional probabilities can be successfully applied to derandomize the 4/3-approximation randomized algorithm for MAXIMUM WEIGHTED SATISFIABILITY presented in this chapter. It can also be used to derandomize the semidefinite programming based algorithm for the weighted version of MAXIMUM CUT, even though this derandomization requires a more sophisticated version of the method.

5.5 Exercises

Exercise 5.1 Consider a greedy algorithm for the weighted version of MINIMUM VERTEX COVER that at each step chooses the vertex with minimum weight among vertices that are an endpoint of an uncovered edge.

Show that the algorithm has an unbounded ratio in the worst case. (Hint: consider a star graph, i.e., a graph in which there exists a vertex v_1 that is connected to all the other $n-1$ vertices $v_2 \ldots v_n$ and in which no other edge exists.)

Exercise 5.2 Consider a greedy algorithm for the weighted version of MINIMUM VERTEX COVER that at each step chooses the vertex that has the least ratio weight/degree among vertices that are an endpoint of an uncovered edge. Show that the algorithm has an unbounded ratio in the worst case. (Hint: Consider an unweighted bipartite graph $G = (V \cup R, E)$, where V has n vertices and R is divided into n subsets $R_1, \ldots, R_n$. Every vertex in R_i is connected to i vertices in V and no two vertices in R_i have a common endpoint in V.)

Exercise 5.3 Modify the proof of Theorem 2.19 to show that if k is the minimum number of literals in a clause, then the expected performance ratio of the algorithm applied to weighted clauses is 2 when $k = 1$ and is at most $4/3$ for $k \geq 2$.

Exercise 5.4 A function $g : [0,1] \mapsto [0,1]$ verifies the *3/4-property* if it satisfies the following inequality

$$1 - \prod_{i=1}^{l}(1 - g(y_i)) \prod_{i=l+1}^{k} g(y_i) \geq \frac{3}{4} \min(1, \sum_{i=1}^{l} y_i + \sum_{i=l+1}^{k} (1 - y_i))$$

for any pair of integers k and l with $k \geq l$, and for any $y_1, \ldots, y_k \in [0,1]$. Prove that if a function g with the 3/4-property is used in Program 5.2, then the expected performance ratio of the algorithm is at most 4/3.

Exercise 5.5 Show that if a function $g : [0,1] \mapsto [0,1]$ satisfies the following conditions:

1. $g(y) \leq 1 - g(1-y)$,
2. $1 - \prod_{i=1}^{k}(1 - g(y_i)) \geq \frac{3}{4} \min(1, \sum_{i=1}^{k} y_i)$,

for any integer k, for any $y \in [0,1]$, and for any tuple $y_1, \ldots, y_n \in [0,1]$, then g verifies the 3/4-property.

Exercise 5.6 Show that the following function verifies the 3/4-property:

$$g_\alpha(y) = \alpha + (1 - 2\alpha)y,$$

where

$$2 - \frac{3}{\sqrt[3]{4}} \leq \alpha \leq \frac{1}{4}.$$

Problem 5.1: Maximum Subgraph

INSTANCE: Directed graph $G = (V,A)$.

SOLUTION: An acyclic spanning subgraph $G' = (V,A')$ of G.

MEASURE: $|A'|$.

Exercise 5.7 Show that the following function verifies the 3/4-property:

$$f(y) = \begin{cases} \frac{3}{4}y + \frac{1}{4} & \text{if } 0 \leq y < \frac{1}{3}, \\ \frac{1}{2} & \text{if } \frac{1}{3} \leq y < \frac{2}{3}, \\ \frac{3}{4}y & \text{if } \frac{2}{3} \leq y \leq 1. \end{cases}$$

Exercise 5.8 (*) Apply randomized rounding to MINIMUM SET COVER. Namely, consider the integer programming relaxation I of MINIMUM SET COVER and set each variable to be 1 with probability given by the value of the optimal solution of the linear programming relaxation of I. Show that the probability that a set S_i is covered is at least $1 - (1/e)$.

Exercise 5.9 Apply the result of Exercise 5.8 to show that there exists a randomized algorithm that finds an $O(\log m)$-approximate solution with probability at least 0.5, where m is the number of sets to be covered.

Exercise 5.10 Show that

$$\min_{0<\alpha\leq\pi} \frac{2\alpha}{\pi(1-\cos\alpha)} > 0.87856.$$

Exercise 5.11 Prove Theorem 5.8.

Exercise 5.12 Consider Problem 5.1 and consider the randomized algorithm that chooses a random ordering of the vertices and picks either the arcs that go forward or the arcs that go backward. Prove that this algorithm has expected performance ratio at most 2.

5.6 Bibliographical notes

The first algorithms in which randomization is used explicitly were introduced in the mid-seventies. A classical paper [Rabin, 1976] on primality test is considered to have started the field of randomized algorithms. In the

same period, [Solovay and Strassen, 1977] introduced another randomized algorithm for the same problem. Since then, a lot of problems that arise in many different areas have been studied from this point of view. Here we will limit ourselves to considering approximation algorithms for combinatorial optimization problems. A wide study of randomized algorithms and a rich bibliography can be found in [Motwani and Raghavan, 1995]. A description of the technique of the conditional probabilities can be found in [Alon and Spencer, 1992].

The randomized approximation algorithm for the weighted version of MINIMUM VERTEX COVER is presented in [Pitt, 1985].

The randomized 2-approximation algorithm for MAXIMUM WEIGHTED SATISFIABILITY follows the greedy approach used in [Johnson, 1974a], while the two 4/3-approximation algorithms were presented in [Goemans and Williamson, 1994]. Another 4/3-approximation deterministic algorithm for MAXIMUM WEIGHTED SATISFIABILITY was given in [Yannakakis, 1994]. The approach here followed is rather different and exploits techniques from the theory of maximum flows. Further improvements to the approximation of MAXIMUM WEIGHTED SATISFIABILITY based on semidefinite programming achieve a performance ratio of 1.318 [Goemans and Williamson, 1995b]. In the specific case of MAXIMUM WEIGHTED 2-SATISFIABILITY, [Feige and Goemans, 1995] have achieved a stronger result obtaining a 1.066 bound. Instead, just considering satisfiable formulas, [Karloff and Zwick, 1997] have shown that it is possible to approximate MAXIMUM WEIGHTED 3-SATISFIABILITY with approximation ratio 8/7. Further improvements are based on combining together almost all the known techniques used to approximate MAXIMUM WEIGHTED SATISFIABILITY, obtaining an approximation ratio of 1.29 (see [Ono, Hirata and Asano, 1996] and [Asano, 1997]).

The technique of randomized rounding was introduced in [Raghavan and Thompson, 1987] and [Raghavan, 1988] while studying a wire routing problem. Randomized rounding algorithms that improve the bound given in Exercise 5.9 for the MINIMUM SET COVER problem have been proposed in [Bertsimas and Vohra, 1994, Srinivasan, 1995, Srinivasan, 1996]. In particular, in [Bertsimas and Vohra, 1994] the technique is applied to a variety of covering problems.

The randomized approximation algorithm for the MAXIMUM CUT problem based on semidefinite programming is presented in [Goemans and Williamson, 1995b] while [Mahajan and Ramesh, 1995] give a derandomized version of the algorithm. A proof that semidefinite programming is solvable efficiently can be found in [Alizadeh, 1995]. [Karger, Motwani, and Sudan, 1998] applied semidefinite programming to MINIMUM GRAPH COLORING.