Rigorous analysis of Markov chain Monte Carlo algorithms (using the permanent as a case study)

Mark Jerrum
(joint work with Alistair Sinclair, U.C. Berkeley,
and Eric Vigoda, Georgia Tech)

Mathematical Institute, Warwick, 29th September 2008

The permanent

Goal: Given a matrix $A = (a_{ij}) \in \{0,1\}^{n \times n}$, compute

$$\operatorname{per} A := \sum_{\sigma} \prod_{i} a_{i,\sigma(i)},$$

where σ ranges over permutations of $\{1, \ldots, n\}$.

Alternative view: Given a bipartite graph G with $n + n$ vertices, compute the number of perfect matchings in G.

Connection: View A as the adjacency matrix of G.

Requirement: Want to do this efficiently, and certainly in time polynomial in n.

Known results: exact computation

- Poly-time algorithm for any *planar* graph [Kasteleyn, 63].
- #P-complete for general bipartite graphs [Valiant, 79].

Known results: approximate computation

- Markov chain Monte Carlo method proposed [Broder, 86].
- Rigorous analysis of Broder's proposal leads to a polynomial-time approximation algorithm (with arbitrarily small relative error) for a certain class of $0,1$-matrices [J. & Sinclair, 89]. This class includes almost all $0,1$-matrices, and all sufficiently dense matrices.
- Polynomial-time algorithm with c^n approximation factor for any $0,1$-matrix ($c \approx 1.31$) [Barvinok, 99].

New(ish) result

Given an $n \times n$ matrix A and real parameters $\epsilon, \delta > 0$, a *fully-polynomial randomized approximation scheme* (FPRAS) for $\operatorname{per} A$ is required to compute a number Z (a r.v.) such that:

$$\Pr\left[(1-\epsilon)\operatorname{per} A \leq Z \leq (1+\epsilon)\operatorname{per} A\right] \geq 1-\delta$$

in time polynomial in n, ϵ^{-1}, and $\log(\delta^{-1})$.

Theorem (J., Sinclair & Vigoda, 00)

There exists an FPRAS for the permanent of an arbitrary $0,1$*-matrix.*

The result extends to arbitrary non-negative real matrices A.

Sampling vs counting

For convenience, adopt the view of counting perfect matchings in a bipartite graph.

Fact

In order to count perfect matchings approximately it is sufficient to sample perfect matchings (almost) uniformly at random.

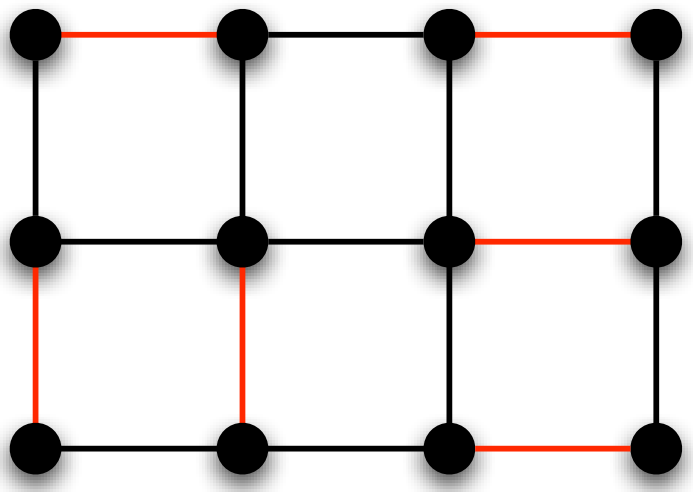
The Markov chain Monte Carlo method

Design an ergodic Markov chain (MC) whose state space Ω is the set all perfect matchings in G, and whose stationary distribution is uniform. Then simulate the MC for "sufficiently many" steps and return the final state.
In fact, to permit free movement about the state space, it is convenient to augment the state space with "near-perfect" matchings (leaving just two vertices uncovered). Thus

$$\Omega := \mathcal{M} \cup \bigcup_{u,v} \mathcal{M}(u,v),$$

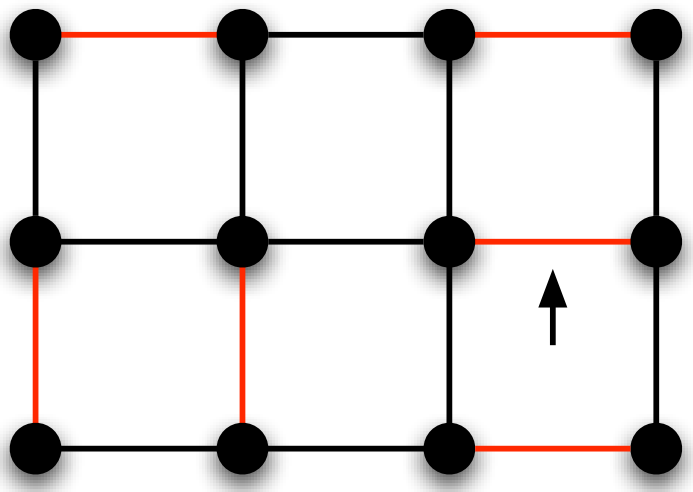
where $\mathcal{M}$ is the set of all perfect matchings, and $\mathcal{M}(u,v)$ is the set of matchings leaving just vertices u and v uncovered.

Transitions from a perfect matching



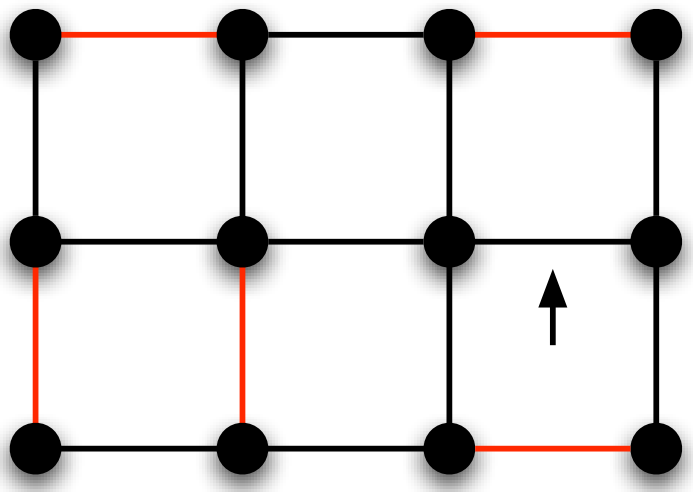
[Broder, 86; J. & Sinclair, 88]

Transitions from a perfect matching



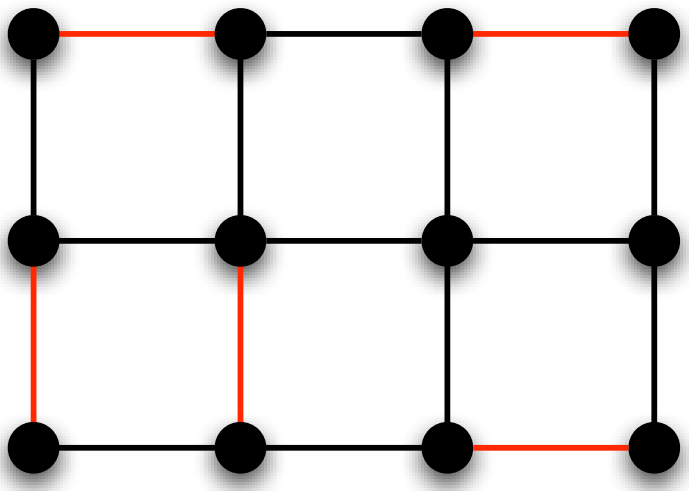
[Broder, 86; J. & Sinclair, 88]

Transitions from a perfect matching

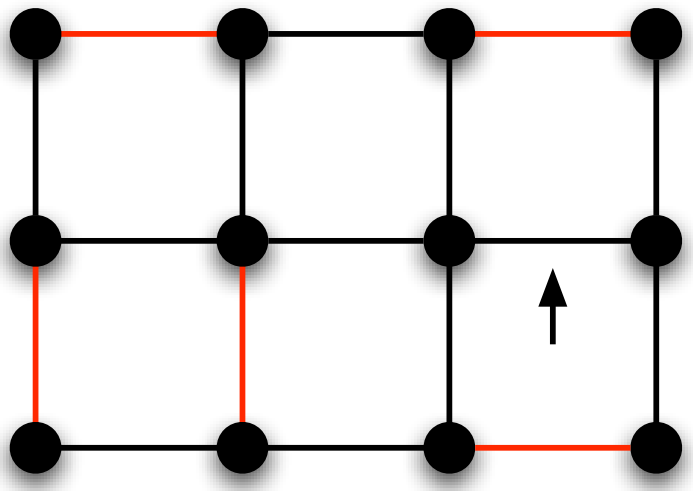


[Broder, 86; J. & Sinclair, 88]

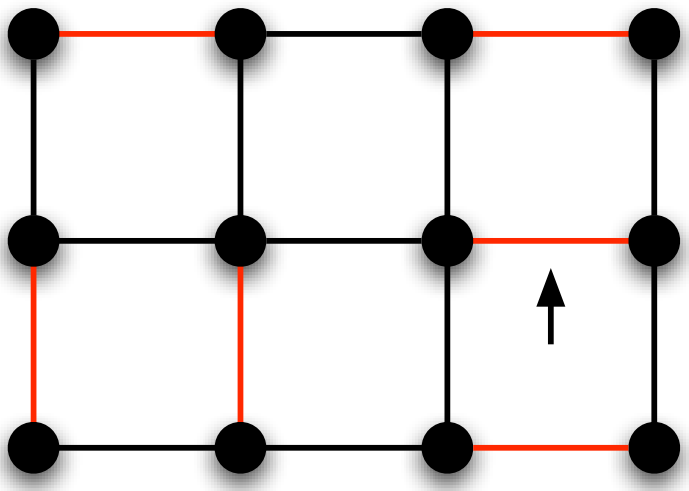
Transitions from a near-perfect matching



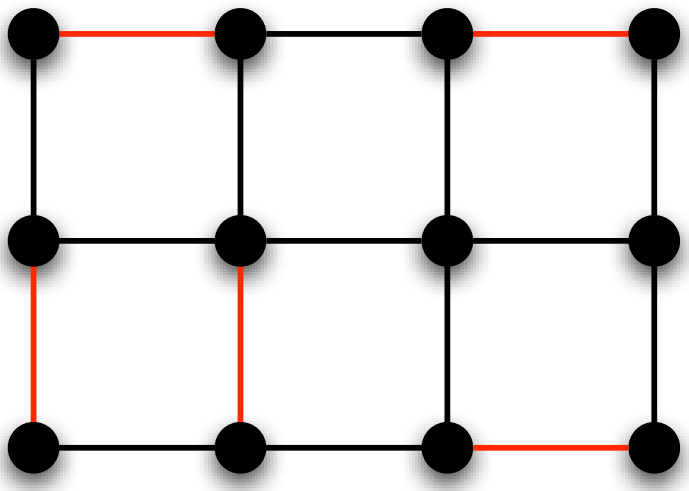
Transitions from a near-perfect matching



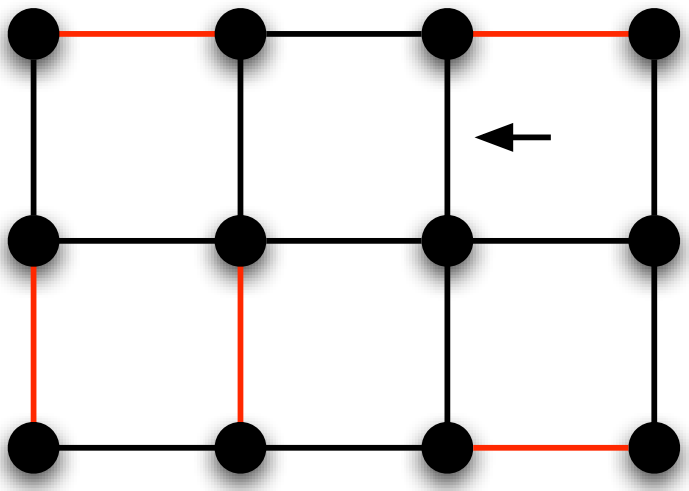
Transitions from a near-perfect matching



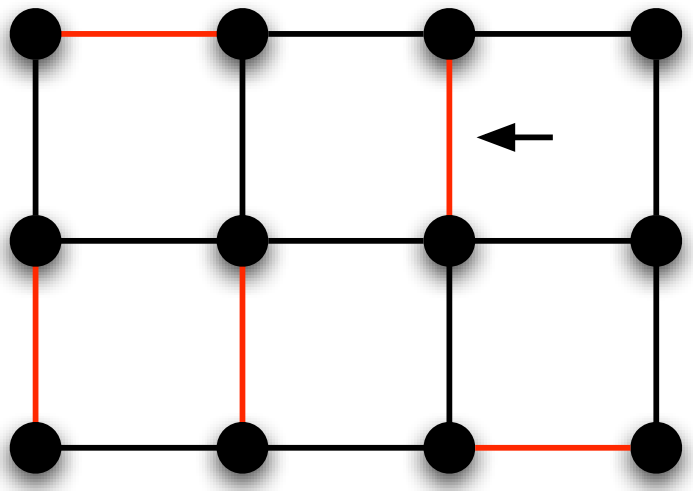
Transitions from a near-perfect matching



Transitions from a near-perfect matching



Transitions from a near-perfect matching



How to bound the "mixing time"

- Transition probabilities $P : \Omega^2 \to \mathbb{R}$;
- Stationary distribution $\pi : \Omega \to [0, 1]$;
- Test function $f : \Omega \to \mathbb{R}$;
- Dirichlet form (measure of local variation): $\mathcal{E}(f, f) := \frac{1}{2} \sum_{x,y} \pi(x) P(x, y)(f(x) - f(y))^2$;
- Variance (measure of global variation): $\operatorname{Var} f := \frac{1}{2} \sum_{x,y} \pi(x)\pi(y)(f(x) - f(y))^2$;
- Poincaré inequality: $\mathcal{E}(f, f) \geq \lambda \operatorname{Var} f$, for all f.

Fact

$\operatorname{Var}(Pf) \leq (1 - \lambda) \operatorname{Var} f.$

Here, $[Pf](x) := \sum_y P(x, y) f(y)$.

"Canonical paths"

For every pair of states $x, y \in \Omega$, define a *canonical path* γ_{xy} from x to y using valid transitions of the MC.
"Congestion constant" ϱ:

$$\sum_{\gamma_{xy} \ni (z,z')} \pi(x)\pi(y)\,|\gamma_{xy}| \leq \varrho\,\pi(z)P(z,z'), \quad \forall z, z'.$$

Fact (Diaconis & Stroock, 91; Sinclair, 92.)

$$\lambda \geq \frac{1}{\varrho}.$$

Thus:

$$\text{Low congestion } \varrho \Rightarrow \text{large Poincaré constant } \lambda$$
$$\Rightarrow \text{rapid mixing.}$$

Intuition

$$\begin{aligned}\operatorname{Var} f \text{ is large} &\Rightarrow \text{many paths have some transition } (x,y)\\ &\qquad\qquad \text{with } (f(x)-f(y))^2 \text{ large}\\ &\Rightarrow \text{many transitions } (x,y)\\ &\qquad\qquad \text{have } (f(x)-f(y))^2 \text{ large}\\ &\Rightarrow \mathcal{E}(f,f) \text{ is large.}\end{aligned}$$

NB. Need the following conditions, which are a consequence of small congestion ϱ:

- canonical paths are short;
- no transition supports a disproportionate number of canonical paths.

Back to the MC on matchings

Algorithm for sampling a perfect matching:

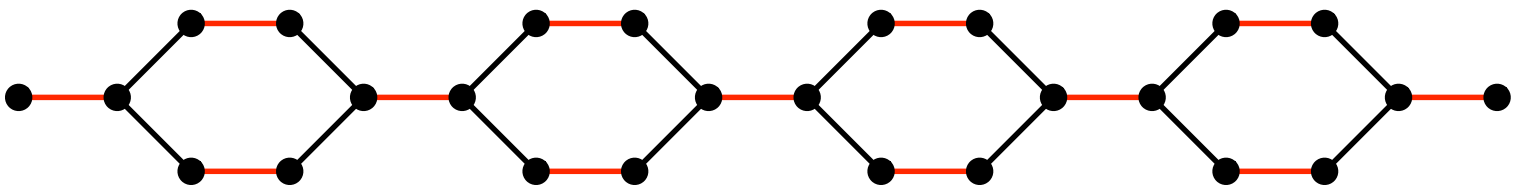
1. Start at an arbitrary matching in Ω.
2. Simulate the MC on Ω as defined above.
3. After "sufficiently many" steps, stop the simulation and let $M \in \Omega$ be the current state.
4. If M is a perfect matching, output it, otherwise repeat the algorithm.

Theorem (J. & Sinclair, 1988)

The above algorithm runs in polynomial time, provided

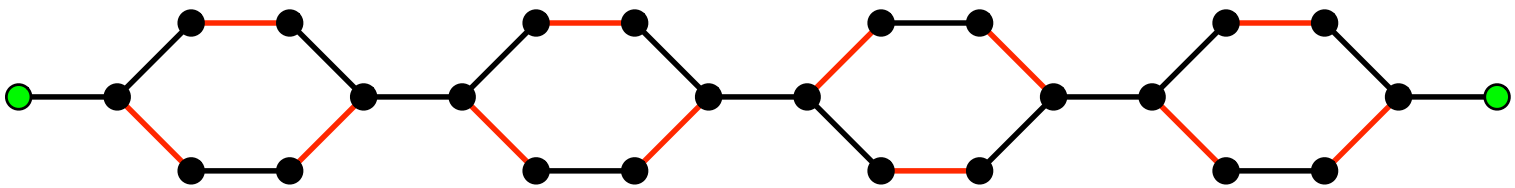
$$\frac{|\mathcal{M}|}{|\Omega|} \geq \frac{1}{\mathrm{poly}(n)}.$$

Key counterexample: problem 1



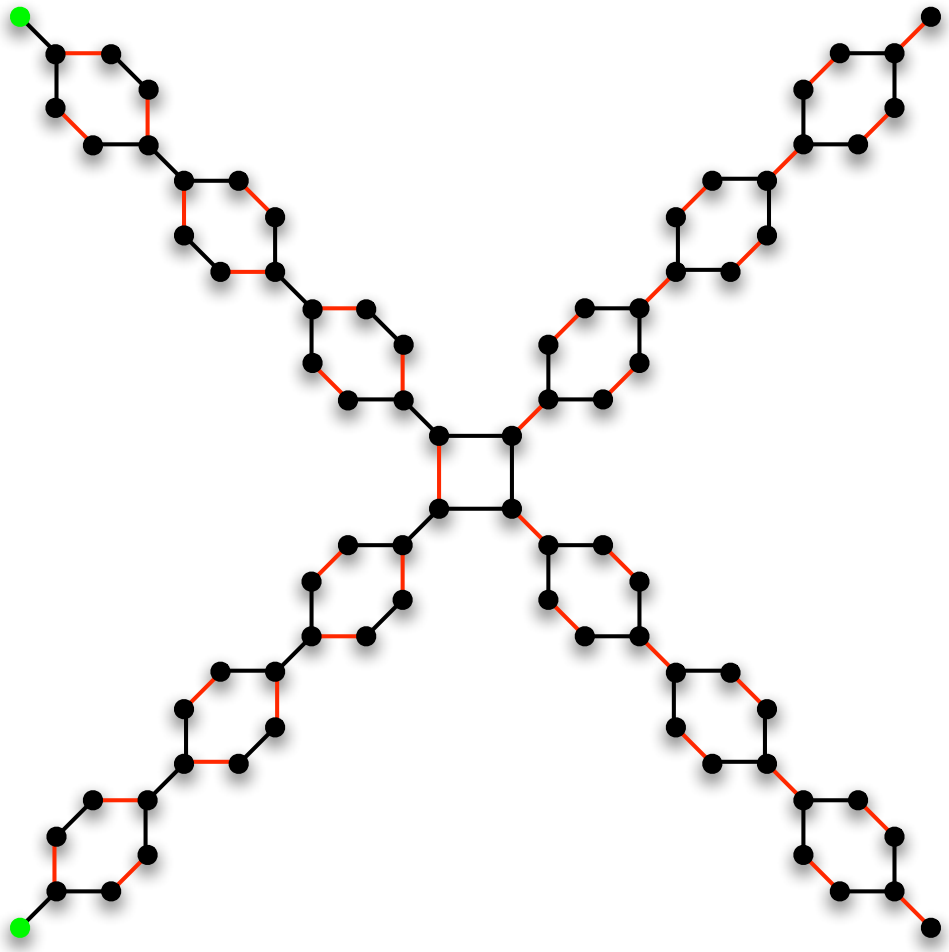
One perfect matching...

Key counterexample: problem 1



2^k near-perfect matchings (k is the number of hexagons)

Key counterexample: problem 2



The fix

Introduce weights on matchings in order to guarantee a weighted analogue of

$$\frac{|\mathcal{M}|}{|\Omega|} \geq \frac{1}{\mathrm{poly}(n)}.$$

The weight of a matching is a function of which vertices are un-matched, i.e., its hole pattern.
For a graph $G = (V_1, V_2, E)$, introduce weights $w : V_1 \times V_2 \to \mathbb{R}^+$, and (provisionally) define

$$\Lambda(M) := \begin{cases} w(u,v) & \text{if } M \in \mathcal{M}(u,v) \\ 1 & \text{if } M \in \mathcal{M}. \end{cases}$$

Modify the transition probabilities of the MC so that $\pi(M) \propto \Lambda(M)$.

Ideal weights

Ideally, we would like to set $w = w^*$, where w^* is the "ideal" weighting given by

$$w^*(u, v) := |\mathcal{M}|/|\mathcal{M}(u, v)|.$$

With this weighting, $\pi(\mathcal{M}(u, v))$ would be independent of the hole pair u, v, and equal to $\pi(\mathcal{M})$.
In other words, in the stationary distribution, all hole patterns (including no holes) would be equally likely.
Note that $\pi(\mathcal{M}) = 1/(n^2 + 1)$, so we have avoided first of our problems!

Ideal weights (continued)

It turns out that setting $w = w^*$ also ensures rapid mixing, by the a similar canonical path argument to that used in the unweighted case (though the analysis is now a little more involved). In fact, we do not need w to be exactly w^*.

Lemma

The MC on weighted matchings, where the weights $w(u, v)$ satisfy:

$$\frac{w^*(u, v)}{2} \leq w(u, v) \leq 2w^*(u, v),$$

has mixing time polynomial in n.

So we have avoided the second of our problems... except that we don't know the ideal weights w^*!

Converging to the ideal weights

We learn the weights $w^*(u, v)$ slowly in an iterative manner. Starting with the complete bipartite graph $K_{n,n}$ where we know $w^*(u, v)$ exactly, we "slowly" remove edges while revising the weights w.
In order to "slowly" remove edges, introduce an activity for each pair of vertices:

$$\lambda : V_1 \times V_2 \to (0, 1].$$

and define

$$\lambda(M) := \prod_{e \in M} \lambda(e),$$

and

$$\Lambda(M) := \begin{cases} w(u, v)\lambda(M) & \text{if } M \in \mathcal{M}(u, v); \\ \lambda(M) & \text{if } M \in \mathcal{M}. \end{cases}$$

Converging to the ideal weights (continued)

Starting with $\lambda = 1$ (constant function) we slowly decrease the activities until

$$\lambda(e) = \begin{cases} 1 & \text{if } e \in E(G); \\ \xi & \text{otherwise,} \end{cases}$$

where $\xi > 0$ is sufficiently small, say $\xi = 1/n!$.

Re-define $w^*(u, v) := \dfrac{\lambda(\mathcal{M})}{\lambda(\mathcal{M}(u, v))}$.

Every time we reduce an activity, we recompute an approximation to to w^*; this we can do by simulating the MC on weighted matchings. Provided we don't change the activities by too much at each step, the MC will remain rapidly mixing.

Open questions

- The canonical path argument (that we skipped) relies crucially on the graph G being bipartite. What is the computational complexity of approximating the number of perfect matchings in a general graph?
- Is there a practical algorithm for sampling perfect matchings (equivalently, estimating the permanent)? Our analysis gives a running time of $\widetilde{O}(n^{10})$! (Current world record is $\widetilde{O}(n^7)$ [Bezáková, Štefankovič, Vazirani & Vigoda, 06].)
- Is there a polynomial-time algorithm for sampling contingency tables?