CHAPTER TEN
The Monte Carlo Method

The Monte Carlo method refers to a collection of tools for estimating values through sampling and simulation. Monte Carlo techniques are used extensively in almost all areas of physical sciences and engineering. In this chapter, we first present the basic idea of estimating a value through sampling, using a simple experiment that gives an estimate of the value of the constant $\pi$. Estimating through sampling is often more complex than this simple example suggests. We demonstrate the potential difficulties that can arise in devising an efficient sampling procedure by considering how to appropriately sample in order to estimate the number of satisfying assignments of a disjunctive normal form (DNF) Boolean formula.

We then move to more general considerations, demonstrating a general reduction from almost uniform sampling to approximate counting of combinatorial objects. This leads us to consider how to obtain almost uniform samples. One method is the Markov chain Monte Carlo (MCMC) technique, introduced in the last section of this chapter.

10.1. The Monte Carlo Method

Consider the following approach for estimating the value of the constant $\pi$. Let $(X, Y)$ be a point chosen uniformly at random in a $2 \times 2$ square centered at the origin $(0, 0)$. This is equivalent to choosing $X$ and $Y$ independently from a uniform distribution on $[-1, 1]$. The circle of radius 1 centered at $(0, 0)$ lies inside this square and has area $\pi$. If we let

$$Z = \begin{cases} 1 & \text{if } \sqrt{X^2 + Y^2} \leq 1, \\ 0 & \text{otherwise,} \end{cases}$$

then – because the point was chosen uniformly from the $2 \times 2$ square – the probability that $Z = 1$ is exactly the ratio of the area of the circle to the area of the square. See Figure 10.1. Hence

$$\Pr(Z = 1) = \frac{\pi}{4}.$$
Assume that we run this experiment \( m \) times (with \( X \) and \( Y \) chosen independently among the runs), with \( Z_i \) being the value of \( Z \) at the \( i \)th run. If \( W = \sum_{i=1}^{m} Z_i \), then

\[
E[W] = E\left[ \sum_{i=1}^{m} Z_i \right] = \sum_{i=1}^{m} E[Z_i] = \frac{m \pi}{4},
\]

and hence \( W' = (4/m)W \) is a natural estimate for \( \pi \). Applying the Chernoff bound of Eqn. (4.6), we compute

\[
\Pr(|W' - \pi| \geq \varepsilon \pi) = \Pr\left( \left| W - \frac{m \pi}{4} \right| \geq \frac{\varepsilon m \pi}{4} \right)
\]

\[
= \Pr(|W - E[W]| \geq \varepsilon E[W])
\]

\[
\leq 2e^{-m \varepsilon^2 / 12}.
\]

Therefore, by using a sufficiently large number of samples we can obtain, with high probability, as tight an approximation of \( \pi \) as we wish.

This method for approximating \( \pi \) is an example of a more general class of approximation algorithms that we now characterize.

**Definition 10.1:** A randomized algorithm gives an \((\varepsilon, \delta)\)-approximation for the value \( V \) if the output \( X \) of the algorithm satisfies

\[
\Pr(|X - V| \leq \varepsilon V) \geq 1 - \delta.
\]

Our method for estimating \( \pi \) gives an \((\varepsilon, \delta)\)-approximation, as long as \( \varepsilon < 1 \) and we choose \( m \) large enough to make

\[
2e^{-m \varepsilon^2 / 12} \leq \delta.
\]

Algebraic manipulation yields that choosing

\[
m \geq \frac{12 \ln(2/\delta)}{\pi \varepsilon^2}
\]

is sufficient.

We may generalize the idea behind our technique for estimating \( \pi \) to provide a relation between the number of samples and the quality of the approximation. We use the following simple application of the Chernoff bound throughout this chapter.
THE MONTE CARLO METHOD

Theorem 10.1: Let $X_1, \ldots, X_m$ be independent and identically distributed indicator random variables, with $\mu = \mathbb{E}[X_i]$. If $m \geq (3 \ln(2/\delta))/\varepsilon^2 \mu$, then

$$\Pr\left(\left| \frac{1}{m} \sum_{i=1}^{m} X_i - \mu \right| \geq \varepsilon \mu \right) \leq \delta.$$ 

That is, $m$ samples provide an $(\varepsilon, \delta)$-approximation for $\mu$.

The proof is left as Exercise 10.1.

More generally, we will want an algorithm that approximates not just a single value but instead takes as input a problem instance and approximates the solution value for that problem. Here we are considering problems that map inputs $x$ to values $V(x)$. For example, given an input graph, we might want to know an approximation to the number of independent sets in the graph.

You might ask why we should settle for an approximation: perhaps we should aim for an exact answer. In the case of $\pi$, we cannot obtain an exact answer because $\pi$ is an irrational number. Another reason for seeking an approximation is that, as we shall see shortly, there are problems for which the existence of an algorithm that gives an exact answer would imply that $P = NP$, and hence it is unlikely that such an algorithm will be found. This, however, does not preclude the possibility of an efficient approximation algorithm.

Definition 10.2: A fully polynomial randomized approximation scheme (FPRAS) for a problem is a randomized algorithm for which, given an input $x$ and any parameters $\varepsilon$ and $\delta$ with $0 < \varepsilon, \delta < 1$, the algorithm outputs an $(\varepsilon, \delta)$-approximation to $V(x)$ in time that is polynomial in $1/\varepsilon, \ln \delta^{-1}$, and the size of the input $x$.

Exercise 10.3 considers a seemingly weaker but actually equivalent definition of an FPRAS that avoids the parameter $\delta$.

The Monte Carlo method essentially consists of the approach we have outlined here to obtain an efficient approximation for a value $V$. We require an efficient process that generates a sequence of independent and identically distributed random samples $X_1, X_2, \ldots, X_n$ such that $\mathbb{E}[X_j] = V$. We then take enough samples to get an $(\varepsilon, \delta)$-approximation to $V$. Generating a good sequence of samples is often a nontrivial task and is a major focus of the Monte Carlo method.

The Monte Carlo method is also sometimes called Monte Carlo simulation. As an example, suppose we want to estimate the expected price of a stock sometime in the future. We may develop a model where the price $p(Y_1, \ldots, Y_k)$ of the stock at that time depends on random variables $Y_1, Y_2, \ldots, Y_k$. If we can repeatedly generate independent random vectors $(y_1, y_2, \ldots, y_k)$ from the joint distribution of the $Y_i$, then we can repeatedly generate independent random variables $X_1, X_2, \ldots$, where

$$X_i = p(Y_1, \ldots, Y_k).$$

We can then use the $X_i$ to estimate the expected future price $\mathbb{E}[p(Y_1, \ldots, Y_k)]$ with the Monte Carlo method. That is, by simulating the possible future outcomes of the $Y_i$ many times, we can estimate the desired expectation.
10.2. Application: The DNF Counting Problem

As an example of an estimation problem that requires a nontrivial sampling technique, we consider the problem of counting the number of satisfying assignments of a Boolean formula in disjunctive normal form (DNF). A DNF formula is a disjunction (OR) of clauses \( C_1 \lor C_2 \lor \cdots \lor C_j \), where each clause is a conjunction (AND) of literals. For example, the following is a DNF formula:

\[
(x_1 \land \overline{x}_2 \land x_3) \lor (x_2 \land x_4) \lor (\overline{x}_1 \land x_2 \land x_4).
\]

Recall from Section 6.2.2 that, in a standard satisfiability problem, the input formula is a conjunction (AND) of a set of clauses, and each clause is the disjunction (OR) of literals. This is commonly called conjunctive normal form (CNF). While determining the satisfiability of a formula in CNF form is difficult, determining the satisfiability of a formula in DNF form is simple. Since a satisfying assignment for a DNF formula needs to satisfy only one clause, it is easy to find a satisfying assignment or prove that it is not satisfiable.

How hard is it to exactly count the number of satisfying assignments of a DNF formula? Given any CNF formula \( \overline{H} \), we can apply de Morgan’s laws to obtain a DNF formula for \( \overline{H} \), the negation of the formula \( H \), with the same number of variables and clauses as the original CNF formula. The formula \( H \) has a satisfying assignment if and only if there is some assignment for the variables that does not satisfy \( \overline{H} \). Thus, \( H \) has a satisfying assignment if and only if the number of satisfying assignments of \( \overline{H} \) is strictly less than \( 2^n \), the total number of possible assignments for \( n \) Boolean variables.

We conclude that counting the number of satisfying assignments of a DNF formula is at least as hard as solving the NP-complete problem SAT.

There is a complexity class associated with the problem of counting solutions to problems in NP, denoted by \( \sharp P \) and pronounced “sharp-P”. Formally, a problem is in the class \( \sharp P \) if there is a polynomial time, nondeterministic Turing machine such that, for any input \( I \), the number of accepting computations equals the number of different solutions associated with the input \( I \). Counting the number of satisfying assignments of a DNF formula is actually \( \sharp P \)-complete; that is, this problem is as hard as any other problem in this class. Other complete problems for the class \( \sharp P \) include counting the number of Hamiltonian cycles in a graph and counting the number of perfect matchings in a bipartite graph.

It is unlikely that there is a polynomial time algorithm that computes the exact number of solutions of a \( \sharp P \)-complete problem, as at the very least such an algorithm would imply that \( P = NP \). It is therefore interesting to find an FPRAS for the number of satisfying assignments of a DNF formula.

10.2.1. The Naïve Approach

We start by trying to generalize the approach that we used to approximate \( \pi \), and we demonstrate why it is unsuitable in general. We then show how to improve our sampling technique in order to solve the problem.
THE MONTE CARLO METHOD

DNF Counting Algorithm I:

Input: A DNF formula F with n variables.
Output: Y = an approximation of c(F).

1. X ← 0.
2. For k = 1 to m, do:
   (a) Generate a random assignment for the n variables, chosen uniformly at
       random from all 2^n possible assignments.
   (b) If the random assignment satisfies F, then X ← X + 1.
3. Return Y ← (X/m)2^n.

Algorithm 10.1: DNF counting algorithm I.

Let c(F) be the number of satisfying assignments of a DNF formula F. Here we
assume that c(F) > 0, since it is easy to check whether c(F) = 0 before running
our sampling algorithm. In Section 10.1 we approximated π by generating points uni-
formly at random from the 2 × 2 square and checking to see if they were in the target:
a circle of radius 1. We try a similar approach in Algorithm 10.1: we generate assign-
ments uniformly at random for the n variables and then see if the resulting assignment
is in the target of satisfying assignments for F.

Let X_k be I if the kth iteration in the algorithm generated a satisfying assignment and
0 otherwise. Then X = \sum_{k=1}^m X_k, where the X_k are independent 0–1 random vari-
ables that each take the value 1 with probability c(F)/2^n. Hence, by linearity of expectations,

\[ E[Y] = \frac{E[X]2^n}{m} = c(F). \]

Applying Theorem 10.1, we see that X/m gives an (ε, δ)-approximation of c(F)/2^n,
and hence that Y gives an (ε, δ)-approximation of c(F), when

\[ m \geq \frac{3 \cdot 2^n \ln(2/δ)}{ε^2 c(F)}. \]

If c(F) ≥ 2^n/α(n) for some polynomial α, then the foregoing analysis tells us we
only need a number of samples m that is polynomial in n, 1/ε, and ln(1/δ). We cannot,
however, exclude the possibility that c(F) is much less than 2^n. In particular, c(F)
might be polynomial in n. Since our analysis requires a number of samples m that is
proportional to 2^n/c(F), our analysis does not yield that the run time of the algorithm
is always polynomial in the problem size.

This is not simply an artifact of the analysis. We provide a rough sketch of an ar-
gerument that is elaborated in Exercise 10.4. If the number of satisfying assignments is
polynomial in n and if at each step we sample uniformly at random from all 2^n pos-
sible assignments, then with high probability we must sample an exponential num-
ber of assignments before finding the first satisfying assignment. We can conclude, for
example, that we cannot distinguish between instances with n, n^2, and n^3 satisfying
assignments without considering exponentially many random assignments, since with high probability we would obtain zero satisfying assignments in all three cases.

The problem with this sampling approach is that the set of satisfying assignments might not be sufficiently dense in the set of all assignments. This is an additional requirement of our sampling technique that was not explicit before. In the phrasing of Theorem 10.1, the value \( \mu \) that we are attempting to approximate needs to be sufficiently large that sampling is efficient.

To obtain an FPRAS for this problem, we need to devise a better sampling scheme that avoids wasting so many steps on assignments that do not satisfy the formula. We need to construct a sample space that includes all the satisfying assignments of \( F \) and, moreover, has the property that these assignments are sufficiently dense in the sample space to allow for efficient sampling.

### 10.2.2. A Fully Polynomial Randomized Scheme for DNF Counting

We now revise our sampling procedure to obtain an FPRAS. Let \( F = C_1 \lor C_2 \lor \cdots \lor C_j \), and assume without loss of generality that no clause includes a variable and its negation. (If there is such a clause, it is not satisfiable and we can eliminate it from the formula.) A satisfying assignment of \( F \) needs to satisfy at least one of the clauses \( C_1, \ldots, C_j \). Each clause is a conjunction of literals, so there is only one assignment of the variables appearing in the clause that satisfies the clause. All other variables can have arbitrary values. For example, for the clause \( (x_1 \land \overline{x}_2 \land x_3) \) to be satisfied, \( x_1 \) and \( x_3 \) must be set to True and \( x_2 \) must be set to False.

It follows that if clause \( C_i \) has \( \ell_j \) literals then there are exactly \( 2^{n-\ell_i} \) satisfying assignments for \( C_i \). Let \( SC_i \) denote the set of assignments that satisfy clause \( i \), and let

\[
U = \{(i, a) \mid 1 \leq i \leq t \text{ and } a \in SC_i\}.
\]

Notice that we know the size of \( U \), since

\[
\sum_{i=1}^{t} |SC_i| = |U|,
\]

and we can compute \( |SC_i| \).

The value that we want to estimate is given by

\[
c(F) = \left| \bigcup_{i=1}^{t} SC_i \right|.
\]

Here \( c(F) \leq |U| \), since an assignment can satisfy more than one clause and thus appear in more than one pair in \( U \).

To estimate \( c(F) \), we define a subset \( S \) of \( U \) with size \( c(F) \). We construct this set by selecting, for each satisfying assignment of \( F \), exactly one pair in \( U \) that has this assignment; specifically, we can use the pair with the smallest clause index number, giving

\[
S = \{(i, a) \mid 1 \leq i \leq t, \ a \in SC_i, \ a \notin SC_j \text{ for } j \prec i\}.
\]

257
**THE MONTE CARLO METHOD**

**DNF Counting Algorithm II:**

**Input:** A DNF formula $F$ with $n$ variables.

**Output:** $Y$ = an approximation of $c(F)$.

1. $X \leftarrow 0$.
2. For $k = 1$ to $m$, do:
   
   (a) With probability $|SC_i|/\sum_{i=1}^{l} |SC_i|$ choose, uniformly at random, an assignment $a \in SC_i$.
   
   (b) If $a$ is not in any $SC_j$, $j < i$, then $X \leftarrow X + 1$.
3. Return $Y \leftarrow (X/m) \sum_{i=1}^{l} |SC_i|$.

*Algorithm 10.2: DNF counting algorithm II.*

Since we know the size of $U$, we can estimate the size of $S$ by estimating the ratio $|S|/|U|$. We can estimate this ratio efficiently if we sample uniformly at random from $U$ using our previous approach, choosing pairs uniformly at random from $U$ and counting how often they are in $S$. We can avoid the problem we encountered when simply sampling assignments at random, because $S$ is relatively dense in $U$. Specifically, since each assignment can satisfy at most $t$ different clauses, $|S|/|U| \geq 1/t$.

The only question left is how to sample uniformly from $U$. Suppose that we first choose the first coordinate, $i$. Because the $i$th clause has $|SC_i|$ satisfying assignments, we should choose $i$ with probability proportional to $|SC_i|$. Specifically, we should choose $i$ with probability $|SC_i|/\sum_{i=1}^{l} |SC_i|$.

We then can choose a satisfying assignment uniformly at random from $SC_i$. This is easy to do: we choose the value True or False independently and uniformly at random for each literal not in clause $i$. Then the probability that we choose the pair $(i, a)$ is

$$Pr((i, a) \text{ is chosen}) = Pr(i \text{ is chosen}) \cdot Pr(a \text{ is chosen } | i \text{ is chosen})$$

$$= \frac{|SC_i|}{|U|} \cdot \frac{1}{|SC_i|} = \frac{1}{|U|},$$

giving a uniform distribution.

These observations are implemented in Algorithm 10.2.

**Theorem 10.2:** DNF counting algorithm II is a fully polynomial randomized approximation scheme (FPRAS) for the DNF counting problem when $m = \lceil (3t/e^2) \ln(2/\delta) \rceil$.

**Proof:** Step 2(a) of the algorithm chooses an element of $U$ uniformly at random. The probability that this element belongs to $S$ is at least $1/t$. Fix any $\varepsilon > 0$ and $\delta > 0$, and let
10.3 FROM APPROXIMATE SAMPLING TO APPROXIMATE COUNTING

\[ m = \left\lceil \frac{3t}{\varepsilon^2 \ln \frac{2}{\delta}} \right\rceil. \]

Then \( m \) is polynomial in \( t, \varepsilon \), and \( \ln(1/\delta) \), and the processing time of each sample is polynomial in \( t \). By Theorem 10.1, with this number of samples, \( X/m \) gives an \((\varepsilon, \delta)\)-approximation of \( c(F)/|U| \) and hence \( Y \) gives an \((\varepsilon, \delta)\)-approximation of \( c(F) \). ■

10.3. From Approximate Sampling to Approximate Counting

The example of DNF formulas demonstrates that there is a fundamental connection between being able to sample from an appropriate space and being able to count the number of objects with some property in that space. In this section we present the outline of a general reduction that shows that, if you can sample almost uniformly a solution to a “self-reducible” combinatorial problem, then you can construct a randomized algorithm that approximately counts the number of solutions to that problem. We demonstrate this technique for the problem of counting the number of independent sets in a graph. In the next chapter, we also consider the problem of counting the number of proper colorings in a graph, applying this technique there as well.

We first need to formulate the concept of approximate uniform sampling. In this setting we are given a problem instance in the form of an input \( x \), and there is an underlying finite sample space \( \Omega(x) \) associated with the input.

**Definition 10.3:** Let \( w \) be the (random) output of a sampling algorithm for a finite sample space \( \Omega \). The sampling algorithm generates an \( \varepsilon \)-uniform sample of \( \Omega \) if, for any subset \( S \) of \( \Omega \),

\[ \left| \Pr(w \in S) - \frac{|S|}{|\Omega|} \right| \leq \varepsilon. \]

A sampling algorithm is a fully polynomial almost uniform sampler (FPAUS) for a problem if, given an input \( x \) and a parameter \( \varepsilon > 0 \), it generates an \( \varepsilon \)-uniform sample of \( \Omega(x) \) and runs in time that is polynomial in \( \ln \varepsilon^{-1} \) and the size of the input \( x \).

In the next chapter, we introduce the notion of total variation distance, which allows for an equivalent definition of an \( \varepsilon \)-uniform sample.

As an example, an FPAUS for independent sets would take as input a graph \( G = (V, E) \) and a parameter \( \varepsilon \). The sample space would be all independent sets in the graph. The output would be an \( \varepsilon \)-uniform sample of the independent sets, and the time to produce such a sample would be polynomial in the size of the graph and \( \ln \varepsilon^{-1} \). In fact, in the reduction that follows we only need the running time to be polynomial in \( \varepsilon^{-1} \), but we use the standard definition given in Definition 10.3.

Our goal is to show that, given an FPAUS for independent sets, we can construct an FPRAS for counting the number of independent sets. Assume that the input \( G \) has \( m \) edges, and let \( \varepsilon_1, \ldots, \varepsilon_m \) be an arbitrary ordering of the edges. Let \( E_i \) be the set of the first \( i \) edges in \( E \) and let \( G_i = (V, E_i) \). Note that \( G = G_m \) and that \( G_{i-1} \) is obtained from \( G_i \) by removing a single edge.

259
THE MONTE CARLO METHOD

We let $\Omega(G_i)$ denote the set of independent sets in $G_i$. The number of independent sets in $G$ can then be expressed as

$$|\Omega(G)| = \frac{|\Omega(G_m)|}{|\Omega(G_{m-1})|} \times \frac{|\Omega(G_{m-1})|}{|\Omega(G_{m-2})|} \times \frac{|\Omega(G_{m-2})|}{|\Omega(G_{m-3})|} \times \cdots \times \frac{|\Omega(G_1)|}{|\Omega(G_0)|} \times |\Omega(G_0)|.$$  

Since $G_0$ has no edges, every subset of $V$ is an independent set and $\Omega(G_0) = 2^n$. In order to estimate $|\Omega(G)|$, we need good estimates for the ratios

$$r_i = \frac{|\Omega(G_i)|}{|\Omega(G_{i-1})|}, \quad i = 1, \ldots, m.$$  

More formally, we will develop estimates $\tilde{r}_i$ for the ratios $r_i$, and then our estimate for the number of independent sets in $G$ will be

$$2^n \prod_{i=1}^{m} \tilde{r}_i,$$

while the true number is

$$|\Omega(G)| = 2^n \prod_{i=1}^{m} r_i.$$  

To evaluate the error in our estimate, we need to bound the ratio

$$R = \prod_{i=1}^{m} \frac{\tilde{r}_i}{r_i}.$$  

Specifically, to have an $(\varepsilon, \delta)$-approximation, we want $\Pr(|R - 1| \leq \varepsilon) \geq 1 - \delta$. We will make use of the following lemma.

**Lemma 10.3:** Suppose that for all $i$, $1 \leq i \leq m$, $\tilde{r}_i$ is an $(\varepsilon/2m, \delta/m)$-approximation for $r_i$. Then

$$\Pr(|R - 1| \leq \varepsilon) \geq 1 - \delta.$$  

**Proof:** For each $1 \leq i \leq m$, we have

$$\Pr\left(\left|\tilde{r}_i - r_i\right| \leq \frac{\varepsilon}{2m} r_i\right) \geq 1 - \frac{\delta}{m}.$$  

Equivalently,

$$\Pr\left(\left|\tilde{r}_i - r_i\right| > \frac{\varepsilon}{2m} r_i\right) < \frac{\delta}{m}.$$  

By the union bound, the probability that $\left|\tilde{r}_i - r_i\right| > (\varepsilon/2m)r_i$ for any $i$ is at most $\delta$; therefore, $\left|\tilde{r}_i - r_i\right| \leq (\varepsilon/2m) r_i$ for all $i$ with probability at least $1 - \delta$. Equivalently,

$$1 - \frac{\varepsilon}{2m} \leq \frac{\tilde{r}_i}{r_i} \leq 1 + \frac{\varepsilon}{2m}$$

holds for all $i$ with probability at least $1 - \delta$. When these bounds hold for all $i$, we can combine them to obtain

260
Estimating $r_i$:

**Input:** Graphs $G_{i-1} = (V, E_{i-1})$ and $G_i = (V, E_i)$.

**Output:** $\tilde{r}_i$ = an approximation of $r_i$.

1. $X \leftarrow 0$.
2. Repeat for $M = \lceil 1296m^2\varepsilon^{-2} \ln (2m/\delta) \rceil$ independent trials:
   
   (a) Generate an $(\varepsilon/6m)$-uniform sample from $\Omega(G_{i-1})$.
   
   (b) If the sample is an independent set in $G_i$, let $X \leftarrow X + 1$.
3. Return $\tilde{r}_i \leftarrow X/M$.

Algorithm 10.3: Estimating $r_i$.

\[
1 - \varepsilon \leq \left( 1 - \frac{\varepsilon}{2m} \right)^{m} \leq \prod_{i=1}^{m} \frac{\tilde{r}_i}{r_i} \leq \left( 1 - \frac{\varepsilon}{2m} \right)^{m} \leq 1 + \varepsilon. 
\]

giving the lemma. ■

Hence all we need is a method for obtaining an $(\varepsilon/2m, \delta/m)$-approximation for the $r_i$.

We estimate each of these ratios by a Monte Carlo algorithm that uses the FPAUCS for sampling independent sets. To estimate $r_i$, we sample independent sets in $G_{i-1}$ and compute the fraction of these sets that are also independent sets in $G_i$, as described in Algorithm 10.3. The constants in the procedure were chosen to facilitate the proof of Lemma 10.4.

**Lemma 10.4:** When $m \geq 1$ and $0 < \varepsilon \leq 1$, the procedure for estimating $r_i$ yields an $(\varepsilon/2m, \delta/m)$-approximation for $r_i$.

**Proof:** We first show that $r_i$ is not too small, avoiding the problem that we found in Section 10.2.1. Suppose that $G_{i-1}$ and $G_i$ differ in that edge $\{u, v\}$ is present in $G_i$ but not in $G_{i-1}$. An independent set in $G_i$ is also an independent set in $G_{i-1}$, so

$\Omega(G_i) \subseteq \Omega(G_{i-1}).$

An independent set in $\Omega(G_{i-1}) \setminus \Omega(G_i)$ contains both $u$ and $v$. To bound the size of the set $\Omega(G_{i-1}) \setminus \Omega(G_i)$, we associate each $I \in \Omega(G_{i-1}) \setminus \Omega(G_i)$ with an independent set $I \setminus \{v\} \in \Omega(G_i)$. In this mapping an independent set $I' \in \Omega(G_i)$ is associated with no more than one independent set $I' \cup \{v\} \in \Omega(G_{i-1}) \setminus \Omega(G_i)$, and thus $\Omega(G_{i-1}) \setminus \Omega(G_i) \leq |\Omega(G_i)|$. It follows that

\[
r_i = \frac{|\Omega(G_i)|}{|\Omega(G_{i-1})|} = \frac{|\Omega(G_i)|}{|\Omega(G_i)| + |\Omega(G_{i-1}) \setminus \Omega(G_i)|} \geq \frac{1}{2}.
\]

Now consider our $M$ samples, and let $X_k = 1$ if the $k$th sample is in $\Omega(G_i)$ and 0 otherwise. Because our samples are generated by an $(\varepsilon/6m)$-uniform sampler, by Definition 10.3 each $X_i$ must satisfy

261
THE MONTE CARLO METHOD

\[ \Pr(X_k = 1) - \frac{|\Omega(G_i)|}{|\Omega(G_{i-1})|} \leq \frac{\varepsilon}{6m}. \]

Since the \( X_k \) are indicator random variables, it follows that

\[ \left| \mathbb{E}[X_k] - \frac{|\Omega(G_i)|}{|\Omega(G_{i-1})|} \right| \leq \frac{\varepsilon}{6m} \]

and further, by linearity of expectations,

\[ \left| \mathbb{E}\left[ \frac{\sum_{k=1}^{M} X_k}{M} \right] - \frac{|\Omega(G_i)|}{|\Omega(G_{i-1})|} \right| \leq \frac{\varepsilon}{6m}. \]

We therefore have

\[ |\mathbb{E}[\tilde{r}_i] - r_i| = \left| \mathbb{E}\left[ \frac{\sum_{k=1}^{M} X_k}{M} \right] - \frac{|\Omega(G_i)|}{|\Omega(G_{i-1})|} \right| \leq \frac{\varepsilon}{6m}. \]

We now complete the lemma by combining (a) the fact just shown that \( \mathbb{E}[\tilde{r}_i] \) is close to \( r_i \) and (b) the fact that \( \tilde{r}_i \) will be close to \( \mathbb{E}[\tilde{r}_i] \) for a sufficiently large number of samples. Using \( r_i \geq 1/2 \), we have

\[ \mathbb{E}[\tilde{r}_i] \geq r_i - \frac{\varepsilon}{6m} \geq \frac{1}{2} - \frac{\varepsilon}{6m} \geq \frac{1}{3}. \]

Applying Theorem 10.1 yields that, if the number of samples \( M \) satisfies

\[ M \geq \frac{3 \ln(2m/\delta)}{(\varepsilon/12m)^2(1/3)} = 1296m^2 \varepsilon^{-2} \ln \frac{2m}{\delta}, \]

then

\[ \Pr\left( \left| \frac{\tilde{r}_i}{\mathbb{E}[\tilde{r}_i]} - 1 \right| \geq \frac{\varepsilon}{12m} \right) = \Pr\left( |r_i - \mathbb{E}[\tilde{r}_i]| \geq \frac{\varepsilon}{12m} \mathbb{E}[\tilde{r}_i] \right) \leq \frac{\delta}{m}. \]

Equivalently, with probability \( 1 - \delta/m \),

\[ 1 - \frac{\varepsilon}{12m} \leq \frac{\tilde{r}_i}{\mathbb{E}[\tilde{r}_i]} \leq 1 + \frac{\varepsilon}{12m}. \quad (10.1) \]

As \( |\mathbb{E}[\tilde{r}_i] - r_i| \leq \varepsilon/6m \), we have that

\[ 1 - \frac{\varepsilon}{6mr_i} \leq \frac{\mathbb{E}[\tilde{r}_i]}{r_i} \leq 1 + \frac{\varepsilon}{6mr_i}. \]

Using that \( r_i \geq 1/2 \) then yields

\[ 1 - \frac{\varepsilon}{3m} \leq \frac{\mathbb{E}[\tilde{r}_i]}{r_i} \leq 1 + \frac{\varepsilon}{3m}. \quad (10.2) \]

Combining equations (10.1) and (10.2), it follows that, with probability \( 1 - \delta/m \),

\[ 1 - \frac{\varepsilon}{2m} \leq \left( 1 - \frac{\varepsilon}{3m} \right) \left( 1 - \frac{\varepsilon}{12m} \right) \leq \frac{\tilde{r}_i}{r_i} \leq \left( 1 + \frac{\varepsilon}{3m} \right) \left( 1 + \frac{\varepsilon}{12m} \right) \leq 1 + \frac{\varepsilon}{2m}. \]

This gives the desired \((\varepsilon/2m, \delta/m)\)-approximation. \( \blacksquare \)
10.4 The Markov Chain Monte Carlo Method

The number of samples $M$ is polynomial in $m$, $\varepsilon$, and $\ln \delta^{-1}$, and the time for each sample is polynomial in the size of the graph and $\ln \varepsilon^{-1}$. We therefore have the following theorem.

**Theorem 10.5:** Given a fully polynomial almost uniform sampler (FPAS) for independent sets in any graph, we can construct a fully polynomial randomized approximation scheme (FPRAS) for the number of independent sets in a graph $G$.

In fact, this theorem is more often used in the following form.

**Theorem 10.6:** Given a fully polynomial almost uniform sampler (FPAS) for independent sets in any graph with maximum degree at most $\Delta$, we can construct a fully polynomial randomized approximation scheme (FPRAS) for the number of independent sets in a graph $G$ with maximum degree at most $\Delta$.

This version of the theorem follows from our previous argument, since our graphs $G_i$ are subgraphs of the initial graph $G$. Hence, if we start with a graph of maximum degree at most $\Delta$, then our FPAS need only work on graphs with maximum degree at most $\Delta$. In the next chapter, we will see how to create an FPAS for graphs with maximum degree 4.

This technique can be applied to a broad range of combinatorial counting problems. For example, in Chapter 11 we consider its application to finding proper colorings of a graph $G$. The only requirement is that we can construct a sequence of refinements of the problem, starting with an instance that is easy to count (the number of independent sets in a graph with no edges, in our example) and ending with the actual counting problem, and such that the ratio between the counts in successive instances is at most polynomial in the size of the problem.

10.4. The Markov Chain Monte Carlo Method

The Monte Carlo method is based on sampling. It is often difficult to generate a random sample with the required probability distribution. For example, we saw in the previous section that we can count the number of independent sets in a graph if we can generate an almost uniform sample from the set of independent sets. But how can we generate an almost uniform sample?

The Markov chain Monte Carlo (MCMC) method provides a very general approach to sampling from a desired probability distribution. The basic idea is to define an ergodic Markov chain whose set of states is the sample space and whose stationary distribution is the required sampling distribution. Let $X_0, X_1, \ldots, X_n$ be a run of the chain. The Markov chain converges to the stationary distribution from any starting state $X_0$ and so, after a sufficiently large number of steps $r$, the distribution of the state $X_r$ will be close to the stationary distribution, so it can be used as a sample. Similarly, repeating this argument with $X_r$ as the starting point, we can use $X_{2r}$ as a sample, and so on. We can therefore use the sequence of states $X_r, X_{2r}, X_{3r}, \ldots$ as almost independent samples from the stationary distribution of the Markov chain. The efficiency of
this approach depends on (a) how large \( r \) must be to ensure a suitably good sample and (b) how much computation is required for each step of the Markov chain. In this section, we focus on finding efficient Markov chains with the appropriate stationary distribution and ignore the issue of how large \( r \) needs to be. Coupling, which is one method for determining the relationship between the value of \( r \) and the quality of the sample, is discussed in the next chapter.

In the simplest case, the goal is to construct a Markov chain with a stationary distribution that is uniform over the state space \( \Omega \). The first step is to design a set of moves that ensures the state space is irreducible under the Markov chain. Let us call the set of states reachable in one step from a state \( x \) (but excluding \( x \)) the neighbors of \( x \), denoted by \( N(x) \). We adopt the restriction that if \( y \in N(x) \) then also \( x \in N(y) \). Generally \( N(x) \) will be a small set, so that performing each move is simple computationally.

We again use the setting of independent sets in a graph \( G = (V, E) \) as an example. The state space is all of the independent sets of \( G \). A natural neighborhood framework is to say that states \( x \) and \( y \), which are independent sets, are neighbors if they differ in just one vertex. That is, \( x \) can be obtained from \( y \) by adding or deleting just one vertex. This neighbor relationship guarantees that the state space is irreducible, since all independent sets can reach (respectively, can be reached from) the empty independent set by a sequence of vertex deletions (respectively, vertex additions).

Once the neighborhoods are established, we need to establish transition probabilities. One natural approach to try would be performing a random walk on the graph of the state space. This might not lead to a uniform distribution, however. We saw in Theorem 7.13 that, in the stationary distribution of a random walk, the probability of a vertex is proportional to the degree of the vertex. Nothing in our previous discussion requires all states to have the same number of neighbors, which is equivalent to all vertices in the graph of the state space having the same degree.

The following lemma shows that, if we modify the random walk by giving each vertex an appropriate self-loop probability, then we can obtain a uniform stationary distribution.

**Lemma 10.7:** For a finite state space \( \Omega \) and neighborhood structure \( \{N(x) \mid x \in \Omega\} \), let \( N = \max_{x \in \Omega} |N(x)| \). Let \( M \) be any number such that \( M \geq N \). Consider a Markov chain where

\[
P_{x,x} = \begin{cases} 
1/M & \text{if } x \neq y \text{ and } y \in N(x), \\
0 & \text{if } x \neq y \text{ and } y \notin N(x), \\
1 - N(x)/M & \text{if } x = y.
\end{cases}
\]

If this chain is irreducible and aperiodic, then the stationary distribution is the uniform distribution.

**Proof:** We show that the chain is time reversible and then apply Theorem 7.10. For any \( x \neq y \), if \( \pi_x = \pi_y \), then

\[
\pi_x P_{x,x} = \pi_y P_{y,x},
\]
since \( P_{x,y} = P_{x,x} = 1/M \). It follows that the uniform distribution \( \pi_x = 1/|\Omega| \) is the stationary distribution.

Consider now the following simple Markov chain, whose states are independent sets in a graph \( G = (V, E) \).

1. \( X_0 \) is an arbitrary independent set in \( G \).
2. To compute \( X_{i+1} \):
   - (a) choose a vertex \( v \) uniformly at random from \( V \):
   - (b) if \( v \in X_i \) then \( X_{i+1} = X_i \setminus \{v\} \):
   - (c) if \( v \notin X_i \) and if adding \( v \) to \( X_i \) still gives an independent set, then \( X_{i+1} = X_i \cup \{v\} \):
   - (d) otherwise, \( X_{i+1} = X_i \).

This chain has the property that the neighbors of a state \( X_i \) are all independent sets that differ from \( X_i \) in just one vertex. Since every state can reach and is reachable from the empty set, the chain is irreducible. Assuming that \( G \) has at least one edge \((u,v)\), then the state \( \{v\} \) has a self-loop \( (P_{v,v} > 0) \), and the chain is aperiodic. Further, when \( y \neq x \), it follows that \( P_{x,y} = 1/|V| \) or 0. Lemma 10.7 therefore applies, and the stationary distribution is the uniform distribution.

10.4.1. The Metropolis Algorithm

We have seen how to construct chains with a uniform stationary distribution. In some cases, however, we may want to sample from a chain with a nonuniform stationary distribution. The Metropolis algorithm refers to a general construction that transforms any irreducible Markov chain on a finite state space \( \Omega \) to a time-reversible Markov chain with a required stationary distribution. The approach generalizes the idea we used before to create chains with uniform stationary distributions: add self-loop probabilities to states in order to obtain the desired stationary distribution.

Let us again assume that we have designed an irreducible state space for our Markov chain; now we want to construct a Markov chain on this state space with a stationary distribution \( \pi_x = b(x)/B \), where for all \( x \in \Omega \) we have \( b(x) > 0 \) and such that \( B = \sum_{x \in \Omega} b(x) \) is finite. As we see in the following lemma (which generalizes Lemma 10.7), we only need the ratios between the required probabilities; the sum \( B \) can be unknown.

**Lemma 10.8:** For a finite state space \( \Omega \) and neighborhood structure \( \{N(x) \mid x \in \Omega\} \), let \( N = \max_{x \in \Omega} |N(x)| \). Let \( M \) be any number such that \( M \geq N \). For all \( x \in \Omega \), let \( \pi_x > 0 \) be the desired probability of state \( x \) in the stationary distribution. Consider a Markov chain where

\[
P_{x,y} = \begin{cases} 
(1/M) \min(1, \pi_y/\pi_x) & \text{if } x \neq y \text{ and } y \in N(x), \\
0 & \text{if } x \neq y \text{ and } y \notin N(x), \\
1 - \sum_{y \neq x} P_{x,y} & \text{if } x = y.
\end{cases}
\]
THE MONTE CARLO METHOD

Then, if this chain is irreducible and aperiodic, the stationary distribution is given by the probabilities $\pi_x$.

**Proof:** As in the proof of Lemma 10.7, we show that the chain is time reversible and apply Theorem 7.10. For any $x \neq y$, if $\pi_x \leq \pi_y$ then $P_{x,y} = 1$ and $P_{y,x} = \pi_x/\pi_y$. It follows that $\pi_y P_{x,y} = \pi_x P_{y,x}$. Similarly, if $\pi_y > \pi_y$ then $P_{x,y} = \pi_y/\pi_x$ and $P_{y,x} = 1$, and it follows that $\pi_x P_{x,y} = \pi_y P_{y,x}$. By Theorem 7.10, the stationary distribution is given by the values $\pi_x$. ■

As an example of how to apply Lemma 10.8, let us consider how to modify our previous Markov chains on independent sets. Let us suppose that now we want to create a Markov chain where, in the stationary distribution, each independent set $I$ has probability proportional to $\lambda^{|I|}$ for some constant parameter $\lambda > 0$. That is, $\pi_x = \lambda^{|I_x|}/B$, where $I_x$ is the independent set corresponding to state $x$ and where $B = \sum_x \lambda^{|I_x|}$. When $\lambda = 1$, this is the uniform distribution; when $\lambda > 1$, larger independent sets have a larger probability than smaller independent sets; and when $\lambda < 1$, larger independent sets have a smaller probability than smaller independent sets.

Consider now the following variation on the previous Markov chain for independent sets in a graph $G = (V,E)$.

1. $X_0$ is an arbitrary independent set in $G$.
2. To compute $X_{i+1}$:
   - **(a)** choose a vertex $v$ uniformly at random from $V$;
   - **(b)** if $v \in X_i$, set $X_{i+1} = X_i \setminus \{v\}$ with probability $\min(1, 1/\lambda)$;
   - **(c)** if $v \notin X_i$ and if adding $v$ to $X_i$ still gives an independent set, then put $X_{i+1} = X_i \cup \{v\}$ with probability $\min(1, \lambda)$;
   - **(d)** otherwise, set $X_{i+1} = X_i$.

We now follow a two-step approach. We first propose a move by choosing a vertex $v$ to add or delete, where each vertex is chosen with probability $1/M$, here $M = |V|$. This proposal is then accepted with probability $\min(1, \pi_x/\pi_y)$, where $x$ is the current state and $y$ is the proposed state to which the chain will move. Here, $\pi_x/\pi_y$ is $\lambda$ if the chain attempts to add a vertex and is $1/\lambda$ if the chain attempts to delete a vertex. This two-step approach is the hallmark of the Metropolis algorithm: each neighbor is selected with probability $1/M$, and then it is accepted with probability $\min(1, \pi_y/\pi_x)$. Using this two-step approach, we naturally obtain that the transition probability $P_{x,y}$ is

$$P_{x,y} = \frac{1}{M} \min\left(1, \frac{\pi_y}{\pi_x}\right),$$

so Lemma 10.8 applies.

It is important that, in designing this Markov chain, we never needed to know $B = \sum \lambda^{|I|}$. A graph with $n$ vertices can have exponentially many independent sets, and calculating this sum directly would be too expensive computationally for many graphs. Our Markov chain gives the correct stationary distribution by using the ratios $\pi_y/\pi_x$, which are much easier to deal with.

266