FROM: WERNER KRAUTH, ALGORITHMS AND COMPUTATIONS Oxford UP

1.1 Popular games in Monaco

The concept of sampling (obtaining the random positions) is truly complex, and we had better get a grasp of the idea in a simplified setting before applying it in its full power and versatility to the complicated cases of later chapters. We must clearly distinguish between two fundamentally different sampling approaches: direct sampling and Markov-chain sampling.

1.1.1 Direct sampling

Direct sampling is exemplified by an amusing game that we can imagine children playing on the beaches of Monaco. In the sand, they first draw a large circle and a square exactly containing it (see Fig. 1.1). They then randomly throw pebbles.[2] Each pebble falling inside the square constitutes a trial, and pebbles inside the circle are also counted as "hits".

By keeping track of the numbers of trials and hits, the children perform a direct-sampling Monte Carlo calculation: the ratio of hits to trials is close to the ratio of the areas of the circle and the square, namely $\pi/4$. The other day, in a game of 4000 trials, they threw 3156 pebbles inside the circle (see Table 1.1). This means that they got 3156 hits, and obtained the approximation $\pi \simeq 3.156$ by just shifting the decimal point.

Let us write up the children's game in a few lines of computer code (see Alg. 1.1 (`direct-pi`)). As it is difficult to agree on language and dialect, we use the universal *pseudocode* throughout this book. Readers can then translate the general algorithms into their favorite programming language, and are strongly encouraged to do so. Suffice it to say here that calls to the function $\texttt{ran}\,(-1,1)$ produce uniformly distributed real random numbers between -1 and 1. Subsequent calls yield independent numbers.

procedure `direct-pi`
$N_{\text{hits}} \leftarrow 0$ (initialize)
for $i = 1, \ldots, N$ **do**
  $x \leftarrow \texttt{ran}\,(-1,1)$
  $y \leftarrow \texttt{ran}\,(-1,1)$
  if $(x^2 + y^2 < 1)$ $N_{\text{hits}} \leftarrow N_{\text{hits}} + 1$
output N_{hits}

Algorithm 1.1 `direct-pi`. Using the children's game with N pebbles to compute π.

The results of several runs of Alg. 1.1 (`direct-pi`) are shown in Table 1.1. During each trial, $N = 4000$ pebbles were thrown, but the ran-

[2]The Latin word for "pebble" is *calculus*.

Table 1.1 Results of five runs of Alg. 1.1 (`direct-pi`) with $N = 4000$

Run	N_{hits}	Estimate of π
1	3156	3.156
2	3150	3.150
3	3127	3.127
4	3171	3.171
5	3148	3.148

dom numbers differed, i.e. the pebbles landed at different locations each run.

We shall return later to this table when computing the statistical rors to be expected from Monte Carlo calculations. In the meantime, intend to show that the Monte Carlo method is a powerful approach the calculation of integrals (in mathematics, physics, and other field But let us not get carried away: none of the results in Table 1.1 h fallen within the tight error bounds already known since Archimec from comparing a circle with regular n-gons:

$$3.141 \simeq 3\frac{10}{71} < \pi < 3\frac{1}{7} \simeq 3.143. \quad (1.$$

The children's value for π is very approximate, but improves and fina becomes exact in the limit of an infinite number of trials. This is Jac Bernoulli's weak law of large numbers (see Subsection 1.3.2). The ch dren also adopt a very sensible rule: they decide on the total number throws before starting the game. The other day, in a game of "N=400(they had at some point 355 hits for 452 trials—this gives a very nice a proximation to the book value of π. Without hesitation, they went until the 4000th pebble was cast. They understand that one must n stop a stochastic calculation simply because the result is just right, n should one continue to play because the result is not close enough what we think the answer should be.

$$\frac{355}{452} = \frac{355}{4 \times 113} = \tfrac{1}{4} \times 3.14159292\ldots$$
$$\pi/4 = \tfrac{1}{4} \times 3.14159265\ldots$$

1.1.2 Markov-chain sampling

In Monte Carlo, it is not only children who play at pebble games. V can imagine that adults, too, may play their own version at the loc heliport, in the late evenings. After stowing away all their helicopte they wander around the square-shaped landing pad (Fig. 1.2), whi looks just like the area in the children's game, only bigger.

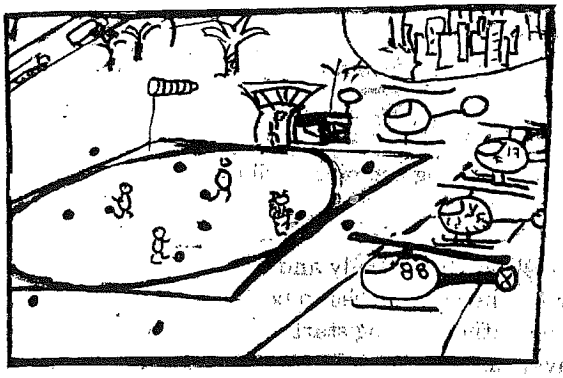

Fig. 1.2 Adults computing the number π at the Monte Carlo heliport.

but does not change the fact that π comes out as four times the ratio o hits to trials.

Those who hear this story for the first time often find it dubious. They observe that perhaps one should not pile up stones, as in Fig. 1.3, if the aim is to spread them out evenly. This objection places these modern critics in the illustrious company of prominent physicists and mathematicians who questioned the validity of this method when it was first published in 1953 (it was applied to the hard-disk system of Chapter 2). Letters were written, arguments were exchanged, and the issue was settled only after several months. Of course, at the time, helicopters and heliports were much less common than they are today.

A proof of correctness and an understanding of this method, called the Metropolis algorithm, will follow later, in Subsection 1.1.4. Here, we start by programming the adults' algorithm according to the above prescription: go from one configuration to the next by following a random throw:

$$\Delta_x \leftarrow \texttt{ran}\,(-\delta, \delta)\,,$$
$$\Delta_y \leftarrow \texttt{ran}\,(-\delta, \delta)$$

(see Alg. 1.2 (`markov-pi`)). Any move that would take us outside the pad is rejected: we do not move, and count the configuration a second time (see Fig. 1.4).

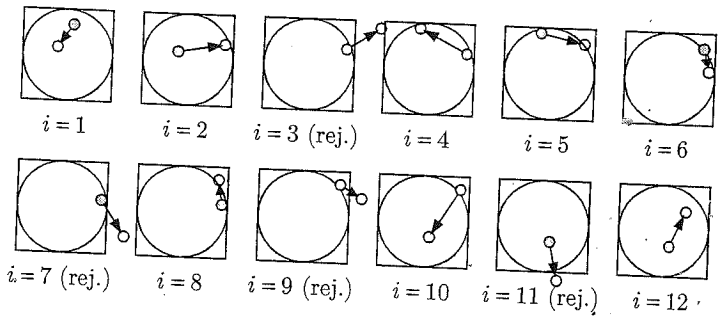

Fig. 1.4 Simulation of Alg. 1.2 (`markov-pi`). A rejection leaves the configuration unchanged (see frames $i = 3, 7, 9, 11$).

Table 1.2 Results of five runs of Alg. 1.2 (`markov-pi`) with $N = 4000$ and a throwing range $\delta = 0.3$

Run	N_{hits}	Estimate of π
1	3123	3.123
2	3118	3.118
3	3040	3.040
4	3066	3.066
5	3263	3.263

Table 1.2 shows the number of hits produced by Alg. 1.2 (`markov-pi`) in several runs, using each time no fewer than $N = 4000$ digital pebbles taken from the lady's bag. The results scatter around the number $\pi = 3.1415\ldots$, and we might be more inclined to admit that the idea of piling up pebbles is probably correct, even though the spread of the data, for an identical number of pebbles, is much larger than for the direct-sampling method (see Table 1.1).

In Alg. 1.2 (`markov-pi`), the throwing range δ, that is to be kept fixed throughout the simulation, should not be made too small: for $\delta \gtrsim 0$, the acceptance rate is high, but the path traveled per step is small. On the other hand, if δ is too large, we also run into problems: for a large range $\delta \gg 1$, most moves would take us outside the pad. Now, the acceptance

```
procedure markov-pi(patch)
input {x, y} (configuration i)
Δ_x ← ...
Δ_y ← ...
⋮
output {x, y} (configuration i + 1)
```

Algorithm 1.3 `markov-pi(patch)`. Going from one configuration to the next, in the Markov-chain Monte Carlo algorithm.

circle and 0 elsewhere (see Fig. 1.5). In both cases, one evaluates

$$\underbrace{\frac{N_{\text{hits}}}{\text{trials}} = \frac{1}{N}\sum_{i=1}^{N} \mathcal{O}_i}_{\text{sampling}} \simeq \underbrace{\langle \mathcal{O} \rangle = \frac{\int_{-1}^{1} dx \int_{-1}^{1} dy\, \pi(x,y)\mathcal{O}(x,y)}{\int_{-1}^{1} dx \int_{-1}^{1} dy\, \pi(x,y)}}_{\text{integration}} . \qquad (1.$$

The probability distribution $\pi(x, y)$ no longer appears on the left: rathe than being evaluated, it is sampled. This is what defines the Monte Carl method. On the left of eqn (1.2), the multiple integrals have disappeared This means that the Monte Carlo method allows the evaluation of high- dimensional integrals, such as appear in statistical physics and other domains, if only we can think of how to generate the samples.

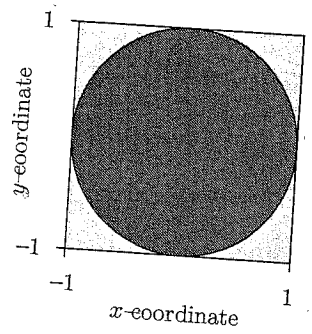

Fig. 1.5 Probability density ($\pi = 1$ inside square, zero outside) and observable ($\mathcal{O} = 1$ inside circle, zero outside) in the Monte Carlo games.

Direct sampling, the approach inspired by the children's game, is like pure gold: a subroutine provides an independent hit at the distribution function $\pi(\mathbf{x})$, that is, it generates vectors $\mathbf{x}$ with a probability proportional to $\pi(\mathbf{x})$. Notwithstanding the randomness in the problem, direct sampling, in computation, plays a role similar to exact solutions in analytical work, and the two are closely related. In direct sampling, there is no throwing-range issue, no worrying about initial conditions (the clubhouse), and a straightforward error analysis—at least if $\pi(\mathbf{x})$ and $\mathcal{O}(\mathbf{x})$

are well behaved. Many successful Monte Carlo algorithms contain exact sampling as a key ingredient.

Markov-chain sampling, on the other hand, forces us to be much more careful with all aspects of our calculation. The critical issue here is the correlation time, during which the pebble keeps a memory of the starting configuration, the clubhouse. This time can become astronomical. In the usual applications, one is often satisfied with a handful of independent samples, obtained through week-long calculations, but it can require much thought and experience to ensure that even this modest goal is achieved. We shall continue our discussion of Markov-chain Monte Carlo methods in Subsection 1.1.4, but want to first take a brief look at the history of stochastic computing.

1.1.3 Historical origins

The idea of direct sampling was introduced into modern science in the late 1940s by the mathematician Ulam, not without pride, as one can find out from his autobiography *Adventures of a Mathematician* (Ulam (1991)). Much earlier, in 1777, the French naturalist Buffon (1707–1788) imagined a legendary needle-throwing experiment, and analyzed it completely. All through the eighteenth and nineteenth centuries, royal courts and learned circles were intrigued by this game, and the theory was developed further. After a basic treatment of the Buffon needle problem, we shall describe the particularly brilliant idea of Barbier (1860), which foreshadows modern techniques of variance reduction.

The Count is shown in Fig. 1.6 randomly throwing needles of length a onto a wooden floor with cracks a distance b apart. We introduce

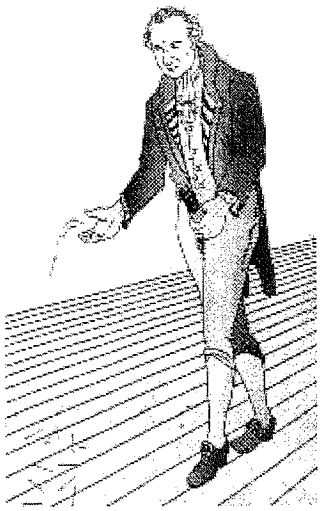

Fig. 1.6 Georges Louis Leclerc, Count of Buffon (1707–1788), performing the first recorded Monte Carlo simulation, in 1777. (Published with permission of *Le Monde*.)

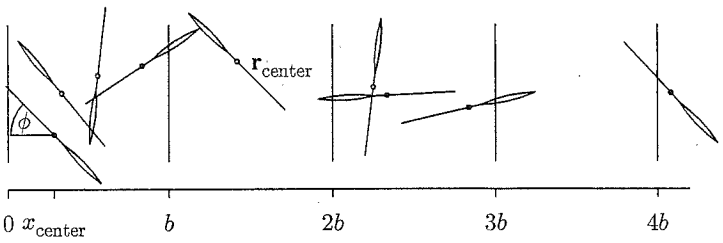

Fig. 1.7 Variables x_{center} and ϕ in Buffon's needle experiment. The needles are of length a.

coordinates $\mathbf{r}_{\text{center}}$ and ϕ as in Fig. 1.7, and assume that the needles' centers $\mathbf{r}_{\text{center}}$ are uniformly distributed on an infinite floor. The needles do not roll into cracks, as they do in real life, nor do they interact with each other. Furthermore, the angle ϕ is uniformly distributed between 0 and 2π. This is the mathematical model for Buffon's experiment.

All the cracks in the floor are equivalent, and there are symmetries $x_{\text{center}} \leftrightarrow b - x_{\text{center}}$ and $\phi \leftrightarrow -\phi$. The variable y is irrelevant to the

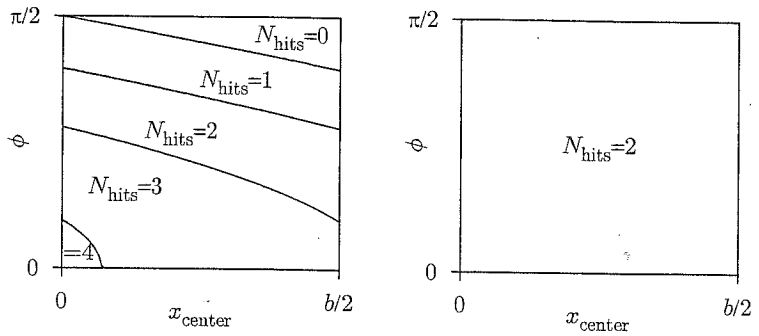

Fig. 1.15 "Landing pads" for the Buffon needle experiment with $a = \pi b$. *Left*: straight needles. *Right*: crazy cobbler's needles.

differ, and only the mean numbers of hits (the mean of N_{hits} over the whole pad) agree.

Barbier's trick is an early example of variance reduction, a powerful strategy for increasing the precision of Monte Carlo calculations. It comes in many different guises and shows that there is great freedom in finding the optimal setup for a computation.

1.1.4 Detailed balance

We left the lady and the heliport, in Subsection 1.1.2, without clarifying why the strange piles in Fig. 1.3 had to be built. Instead of the heliport game, let us concentrate on a simplified discrete version, the 3×3 pebble game shown in Fig. 1.17. The pebble can move in at most four directions: up, down, left, and right. In this subsection, we perform a complete analysis of Markov-chain algorithms for the pebble game, which is easily generalized to the heliport.

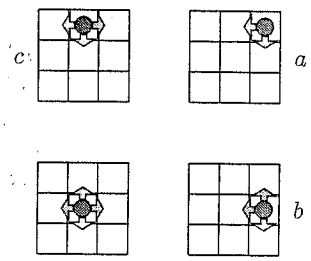

Fig. 1.17 Discrete pebble game. The corner configuration a is in contact with configurations b and c.

We seek an algorithm for moving the pebble one step at a time such

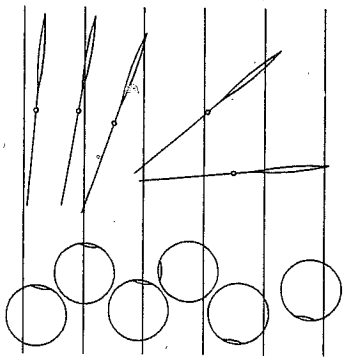

Fig. 1.16 Straight needles of length πb, with between zero and four hits, and round (crazy cobbler's) needles, which always hit twice.

that, after many iterations, it appears with the same probability in of the fields. Anyone naive who had never watched ladies at heli would simply chuck the pebble a few times in a random direction, i.e. of four directions from the center, one of three directions from the ec or one of two directions from the corners. But this natural algori is wrong. To understand why we must build piles, let us consider corner configuration a, which is in contact with the configurations b c (see Fig. 1.17). Our algorithm (yet to be found) must generate configurations a, b, and c with prescribed probabilities $\pi(a)$, $\pi(b)$, $\pi(c)$, respectively, which we require to be equal. This means that want to create these configurations with probabilities

$$\{\pi(a), \pi(b), \ldots\} : \left\{ \begin{array}{c} \text{stationary probability} \\ \text{for the system to be at } a, b, \text{ etc.} \end{array} \right\}, \qquad (1.$$

with the help of our Monte Carlo algorithm, which is nothing but a of transition probabilities $p(a \to b)$ for moving from one configurat to the other (from a to b),

$$\{p(a \to b), p(a \to c), \ldots\} : \left\{ \begin{array}{c} \text{probability of the algorithm} \\ \text{to move from } a \text{ to } b, \text{ etc.} \end{array} \right\}.$$

Furthermore, we enforce a normalization condition which tells us t the pebble, once at a, can either stay there or move on to b or c:

$$p(a \to a) + p(a \to b) + p(a \to c) = 1. \qquad (1.$$

The two types of probabilities can be linked by observing that the c figuration a can only be generated from b or c or from itself:

$$\pi(a) = \pi(b)p(b \to a) + \pi(c)p(c \to a) + \pi(a)p(a \to a), \qquad (1.1$$

which gives

$$\pi(a)[1 - p(a \to a)] = \pi(b)p(b \to a) + \pi(c)p(c \to a).$$

Writing eqn (1.10) as $1 - p(a \to a) = p(a \to b) + p(a \to c)$ and introduci it into the last equation yields

$$\pi(a)\underbrace{p(a \to b) + \pi(a)\overbrace{p(a \to c)}^{=\pi(c)}}p(c \to a) + \pi(b)p(b \to a).$$

This equation can be satisfied by equating the braced terms separatel and thus we arrive at the crucial condition of detailed balance,

$$\left\{ \begin{array}{c} \text{detailed} \\ \text{balance} \end{array} \right\} : \begin{array}{l} \pi(a)p(a \to b) = \pi(b)p(b \to a) \\ \pi(a)p(a \to c) = \pi(c)p(c \to a) \end{array} \quad \text{etc.} \qquad (1.1$$

This rate equation renders consistent the Monte Carlo algorithm (th probabilities $\{p(a \to b)\}$) and the prescribed stationary probabilitie $\{\pi(a), \pi(b), \ldots\}$.

In the pebble game, detailed balance is satisfied because all proba bilities for moving between neighboring sites are equal to $1/4$, and th

```
procedure markov-discrete-pebble
input k (position of pebble)
n ← nran(1, 4)
if (Nbr(n, k) ≠ 0) then (see Table 1.3)
  { k ← Nbr(n, k)
output k (next position)
```

Algorithm 1.6 `markov-discrete-pebble`. Discrete Markov-ch Monte Carlo algorithm for the pebble game.

pebble-throwing lady. In this way, we can make sense of the conce the probability of being in configuration a at iteration i, which we implicitly used, for example in eqn (1.11), during our derivation o detailed-balance condition. Let us use the 3×3 pebble game to s this point in more detail. The ensemble of all transition probabilitie tween sites can be represented in a matrix, the system's transfer m P:

$$P = \{p(a \to b)\} = \begin{bmatrix} p(1 \to 1) & p(2 \to 1) & p(3 \to 1) & \dots \\ p(1 \to 2) & p(2 \to 2) & p(3 \to 2) & \dots \\ p(1 \to 3) & p(2 \to 3) & p(3 \to 3) & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \quad (1$$

The normalization condition in eqn (1.10) (the pebble must go so where) implies that each column of the matrix in eqn (1.13) adds u one.

With the numbering scheme of Fig. 1.18, the transfer matrix is

$$\{p(a \to b)\} = \begin{bmatrix} \boxed{\frac{1}{2}} & \frac{1}{4} & \cdot & \frac{1}{4} & \cdot & \cdot & \cdot & \cdot & \cdot \\ \frac{1}{4} & \boxed{\frac{1}{4}} & \frac{1}{4} & \cdot & \frac{1}{4} & \cdot & \cdot & \cdot & \cdot \\ \cdot & \frac{1}{4} & \boxed{\frac{1}{2}} & \cdot & \cdot & \frac{1}{4} & \cdot & \cdot & \cdot \\ \frac{1}{4} & \cdot & \cdot & \boxed{\frac{1}{4}} & \frac{1}{4} & \cdot & \frac{1}{4} & \cdot & \cdot \\ \cdot & \frac{1}{4} & \cdot & \frac{1}{4} & \boxed{0} & \frac{1}{4} & \cdot & \frac{1}{4} & \cdot \\ \cdot & \cdot & \frac{1}{4} & \cdot & \frac{1}{4} & \boxed{\frac{1}{4}} & \cdot & \cdot & \frac{1}{4} \\ \cdot & \cdot & \cdot & \frac{1}{4} & \cdot & \cdot & \boxed{\frac{1}{2}} & \frac{1}{4} & \\ \cdot & \cdot & \cdot & \cdot & \frac{1}{4} & \cdot & \frac{1}{4} & \boxed{\frac{1}{4}} & \frac{1}{4} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \frac{1}{4} & \cdot & \frac{1}{4} & \boxed{\frac{1}{2}} \end{bmatrix}, \quad (1.$$

where the symbols "." stand for zeros. All simulations start at the cl house, site 9 in our numbering scheme. For the ensemble of Monte Ca simulations, this means that the probability vector at iteration $i = 0$

$$\{\pi^0(1), \dots, \pi^0(9)\} = \{0, \dots, 0, 1\}.$$

After one iteration of the Monte Carlo algorithm, the pebble is at the clubhouse with probability 1/2, and at positions 6 and 8 with probabilities 1/4. This is mirrored by the vector $\{\pi^{i=1}(1), \ldots, \pi^{i=1}(9)\}$ after one iteration, obtained by a matrix–vector multiplication

$$\pi^{i+1}(a) = \sum_{b=1}^{9} p(b \to a)\pi^i(b) \tag{1.15}$$

for $i = 0$, and $i + 1 = 1$. Equation (1.15) is easily programmed (see Alg. 1.7 (`transfer-matrix`); for the matrix in eqn (1.13), the eqn (1.15) corresponds to a matrix–vector multiplication, with the vector to the right). Repeated application of the transfer matrix to the initial probability vector allows us to follow explicitly the convergence of the Monte Carlo algorithm (see Table 1.4 and Fig. 1.19).

procedure `transfer-matrix`
input $\{p(a \to b)\}$ (matrix in eqn (1.14))
input $\{\pi^i(1), \ldots, \pi^i(9)\}$
for $a = 1, \ldots, 9$ **do**
{
  $\pi^{i+1}(a) \leftarrow 0$
  for $b = 1, \ldots, 9$ **do**
    { $\pi^{i+1}(a) \leftarrow \pi^{i+1}(a) + p(b \to a)\pi^i(b)$
output $\{\pi^{i+1}(1), \ldots, \pi^{i+1}(9)\}$

Algorithm 1.7 `transfer-matrix`. Computing pebble-game probabilities at iteration $i + 1$ from the probabilities at iteration i.

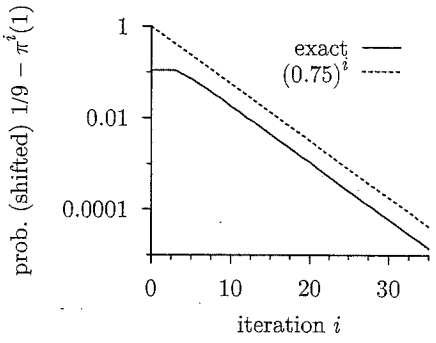

Fig. 1.19 Pebble-game probability of site 1, shifted by 1/9 (from Alg. 1.7 (`transfer-matrix`); see Table 1.4).

To fully understand convergence in the pebble game, we must analyze the eigenvectors $\{\boldsymbol{\pi}_e^1, \ldots, \boldsymbol{\pi}_e^9\}$ and the eigenvalues $\{\lambda_1, \ldots, \lambda_9\}$ of the transfer matrix. The eigenvectors $\boldsymbol{\pi}_e^k$ are those vectors that essentially reproduce under the application of P:

$$P\boldsymbol{\pi}_e^k = \lambda_k \boldsymbol{\pi}_e^k.$$

Table 1.4 Input/output of Alg. 1.7 (`transfer-matrix`), initially started at the clubhouse

Prob.	Iteration i				
	0	1	2	...	∞
$\pi^i(1)$	0	0	0	...	1/9
$\pi^i(2)$	0	0	0	...	1/9
$\pi^i(3)$	0	0	0.062	...	1/9
$\pi^i(4)$	0	0	0	...	1/9
$\pi^i(5)$	0	0	1/8	...	1/9
$\pi^i(6)$	0	1/4	0.188	...	1/9
$\pi^i(7)$	0	0	0.062	...	1/9
$\pi^i(8)$	0	1/4	0.188	...	1/9
$\pi^i(9)$	1	1/2	0.375	...	1/9

Writing a probability vector $\boldsymbol{\pi} = \{\pi(1), \ldots, \pi(9)\}$ in terms of the eigen vectors, i.e.

$$\boldsymbol{\pi} = \alpha_1 \boldsymbol{\pi}_e^1 + \alpha_2 \boldsymbol{\pi}_e^2 + \cdots + \alpha_9 \boldsymbol{\pi}_e^9 = \sum_{k=1}^{9} \alpha_k \boldsymbol{\pi}_e^k,$$

allows us to see how it is transformed after one iteration,

$$P\boldsymbol{\pi} = \alpha_1 P\boldsymbol{\pi}_e^1 + \alpha_2 P\boldsymbol{\pi}_e^2 + \cdots + \alpha_9 P\boldsymbol{\pi}_e^9 = \sum_{k=1}^{9} \alpha_k P\boldsymbol{\pi}_e^k$$

$$= \alpha_1\lambda_1\boldsymbol{\pi}_e^1 + \alpha_2\lambda_2\boldsymbol{\pi}_e^2 + \cdots + \alpha_9\lambda_9\boldsymbol{\pi}_e^9 = \sum_{k=1}^{9} \alpha_k\lambda_k\boldsymbol{\pi}_e^k,$$

or after i iterations,

$$P^i\boldsymbol{\pi} = \alpha_1\lambda_1^i\boldsymbol{\pi}_e^1 + \alpha_2\lambda_2^i\boldsymbol{\pi}_e^2 + \cdots + \alpha_9\lambda_9^i\boldsymbol{\pi}_e^9 = \sum_{k=1}^{9} \alpha_k(\lambda_k)^i\boldsymbol{\pi}_e^k.$$

Only one eigenvector has components that are all nonnegative, so tha it can be a vector of probabilities. This vector must have the larges eigenvalue λ_1 (the matrix P being positive). Because of eqn (1.10), w have $\lambda_1 = 1$. Other eigenvectors and eigenvalues can be computed ex plicitly, at least in the 3×3 pebble game. Besides the dominant eigen value λ_1, there are two eigenvalues equal to 0.75, one equal to 0.5, etc This allows us to follow the precise convergence towards the asymptoti equilibrium solution:

$$\{\pi^i(1), \ldots, \pi^i(9)\}$$
$$= \underbrace{\{\tfrac{1}{9}, \ldots, \tfrac{1}{9}\}}_{\substack{\text{first eigenvector}\\ \text{eigenvalue } \lambda_1 = 1}} + \alpha_2 \cdot (0.75)^i \underbrace{\{-0.21, \ldots, 0.21\}}_{\substack{\text{second eigenvector}\\ \text{eigenvalue } \lambda_2 = 0.75}} + \cdots .$$

In the limit $i \to \infty$, the contributions of the subdominant eigenvector disappear and the first eigenvector, the vector of stationary probabilitie in eqn (1.9), exactly reproduces under multiplication by the transfe matrix. The two are connected through the detailed-balance condition as discussed in simpler terms at the beginning of this subsection.

The difference between $\{\pi^i(1), \ldots, \pi^i(9)\}$ and the asymptotic solutio is determined by the second largest eigenvalue of the transfer matrix an is proportional to

$$(0.75)^i = \mathrm{e}^{i \cdot \log 0.75} = \exp\left(-\frac{i}{3.476}\right). \qquad (1.16$$

The data in Fig. 1.19 clearly show the $(0.75)^i$ behavior, which is equiv alent to an exponential $\propto \mathrm{e}^{-i/\Delta_i}$ where $\Delta_i = 3.476$. Δ_i is a timescale and allows us to define short times and long times: a short simulatio

has fewer than Δ_i iterations, and a long simulation has many more than that.

In conclusion, we see that transfer matrix iterations and Monte Carlo calculations reach equilibrium only after an infinite number of iterations. This is not a very serious restriction, because of the existence of a timescale for convergence, which is set by the second largest eigenvalue of the transfer matrix. To all intents and purposes, the asymptotic equilibrium solution is reached after the convergence time has passed a few times. For example, the pebble game converges to equilibrium after a few times 3.476 iterations (see eqn (1.16)). The concept of equilibrium is far-reaching, and the interest in Monte Carlo calculations is rightly strong because of this timescale, which separates fast and slow processes and leads to exponential convergence.

1.1.5 The Metropolis algorithm

In Subsection 1.1.4, direct inspection of the detailed-balance condition in eqn (1.12) allowed us to derive Markov-chain algorithms for simple games where the probability of each configuration was either zero or one. This is not the most general case, even for pebbles, which may be less likely to be at a position a on a hilltop than at another position b located in a valley (so that $\pi(a) < \pi(b)$). Moves between positions a and b with arbitrary probabilities $\pi(a)$ and $\pi(b)$, respecting the detailed-balance condition in eqn (1.12), are generated by the Metropolis algorithm (see Metropolis *et al.* (1953)), which accepts a move $a \to b$ with probability

$$p(a \to b) = \min\left[1, \frac{\pi(b)}{\pi(a)}\right]. \tag{1.17}$$

In the heliport game, we have unknowingly used eqn (1.17): for a and b both inside the square, the move was accepted without further tests ($\pi(b)/\pi(a) = 1$, $p(a \to b) = 1$). In contrast, for a inside but b outside the square, the move was rejected ($\pi(b)/\pi(a) = 0$, $p(a \to b) = 0$).

Table 1.5 Metropolis algorithm represented by eqn (1.17): detailed balance holds because the second and fourth rows of this table are equal

Case	$\pi(a) > \pi(b)$	$\pi(b) > \pi(a)$
$p(a \to b)$	$\pi(b)/\pi(a)$	1
$\pi(a)p(a \to b)$	$\pi(b)$	$\pi(a)$
$p(b \to a)$	1	$\pi(a)/\pi(b)$
$\pi(b)p(b \to a)$	$\pi(b)$	$\pi(a)$

To prove eqn (1.17) for general values of $\pi(a)$ and $\pi(b)$, one has only to write down the expressions for the acceptance probabilities $p(a \to b)$ and $p(b \to a)$ from eqn (1.17) for the two cases $\pi(a) > \pi(b)$ and $\pi(b) > \pi(a)$ (see Table 1.5). For $\pi(a) > \pi(b)$, one finds that $\pi(a)p(a \to b) =$

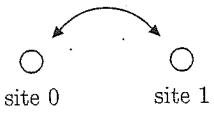

Fig. 1.20 Two-site problem. The probabilities to be at site 0 and site 1 are proportional to $\pi(0)$ and $\pi(1)$, respectively.

$\pi(b)p(b \to a) = \pi(b)$. In this case, and likewise for $\pi(b) > \pi(a)$, detaile balance is satisfied. This is all there is to the Metropolis algorithm.

Let us implement the Metropolis algorithm for a model with just tv sites: site 0, with probability $\pi(0)$, and site 1, with $\pi(1)$, probabiliti that we may choose to be arbitrary positive numbers (see Fig. 1.2(The pebble is to move between the sites such that, in the long ru the times spent on site 0 and on site 1 are proportional to $\pi(0)$ a $\pi(1)$, respectively. This is achieved by computing the ratio of statistic weights $\pi(1)/\pi(0)$ or $\pi(0)/\pi(1)$, and comparing it with a random numb `ran`$(0,1)$, a procedure used by almost all programs implementing t Metropolis algorithm (see Fig. 1.21 and Alg. 1.8 (`markov-two-site`).

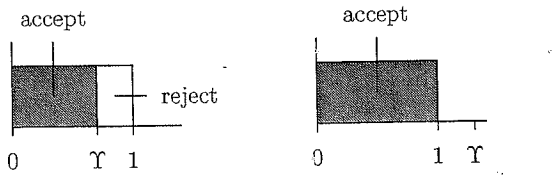

Fig. 1.21 Accepting a move with probability $\min(1, \Upsilon)$ with the help of a random number `ran`$(0,1)$.

We may run this program for a few billion iterations, using the outp of iteration i as the input of iteration $i+1$. While waiting for the outp we can also clean up Alg. 1.8 (`markov-two-site`) a bit, noticing tha $\Upsilon > 1$, its comparison with a random number between 0 and 1 makes sense: the move will certainly be accepted. For $\pi(l) > \pi(k)$, we sho thus work around the calculation of the quotient, the generation o random number and the comparison with that number.

procedure `markov-two-site`
input k (either 0 or 1)
if $(k = 0)$ $l \leftarrow 1$
if $(k = 1)$ $l \leftarrow 0$
$\Upsilon \leftarrow \pi(l)/\pi(k)$
if (`ran`$(0,1) < \Upsilon$) $k \leftarrow l$
output k (next site)

Algorithm 1.8 `markov-two-site`. Sampling sites 0 and 1 with station ary probabilities $\pi(0)$ and $\pi(1)$ by the Metropolis algorithm.

1.1.6 A priori probabilities, triangle algorithm

On the heliport, the moves Δ_x and Δ_y were restricted to a small squ of edge length 2δ, the throwing range, centered at the present posit (see Fig. 1.22(A)). This gives an example of an a priori probability tribution, denoted by $\mathcal{A}(a \to b)$, from which we sample the move a -

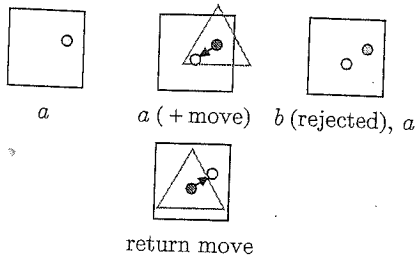

Fig. 1.23 Rejected move $a \to b$ in the triangle algorithm.

from a to b. (Alg. 7.3 (`direct-triangle`) allows us to sample a point inside an arbitrary triangle).

The triangle algorithm can be generalized to an arbitrary a pri ability $\mathcal{A}(a \to b)$, and the generalized Metropolis algorithm (eq will ensure that the detailed-balance condition remains satisfie ever, only good choices for $\mathcal{A}(a \to b)$ have an appreciable acc rate (the acceptance probability of each move averaged over all and actually move the chain forward. As a simple example, we ca of a configuration a with a high probability ($\pi(a)$ large), close figurations b with $\pi(b)$ small. The original Metropolis algorith to many rejections in this situation, slowing down the simulatior ducing a priori probabilities to propose configurations b less fre wastes less computer time with rejections. Numerous examples chapters illustrate this point.

A case worthy of special attention is $\mathcal{A}(a \to b) = \pi(b)$ and $a) = \pi(a)$, for which the acceptance rate in eqn (1.18) of the gene Metropolis algorithm is equal to unity: we are back to direct sa which we abandoned because we did not know how to put it intc However, no circular argument is involved. A priori probabilit crucial when we can almost do direct sampling, or when we can directly sample a subsystem. A priori probabilities then present th putational analogue of perturbation theory in theoretical physics

1.1.7 Perfect sampling with Markov chains

The difference between the ideal world of children (direct samplin that of adults (Markov-chain sampling) is clear-cut: in the former, access to the probability distribution $\pi(\mathbf{x})$ is possible, but in the convergence towards $\pi(\mathbf{x})$ is reached only in the long-time limit. trolling the error from within the simulation poses serious difficulti may have the impression that we have decorrelated from the clubl without suspecting that it is—figuratively speaking—still aroun corner. It has taken half a century to notice that this difficulty can s times be resolved, within the framework of Markov chains, by prod

perfect chain samples, which are equivalent to the children's throws and guaranteed to be totally decorrelated from the initial condition.

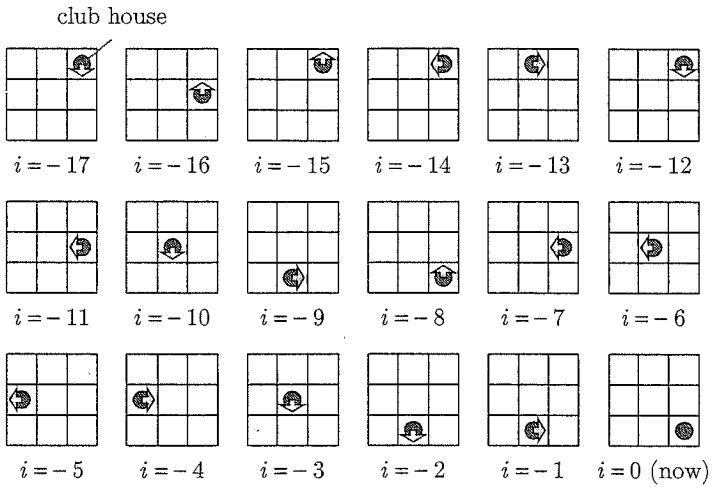

Fig. 1.24 A 3×3 pebble game starting at the clubhouse at iteration $i = -17$, arriving at the present configuration at $i = 0$ (now).

For concreteness, we discuss perfect sampling in the context of the 3×3 pebble game. In Fig. 1.24, the stone has moved in 17 steps from the clubhouse to the lower right corner. As the first subtle change in the setting, we let the simulation start at time $i = -17$, and lead up to the present time $i = 0$. Because we started at the clubhouse, the probability of being in the lower right corner at $i = 0$ is slightly smaller than $1/9$. This correlation goes to zero exponentially in the limit of long running times, as we have seen (see Fig. 1.19).

The second small change is to consider random maps rather than random moves (see Fig. 1.25: from the upper right corner, the pebble must move down; from the upper left corner, it must move right; etc.). At each iteration i, a new random map is drawn. Random maps give a consistent, alternative definition of the Markov-chain Monte Carlo method, and for any given trajectory it is impossible to tell whether it was produced by random maps or by a regular Monte Carlo algorithm (in Fig. 1.26, the trajectory obtained using random maps is the same as in Fig. 1.24).

In the random-map Markov chain of Fig. 1.26, it can, furthermore, be verified explicitly that any pebble position at time $i = -17$ leads to the lower right corner at iteration $i = 0$. In addition, we can imagine that $i = -17$ is not really the initial time, but that the simulation has been going on since $i = -\infty$. There have been random maps all along the way, and Fig. 1.26 shows only the last stretch. The pebble position at $i = 0$ is the same for any configuration at $i = -17$: it is also the outcome of an infinite simulation, with an initial position at $i = -\infty$, from which it has

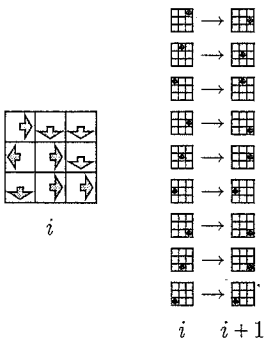

Fig. 1.25 A random map at iteration i and its action on all possible pebble positions.

decorrelated. The $i = 0$ pebble position in the lower right corner i
a direct sample—obtained by a Markov-chain Monte Carlo methc

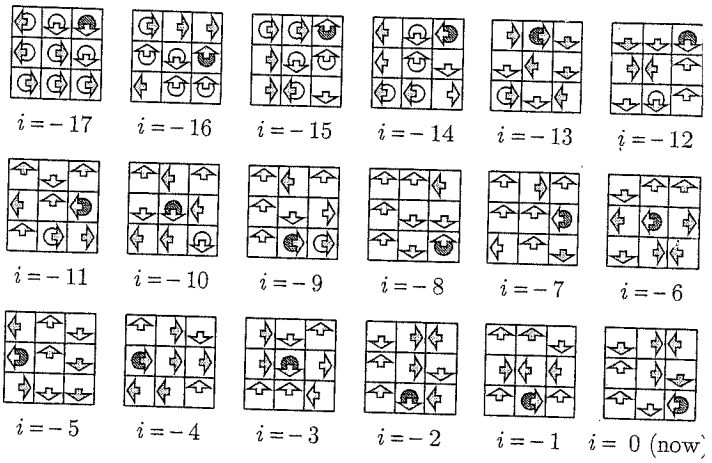

Fig. 1.26 Monte Carlo dynamics using time-dependent random ma
All positions at $i = -17$ give an identical output at $i = 0$.

The idea of perfect sampling is due to Propp and Wilson (19
It mixes a minimal conceptual extension of the Monte Carlo me
(random maps) with the insight that a finite amount of backtrac
(called coupling from the past) may be sufficient to figure out the pre
state of a Markov chain that has been running forever (see Fig. 1.2

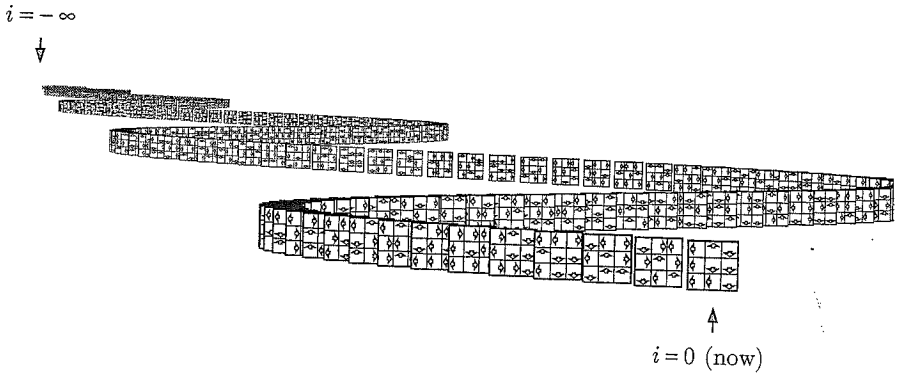

Fig. 1.27 A random-map Markov chain that has been running sinc
$i = -\infty$.

Producing direct samples for a 3×3 pebble game by Markov chain
a conceptual breakthrough, but not yet a great technical achieveme
Later on, in Chapter 5, we shall construct direct samples (using Mar
chains) with $2^{100} = 1\,267\,650\,600\,228\,229\,401\,496\,703\,205\,376$ configu
tions. Going through all of them to see whether they have merged