Approximating the Permanent

Based on work by Broder, Jerrum, Sinclair, Vigoda,...

Andrew Lyons

Department of Computer Science, Dartmouth College
lyonsam@gmail.com

Math 100: Markov Chain Monte Carlo
Department of Mathematics, Dartmouth College
February 15, 2011

How hard is it to marry at random?
(On the approximation of the permanent)

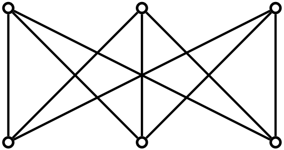
ANDREI Z. BRODER
DEC – Systems Research Center
130 Lytton Avenue
Palo Alto, CA. 94301

Extended abstract

Permanents and Perfect Matchings

$$\mathrm{Perm}(A) = \sum_{\sigma} \prod_{i=1}^{n} a_{i\sigma(i)}$$

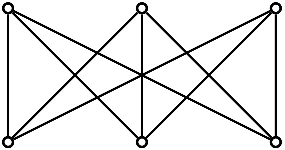
$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$



Permanents and Perfect Matchings

$$\mathrm{Perm}(A) = \sum_{\sigma} \prod_{i=1}^{n} a_{i\sigma(i)}$$

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

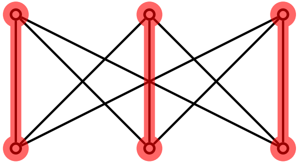


$$\mathrm{Perm}(A) =$$

Permanents and Perfect Matchings

$$\mathrm{Perm}(A) = \sum_{\sigma} \prod_{i=1}^{n} a_{i\sigma(i)}$$

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

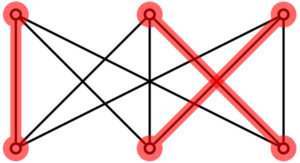


$$\mathrm{Perm}(A) = a_{11} a_{22} a_{33}$$

Permanents and Perfect Matchings

$$\mathrm{Perm}(A) = \sum_{\sigma} \prod_{i=1}^{n} a_{i\sigma(i)}$$

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

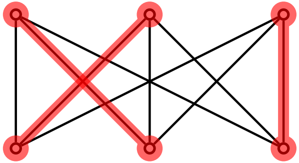


$$\mathrm{Perm}(A) = a_{11}a_{22}a_{33} + a_{11}a_{23}a_{32}$$

Permanents and Perfect Matchings

$$\mathrm{Perm}(A) = \sum_{\sigma} \prod_{i=1}^{n} a_{i\sigma(i)}$$

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

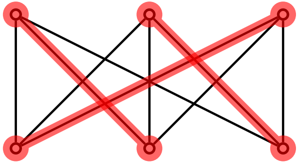


$$\mathrm{Perm}(A) = a_{11}a_{22}a_{33} + a_{11}a_{23}a_{32} + a_{12}a_{21}a_{33}$$

Permanents and Perfect Matchings

$$\mathrm{Perm}(A) = \sum_{\sigma} \prod_{i=1}^{n} a_{i\sigma(i)}$$

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

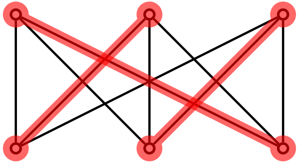


$$\mathrm{Perm}(A) = a_{11}a_{22}a_{33} + a_{11}a_{23}a_{32} + a_{12}a_{21}a_{33} + a_{12}a_{23}a_{31}$$

Permanents and Perfect Matchings

$$\mathrm{Perm}(A) = \sum_{\sigma} \prod_{i=1}^{n} a_{i\sigma(i)}$$

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

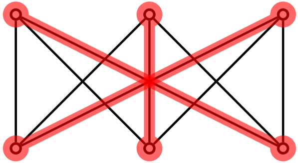


$$\mathrm{Perm}(A) = a_{11}a_{22}a_{33} + a_{11}a_{23}a_{32} + a_{12}a_{21}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32}$$

Permanents and Perfect Matchings

$$\mathrm{Perm}(A) = \sum_{\sigma} \prod_{i=1}^{n} a_{i\sigma(i)}$$

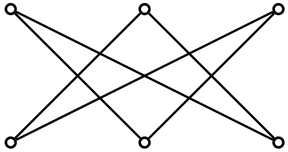
$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

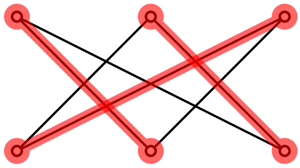


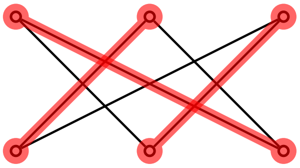
$$\begin{aligned} \mathrm{Perm}(A) = {} & a_{11}a_{22}a_{33} + a_{11}a_{23}a_{32} + a_{12}a_{21}a_{33} \\ & + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} + a_{13}a_{22}a_{31} \end{aligned}$$

Permanents and Perfect Matchings

$$A = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$







Canadian Journal of Mathematics, 17: 449-467, 1965

PATHS, TREES, AND FLOWERS

JACK EDMONDS

1. Introduction. A *graph G* for purposes here is a finite set of elements called *vertices* and a finite set of elements called *edges* such that each edge *meets* exactly two vertices, called the *end-points* of the edge. An edge is said to *join* its end-points.

A *matching* in G is a subset of its edges such that no two meet the same vertex. We describe an efficient algorithm for finding in a given graph a matching of maximum cardinality. This problem was posed and partly solved by C. Berge; see Sections 3.7 and 3.8.

Maximum matching is an aspect of a topic, treated in books on graph theory, which has developed during the last 75 years through the work of about a dozen authors. In particular, W. T. Tutte **(8)** characterized graphs which do not contain a *perfect* matching, or *1-factor* as he calls it—that is a set of edges with exactly one member meeting each vertex. His theorem prompted attempts at finding an efficient construction for perfect matchings.

This and our two subsequent papers will be closely related to other work on the topic. Most of the known theorems follow nicely from our treatment, though for the most part they are not treated explicitly. Our treatment is independent and so no background reading is necessary.

Section 2 is a philosophical digression on the meaning of "efficient algorithm." Section 3 discusses ideas of Berge, Norman, and Rabin with a new proof of Berge's theorem. Section 4 presents the bulk of the matching algorithm. Section 7 discusses some refinements of it.

There is an extensive combinatorial-linear theory related on the one hand to matchings in bipartite graphs and on the other hand to linear programming. It is surveyed, from different viewpoints, by Ford and Fulkerson in **(5)** and by A. J. Hoffman in **(6)**. They mention the problem of extending this relationship to non-bipartite graphs. Section 5 does this, or at least begins to do it. There, the König theorem is generalized to a matching-duality theorem for arbitrary graphs. This theorem immediately suggests a polyhedron which in a subsequent paper **(4)** is shown to be the convex hull of the vectors associated with the matchings in a graph.

Maximum matching in non-bipartite graphs is at present unusual among combinatorial extremum problems in that it is very tractable and yet not of the "unimodular" type described in **(5** and **6)**.

Received November 22, 1963. Supported by the O.N.R. Logistics Project at Princeton University and the A.R.O.D. Combinatorial Mathematics Project at N.B.S.

Finding a perfect matching is easy

Canadian Journal of Mathematics, 17: 449-467, 1965

PATHS, TREES, AND FLOWERS

JACK EDMONDS

1. Introduction. A *graph G* for purposes here is a finite set of elements called *vertices* and a finite set of elements called *edges* such that each edge *meets* exactly two vertices, called the *end-points* of the edge. An edge is said to *join* its end-points.

A *matching* in *G* is a subset of its edges such that no two meet the same vertex. We describe an efficient algorithm for finding in a given graph a matching of maximum cardinality. This problem was posed and partly solved by C. Berge; see Sections 3.7 and 3.8.

Maximum matching is an aspect of a topic, treated in books on graph theory, which has developed during the last 75 years through the work of about a dozen authors. In particular, W. T. Tutte **(8)** characterized graphs which do not contain a *perfect* matching, or *1-factor* as he calls it—that is a set of edges with exactly one member meeting each vertex. His theorem prompted attempts at finding an efficient construction for perfect matchings.

This and our two subsequent papers will be closely related to other work on the topic. Most of the known theorems follow nicely from our treatment, though for the most part they are not treated explicitly. Our treatment is independent and so no background reading is necessary.

Section 2 is a philosophical digression on the meaning of "efficient algorithm." Section 3 discusses ideas of Berge, Norman, and Rabin with a new proof of Berge's theorem. Section 4 presents the bulk of the matching algorithm. Section 7 discusses some refinements of it.

There is an extensive combinatorial-linear theory related on the one hand to matchings in bipartite graphs and on the other hand to linear programming. It is surveyed, from different viewpoints, by Ford and Fulkerson in **(5)** and by A. J. Hoffman in **(6)**. They mention the problem of extending this relationship to non-bipartite graphs. Section 5 does this, or at least begins to do it. There, the König theorem is generalized to a matching-duality theorem for arbitrary graphs. This theorem immediately suggests a polyhedron which in a subsequent paper **(4)** is shown to be the convex hull of the vectors associated with the matchings in a graph.

Maximum matching in non-bipartite graphs is at present unusual among combinatorial extremum problems in that it is very tractable and yet not of the "unimodular" type described in **(5** and **6)**.

Received November 22, 1963. Supported by the O.N.R. Logistics Project at Princeton University and the A.R.O.D. Combinatorial Mathematics Project at N.B.S.

Theoretical Computer Science 8 (1979) 189–201.

THE COMPLEXITY OF COMPUTING THE PERMANENT

L.G. VALIANT
Computer Science Department, University of Edinburgh, Edinburgh EH9 3JZ, Scotland

Communicated by M.S. Paterson
Received October 1977

Abstract. It is shown that the permanent function of (0, 1)-matrices is a complete problem for the class of counting problems associated with nondeterministic polynomial time computations. Related counting problems are also considered. The reductions used are characterized by their nontrivial use of arithmetic.

1. Introduction

Let A be an $n \times n$ matrix. The *permanent* of A is defined as

$$\text{Perm } A = \sum_{\sigma} \prod_{i=1}^{n} A_{i,\sigma(i)}$$

where the summation is over the $n!$ permutations of $(1, 2, \ldots, n)$. It is the same as the *determinant* except that all the terms have positive sign. Despite this similarity, while there are efficient algorithms for computing the determinant all known methods for evaluating the permanent take exponential time. This discrepancy is annoyingly obvious even for small matrices, and has been noted repeatedly in the literature since the last century [15]. Several attempts have been made to determine whether the permanent could be reduced to the determinant via some simple matrix transformation. The results have always been negative, except in certain special cases [12, 13, 16].

The aim of this paper is to explain the apparent intractability of the permanent by showing that it is "complete" as far as counting problems. The results can be summarized informally as follows:

Theorem 1. *The complexity of computing the permanent of $n \times n$ (0, 1)-matrices is NP-hard* [3, 11] *and, in fact, of at least as great difficulty (to within a polynomial factor) as that of counting the number of accepting computations of any nondeterministic polynomial time Turing machine.*

Finding a perfect matching is easy

Counting matchings is hard (#P-complete)

How hard is it to marry at random?
(On the approximation of the permanent)

ANDREI Z. BRODER
DEC – Systems Research Center
130 Lytton Avenue
Palo Alto, CA. 94301

Extended abstract

Abstract. It is well-known that the computation of the permanent of 0-1 matrices, which is the same as computing the number of matchings in bipartite graph, is #P-complete. However the complexity of computing a good approximation of the number of matchings, is an open question and it is the leading candidate for a problem for which *one* solution can be found in polynomial time, but for which even approximating the number of solutions is hard. In this paper we present a fully polynomial (ϵ, δ)-approximation scheme for the permanent of 0-1 matrices where at least half of the entries in every row and every column are 1's.

The novel algorithm uses a Markov chain that converges to the uniform distribution on the space of perfect matchings for any given graph. We show that it converges in polynomial time (in terms of the variation distance) for all dense enough graphs. Based on this chain we construct a sampling scheme that allows us to approximate the permanent of dense 0-1 matrices in polynomial time. Finally we show that the exact computation of the permanent of such matrices is still #P-complete.

1. Introduction

One of the most surprising results in computational complexity is that computing the number of perfect matchings in a bipartite graph is #P-complete, that is, as hard as computing the number of solutions of any problem in NP [Valiant79]. In other words although finding *a* perfect matching is easy and finding *a* Hamiltonian circuit is hard, counting perfect matchings and counting Hamiltonian circuits is equally hard.

The number of perfect matchings in a bipartite graph $G(V_1, V_2, E)$ where $|V_1| = |V_2| = n$ and $E \subset V_1 \times V_2$ is the same as computing the *permanent* of a square 0-1 matrix $M = (m_{i,j})$ of size n where

$$m_{i,j} = \begin{cases} 1, & \text{if } (i,j) \in E; \\ 0, & \text{otherwise,} \end{cases}$$

and the permanent of M, $\mathbf{per}(M)$ is defined by

$$\mathbf{per}(M) = \sum_{\sigma} \prod_{1 \le i \le n} m_{i,\sigma(i)},$$

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

The permanent function has a long and noble history; it was introduced by Cauchy in 1812 in its celebrated memoir on determinants and almost simultaneously by Binet. (See [Minc78] for detailed history.) It has important applications in statistical physics and chemistry and plays a central role in many enumeration and linear algebra problems.

Despite many efforts, the fastest known algorithm for the exact computation of the permanent requires $O(n2^n)$ operations. (It is based on Ryser's formula [Ryser63]. See [NW75] for implementation.) Some of the difficulty seems to reside in the fact that although the permanent is closely related to the determinant, it lacks the symmetries of the

Errata to
"How hard is it to marry at random? (On the approximation of the permanent)" [1]

Andrei Broder
DEC – SRC, 130 Lytton Ave., Palo Alto, CA 94301

The coupling argument used in the proof of Theorem 8 (the rapid convergence of the Markov chain MC1) as sketched in Appendix A, is incorrect. Recently however, M. Jerrum and A. Sinclair [2] showed by an entirely different method that MC1 is indeed rapidly converging (that is, the variation distance becomes less than ϵ in time polynomial in n, $1/\epsilon$, and the ratio M_{n-1}/M_n.) Therefore the approximation scheme for the permanent of dense graphs works in polynomial time, as stated.

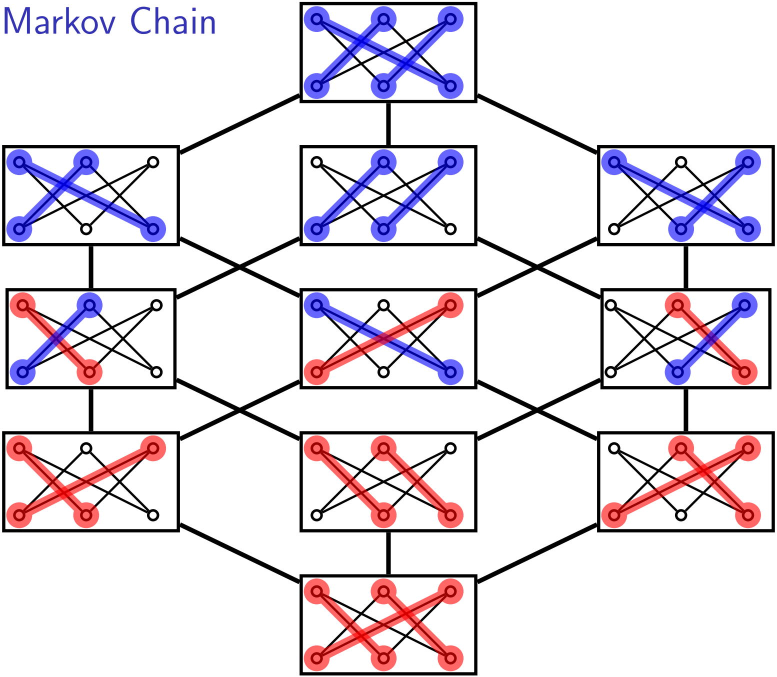
The flaw in the original proof comes from the fact that although the distribution of a card in the queue of moves for some position is uniform at the time of its insertion, it is not necessarily uniform at the time of its removal. This was first observed by M. Mihail [3].

[1] *Proceedings of the 18-th Annual ACM Symposium on Theory of Computing*, 1986, 50–58.

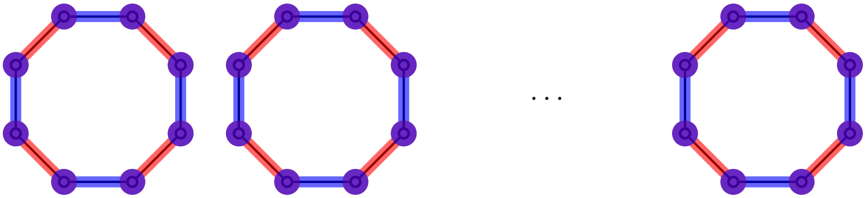
[2] M. Jerrum and A. Sinclair, "Conductance and the Rapid Mixing Property for Markov Chains: The Approximation of the Permanent resolved," *Proceedings of the 20-th Annual ACM Symposium on Theory of Computing*, 1988.

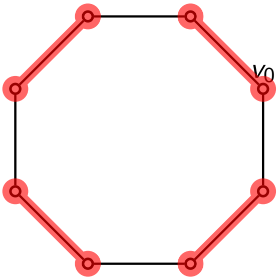
[3] M. Mihail, "The approximation of the permanent is still open," manuscript, Aiken Computation Laboratory, Harvard University, 1987.

The Markov Chain

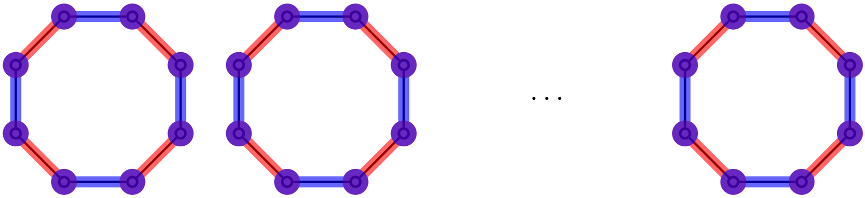


Symmetric Differences of Matchings

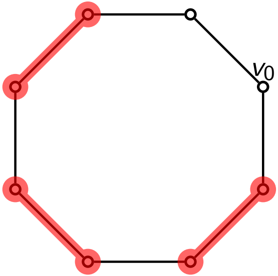




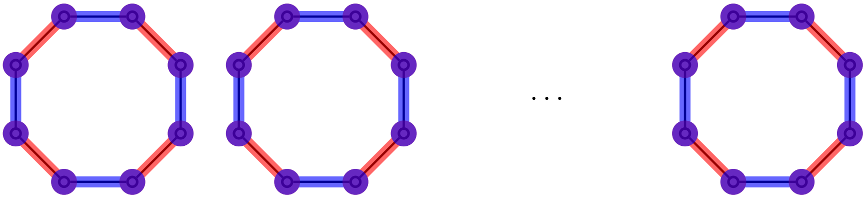
Symmetric Differences of Matchings



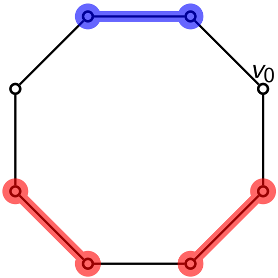
Unwinding Cycles



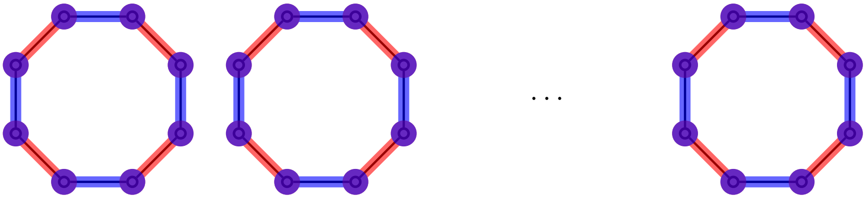
Symmetric Differences of Matchings



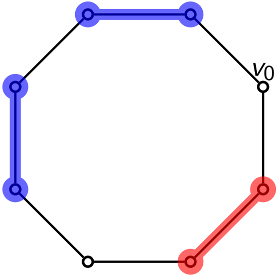
Unwinding Cycles



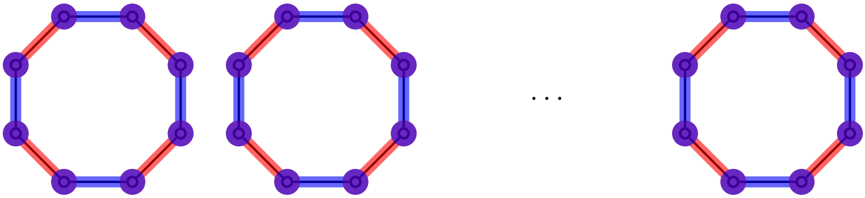
Symmetric Differences of Matchings



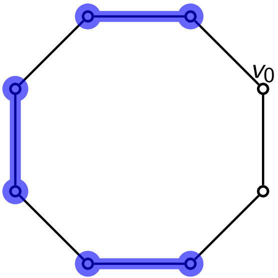
Unwinding Cycles



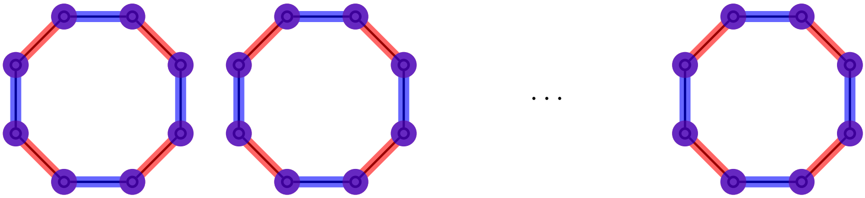
Symmetric Differences of Matchings



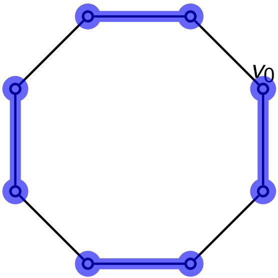
Unwinding Cycles



Symmetric Differences of Matchings



Unwinding Cycles



Canonical Paths Using Transition t