Derandomized parallel simulation of Markov chains via P-machines

Jim Propp
(U. Mass. Lowell)

July 11, 2008

(based on articles I'm writing with Ander Holroyd and Lionel Levine, as well as articles I didn't write authored by Joshua Cooper, Benjamin Doerr, Tobias Friedrich, Joel Spencer, and Gábor Tardos; with thanks also to Stephen Witham)

Slides for this talk are on-line at
`jamespropp.org/mcqmc08-2.pdf`

I. Deriving the rotor mechanism from first principles

Question: If a finite-state irreducible aperiodic Markov chain is started in state x_0, what is the probability p that it hits state v before hitting state w? (With probability 1, eventually the chain will hit one or the other.)

Parallel simulation (random or deterministic): Put N particles at state x_0. Advance them until each has hit v or w. Estimate p by K/N where K is the number of particles that "escaped" (i.e., that hit v).

If we do random independent simulation, the expected magnitude of the discrepancy $|K/N - p|$ is $O(1/\sqrt{N})$.

That is, the un-rescaled, signed discrepancy $D := K - Np$ is $O(\sqrt{N})$.

What sort of (random or deterministic) simulation might give us smaller discrepancy?

Idea: Write the "global discrepancy" D as a sum of "local discrepancies" and choose a mechanism that will control the local discrepancies individually.

In our case, let $h(x)$ be the probability that a random walk started at x escapes (i.e. hits v before hitting w), so that $h(x_0) = p$, $h(v) = 1$, and $h(w) = 0$.

The function h is *harmonic* outside of $\{v, w\}$; that is, for all x not in $\{v, w\}$, $h(x) = \sum_y p(x, y)h(y)$.

We can use the harmonicity property of $h(\cdot)$ to write the global discrepancy D as a sum of local discrepancies $D(x)$ with x varying over all states other than v and w.

Specifically, we have

$$D = \sum_{x \notin \{v,w\}} D(x)$$

with $D(x)$ equal to

$$\sum_y (N(x,y) - N(x)p(x,y))(h(y) - h(x))$$

where $N(x,y)$ is the number of particles that moved from x to y (note: a particle that moved from x to y exactly k times contributes k to $N(x,y)$) and $N(x) = \sum_y N(x,y)$.

We can't control $h(y)-h(x)$, but we can choose to "simulate" the Markov chain with steps that minimize the magnitude of the multipliers $N(x,y)-N(x)p(x,y)$.

Example: Suppose that at time 0 there are 3 particles at x and at time 1 there are 5 particles at x. Suppose $p(x,y) = p(x,z) = 1/2$. We can't split the 3 particles at x at time 0 evenly between y and z, nor can we split the 5 particles at x at time 1 evenly between y and z, but we can split the $3+5=8$ particles at x evenly between y and z over time.

To do this, we must ensure that if y gets an "extra" half-particle in the first step, z must get an extra half-particle in the second step, and vice versa.

This is exactly what a rotor-router does: It keeps track of imbalances in allocations and rectifies them over time to keep the local discrepancy as small as possible.

Holroyd-Propp: We can use deterministic rotor-routing of N particles to estimate the escape probability p to within an error of C/N, where C is

$$\sum_{x,y:\ p(x,y)\neq 0} |h(y) - h(x)|.$$

In the case where every $p(x, y)$ is either 0 or $1/m$ for some fixed m, the rotor-router mechanism is simple:

For each x, match up the y's with $p(x, y) > 0$ with the integers mod m.

At each time-step in the simulation, there is a rotor at each state x whose value is in $\mathbf{Z}$ mod m.

If there is just one particle at x at time t, we increment the rotor at x and send the particle to the ith neighbor of x, where i is the new value of the rotor.

If there are k particles at x at time t, we do the above procedure k times.

To find the state of the system at time $t + 1$, we do all of the foregoing for every x such that there are one or more particles at x at time t.

The same rotor-router mechanism also gives us ways to estimate other quantities associated with finite-state Markov chains with precision $O(1/N)$, such as the steady-state probability of a state, or the expected time to get from one state to another.

More complicated variants of the rotor-router mechanism can be used to compute or estimate other quantities, such as the expected squared time to get from one state to another.

II. Linear machines and P-machines

Let π (the "source") be a probability distribution on the state-space of a Markov chain, let A (the "target") be a subset of the state-space, and let T be a positive integer. We can use rotor-routers to estimate the probability p that a particle that starts at a π-random site will be in the set A after T time-steps:

For some large N, put $\pi(x)N \pm 1$ particles at each site x and run them via rotor-router dynamics for T steps. Then p can be estimated by K/N where K is the number of particles that ended up in the target set A after T steps of rotor-router walk.

This is the $\underline{P\text{-machine}}$ for estimating p.

If we could split the particles at x and send them to the successor states y in the exact proportions dictated by the transition probabilities $p(x, y)$, then p would exactly equal $\tilde{K}/N$, where $\tilde{K}$ is the "number of particles" in A at time T counted with fractional multiplicity.

In this case, we might as well take N to be 1 and use the term "mass" instead of "number of particles". Then the mass at time 0 is distributed according to π, and the mass at y at time $t + 1$ equals the sum, over all x, of $p(x, y)$ times the mass at x at time t (the "heat-flow rule").

This is the linear machine for computing p (p is just the mass in A at time T).

What's surprising here is that the P-machine is such a good approximation to the linear machine, at least for certain Markov chains.

Theorem (Cooper and Spencer):
Suppose our Markov chain is random walk in the d-dimensional lattice, our source distribution is concentrated on $\{(x_1, x_2, \ldots, x_d) \in \mathbf{Z}^d : x_1 + x_2 + ... + x_d \text{ is even}\}$, and our target set contains just one point.
Then the difference between K/N and p is at most C/N, where the constant C depends only on d, and not on π or A or T.

That is, the number of particles in A at time T (under rotor-router walk) differs from the *expected* number of particles in A at time T (under random walk) by at most a constant C that does not depend on the size of N (the number of particles being sent through the P-machine), or the size of T (the number of time-steps we are simulating), or on how far apart the source-distribution π and the target-set A are; all C depends on is the dimension of the lattice.

Furthermore, for small d, the constant C is fairly small. E.g., for $d = 1$ we can take $C = 3$, and for $d = 2$ we can take $C = 8$.

If the set A is larger, the error $|K/N - p|$ can be larger than C/N, but if the sites in A are largely contiguous, the error will probably be substantially smaller than the trivial upper bound $|A|C/N$.

An initial result in this direction (proved by Cooper, Doerr, Spencer, and Tardos) is that if $d = 1$ and A is an interval of length L, then $|K/N - p|$ is at most $O((\log L)/N)$.

III. Variants

Order matters: Doerr and Friedrich have done simulations showing that for $\mathbf{Z}^2$, the P-machine approximates the linear machine more closely if one uses rotors that rotate through the states in a non-cyclic order like

North, East, West, South, North, ...

instead of rotors that rotate through the states in a cyclic order like

North, East, South, West, North, ...

This ranking of the different RR schemes is consistent with Doerr and Friedrich's findings for cyclic vs. non-cyclic rotors in derandomized IDLA in two dimensions (as described in my other talk).

Cooper, Friedrich, Spencer, and Tardos have also studied P-machines on regular trees. They show that in that context, the discrepancy is $O(\sqrt{\log N}/N)$.

IV. Rounding

The single-particle rotor-walk algorithm, parallelized, becomes the P-machine algorithm. The latter can be viewed as a form of heat flow in fixed precision arithmetic, with a twist: the rotors control the rounding of the least significant bits.

Rotors actually give an improvement over naive methods of simulating heat flow in discrete space and discrete time with fixed-point arithmetic. (The method might generalize to variants of diffusion that include convection and reaction terms. But this will probably be of only minor interest for PDE, since rounding error isn't as big an issue as error introduced by discretization of space and time.)

V. Diffusion

The rotor-router model was invented at least three times: once by physicists (studying self-organized criticality), once by computer scientists (studying load-balancing networks), and once by mathematicians (seeking deterministic analogues of stochastic processes).

The physicists who studied rotor-routing (Priezzhev, Dhar, Dhar, and Krishnamurthy) called it the *Eulerian walkers model*.

The physicists found that when a single particle does rotor-routing in a large rectangular array with random rotor-settings at the sites, repeatedly walking from a randomly chosen site to the boundary of the rectangle, the rotors evolve into an organized state, and the distance travelled by the particle in T consecutive steps tends to be on the order of $T^{1/3}$ rather than $T^{1/2}$.

This tells us that although rotor-router walk behaves like diffusion for *some* purposes, it does not behave like diffusion for *all* purposes. This is a basic fact of life for the kind of quasirandomness you get from rotor-routers and similar mechanisms: they are highly tailored to estimating a *particular quantity* for a *particular stochastic process*, and if you keep the process the same but change the question you're asking, a different kind of simulation may be needed.

For pseudorandom simulation, when you generate a sample path you are essentially studying all possible questions you might ever want to ask about the system. The trouble is, you get no guarantees that your answers are any good.

Rotor-router simulations, when applicable, often come with rigorous, non-asymptotic bounds that sometimes let you replace confidence intervals for quantities of interest by certainty intervals.