A Permanent Formula With Many Zero-Valued Terms

Eric Bax*and Joel Franklin†

June 19, 1997

Abstract

Applying finite-differences to a generating function produces formulas for the permanent of a matrix. We present a setting of the finite-difference parameters for which the permanent formula has many zero-valued terms when applied to 0-1 matrices. We outline a method to reduce computation by eliminating sets of zero-valued terms and show that the method significantly increases the computation speed.

Key words algorithms, combinatorial problems, #P, permanent, finite-difference, generating function.

AMS subject classifications 05,68

*Computer Science Department, California Institute of Technology 256-80, Pasadena, California, 91125 (eric@cs.caltech.edu).

†Department of Applied Mathematics, California Institute of Technology 217-50, Pasadena, California, 91125 (jnf@ama.caltech.edu).

1 Introduction

The permanent of an $n \times n$ 0-1 matrix is the number of n-sets of one-valued entries with one entry from each row and one entry from each column. If the matrix indicates adjacencies in a graph, then the permanent is the number of sets of cycles with each vertex on exactly one cycle. If the rows represent workers, the columns represent jobs, and one-valued entries indicate worker-job compatibility, then the permanent is the number of assignments in which each worker has a job and each job has a worker.

The problem of computing the 0-1 matrix permanent is #P-complete [9]. For example, #P problems include counting the Hamiltonian cycles in a graph, vertex colorings, and assignments that satisfy a CNF formula.

Ryser [8] developed an inclusion and exclusion algorithm for the permanent. The algorithm has time complexity $O(2^n \text{ poly } n)$ and space complexity $O(\text{poly } n)$. Finite-difference formulas [3] include Ryser's formula as a special case, and they have the same worst-case complexities. However, finite-difference formulas have some free parameters, and, in the average case, the computation can be significantly reduced by choosing appropriate parameter settings.

For a parameter setting analyzed in [3], the expected fraction of zero-valued terms goes to 100% as the matrix size increases. This paper introduces a superior parameter setting, with even more zero-valued terms. The new parameter setting produces increases in computation speed that are especially dramatic for sparse matrices.

2 Finite-Differences

Definition 1 (Finite-Difference Operator) *The finite-difference operator with respect to x_j, written $D_j(u_j, v_j)$, is defined as follows:*

$$\forall\ u_j \neq v_j\ D_j(u_j, v_j) f(x_1, \ldots, x_n) \equiv \frac{f(x_1, \ldots, x_j = u_j, \ldots, x_n) - f(x_1, \ldots, x_j = v_j, \ldots, x_n)}{u_j - v_j} \tag{1}$$

2.1 Notation

To simplify formulas, we denote assignments to variables in $f(x_1, \ldots, x_n)$ by their assigned values and omit free variables. Some examples:

- $f()$ represents $f(x_1, \ldots, x_n)$.
- $f(u_i)$ represents $f(x_1, \ldots, x_i = u_i, \ldots, x_n)$.
- $f(u_i, v_j)$ represents $f(x_1, \ldots, x_i = u_i, \ldots, x_j = v_j, \ldots, x_n)$.

Also, we abbreviate $D_j(u_j, v_j)$ by D_j. In the shortened notation, the definition of the finite-difference operator can be rewritten as:

$$D_j f() \equiv \frac{f(u_j) - f(v_j)}{u_j - v_j} \tag{2}$$

2.2 Finite-Differences and Multilinear Terms

For $P()$ a polynomial with every term of degree n or less, $D_1 \cdots D_n P()$ is the coefficient of the multilinear term $x_1 \cdots x_n$. A proof of this property is given in [3]. Intuitively, each D_j acts as a derivative with respect to x_j, so the composition $D_1 \cdots D_n$ finds the derivative with respect to every variable. Since the polynomial's terms have degree n or less, each term other than the multilinear term lacks some variable. Since the term is constant with respect to the missing variable, the difference operator that corresponds to the missing variable kills the term.

2.3 Finite-Difference Formulas

Expanding the finite difference operators produces a formula for the multilinear term:

$$D_1(u_1, v_1) \cdots D_n(u_n, v_n) P(x_1, \ldots, x_n) = \tag{3}$$

$$\frac{1}{(u_1 - v_1) \cdots (u_n - v_n)} \sum_{(x_1, \ldots, x_n) \in \{u_1, v_1\} \times \cdots \times \{u_n, v_n\}} (-1)^{s(x_1, \ldots, x_n)} P(x_1, \ldots, x_n) \tag{4}$$

where $s(x_1, \ldots, x_n)$ is the number of variables x_j set to v_j. Computing the formula requires 2^n evaluations of $P()$, so if each evaluation requires O(poly n) time and O(poly n) space, then the entire computation requires O(2^npoly n) time and O(poly n) space.

3 The Permanent is the Multilinear Term of a Polynomial

The permanent of matrix A is defined by:

$$\text{per } A = \sum_{j_1 \cdots j_n} a_{1j_1} \cdots a_{nj_n} \tag{5}$$

where $j_1 \cdots j_n$ is a permutation of $1 \ldots n$. Note that each term has one entry from each row and one entry from each column.

Examine the product of row sums:

$$\prod_{i=1}^{n} \sum_{j=1}^{n} a_{ij} = \sum_{(j_1,\ldots,j_n) \in \{1,\ldots,n\}^n} a_{1j_1} \cdots a_{nj_n} \tag{6}$$

Each term has one entry from each row. The permanent terms are the terms that also have one entry from each column.

Define $[A(\mathbf{x})]_{ij} = [A]_{ij} x_j$, i.e. multiply each column j by x_j. Examine the product of row sums of $A(\mathbf{x})$:

$$P(\mathbf{x}) = \prod_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_j = \sum_{(j_1,\ldots,j_n) \in \{1,\ldots,n\}^n} a_{1j_1} \cdots a_{nj_n} x_{j_1} \ldots x_{j_n} \tag{7}$$

The variables x_j indicate the columns from which the entries were drawn. Since the permanent terms have one element from each column, the permanent is the coefficient of the multilinear term of $P(\mathbf{x})$. Hence,

$$\text{per } A = D_1 \cdots D_n P(\mathbf{x}) \tag{8}$$

4 Setting Finite-Difference Parameters to Produce Zero-Valued Terms

Assume the entries of A are determined randomly and independently, with each entry taking value one with probability p and value zero with probability $q = 1 - p$. Assume $0 < p < 1$.

In finite-difference formula (4), a term has value zero when $P(\mathbf{x})$ has value zero. This occurs when the entries of some row of $A(\mathbf{x})$ sum to zero.

Given $\mathbf{x}$, the probabilities of row sums being zero are independent. So the expected fraction of zero-valued terms in the formula is:

$$1 - 2^{-n} \sum_{(x_1,\ldots,x_n)\in\{u_1,v_1\}\times\cdots\times\{u_n,v_n\}} [1 - \Pr\{a_1x_1 + \ldots + a_nx_n = 0\}]^n \qquad (9)$$

where $a_j = 1$ with probability p and $a_j = 0$ with probability $q = 1 - p$.

We will specify expression (9) for three finite-difference settings – the inclusion and exclusion setting, a setting previously used to produce many zero-valued terms [3], and the new setting that produces more zero-valued terms.

Setting $\mathbf{u} = \mathbf{1}$ and $\mathbf{v} = \mathbf{0}$ produces the inclusion and exclusion formula [8, 3]. The expected fraction of terms with value zero is:

$$1 - 2^{-n} \sum_{(x_1,\ldots,x_n)\in\{0,1\}\times\cdots\times\{0,1\}} [1 - \Pr\{a_1x_1 + \ldots + a_nx_n = 0\}]^n \qquad (10)$$

Let k be the number of variables assigned one.

$$= 1 - \sum_{k=0}^{n} \frac{C(n,k)}{2^n} [1 - \Pr\{a_1 + \ldots + a_k = 0\}]^n \qquad (11)$$

$$= 1 - \sum_{k=0}^{n} \frac{C(n,k)}{2^n} [1 - q^k]^n \qquad (12)$$

For fixed $p > 0$, this value goes to zero as n increases, i.e. the expected fraction of zero-valued terms goes to 0% (see Figure 1).

Now set $\mathbf{u} = \mathbf{1}$ and $\mathbf{v} = -\mathbf{1}$. Once again, let k be the number of variables assigned one. The expected fraction of zero terms is:

$$1 - \sum_{k=0}^{n} \frac{C(n,k)}{2^n} [1 - \Pr\{a_1 + \ldots + a_k - (a_{k+1} + \ldots + a_n) = 0\}]^n \qquad (13)$$

Let ν be the number of $a_1, \ldots, a_k$ with value one, and sum over cases.

$$= 1 - Pr\{a_1 + \ldots + a_k - (a_{k+1} + \ldots + a_n) = 0\} = \sum_{\nu=0}^{\min(k,n-k)} C(k,\nu) p^\nu q^{k-\nu} C(n-k,\nu) p^\nu q^{(n-k)-\nu} \qquad (14)$$

For fixed $p < 1$, this value goes to one as n increases, i.e. the expected fraction of zero-valued terms goes to 100% (see Figure 1). The expected fraction of nonzero-valued terms is $\mathrm{O}(\frac{1}{\sqrt{n^{\frac{q}{p}} \log n}})$ [3].

The new setting is as follows: $u_j = 1$ for odd j, $u_j = -1$ for even j, and $\mathbf{v} = \mathbf{0}$. Let k_1 be the number of variables assigned one, and let k_2 be the number of variables assigned negative one. To simplify the derivation, assume n is even and let $m = \frac{n}{2}$. The expected fraction of zero-valued terms is:

$$1 - \sum_{k_1=0}^{m} \frac{C(m,k_1)}{2^m} \sum_{k_2=0}^{m} \frac{C(m,k_2)}{2^m} [1 - \Pr\{a_1 + \ldots + a_{k_1} - (a_{k_1+1} + \ldots + a_{k_1+k_2}) = 0\}]^n \tag{15}$$

Let ν be the number of $a_1, \ldots, a_{k_1}$ with value one, and sum over cases.

$$1 - \sum_{k_1=0}^{m} \frac{C(m,k_1)}{2^m} \sum_{k_2=0}^{m} \frac{C(m,k_2)}{2^m} [1 - \sum_{\nu=0}^{\min(k_1,k_2)} C(k_1,\nu) p^\nu q^{k_1-\nu} C(k_2,\nu) p^\nu q^{k_2-\nu}]^n \tag{16}$$

For fixed $p < 1$, this value goes to one as n increases, i.e. the expected fraction of zero-valued terms goes to 100%. The expected fraction of nonzero-valued terms is $O(\frac{1}{n^{\frac{q}{p}} \sqrt{\log n}})$ [4].

The new setting produces a significantly higher fraction of zero terms than the $\mathbf{u} = \mathbf{1}$ and $\mathbf{v} = -\mathbf{1}$ setting (see Figure 1). Figure 2 shows the expected fractions of zero-valued terms in the new formula for various entry probabilities p. For sparse matrices ($p = 0.25$), the expected fraction of zero-valued terms is nearly 100% for matrices of size 20 and larger.

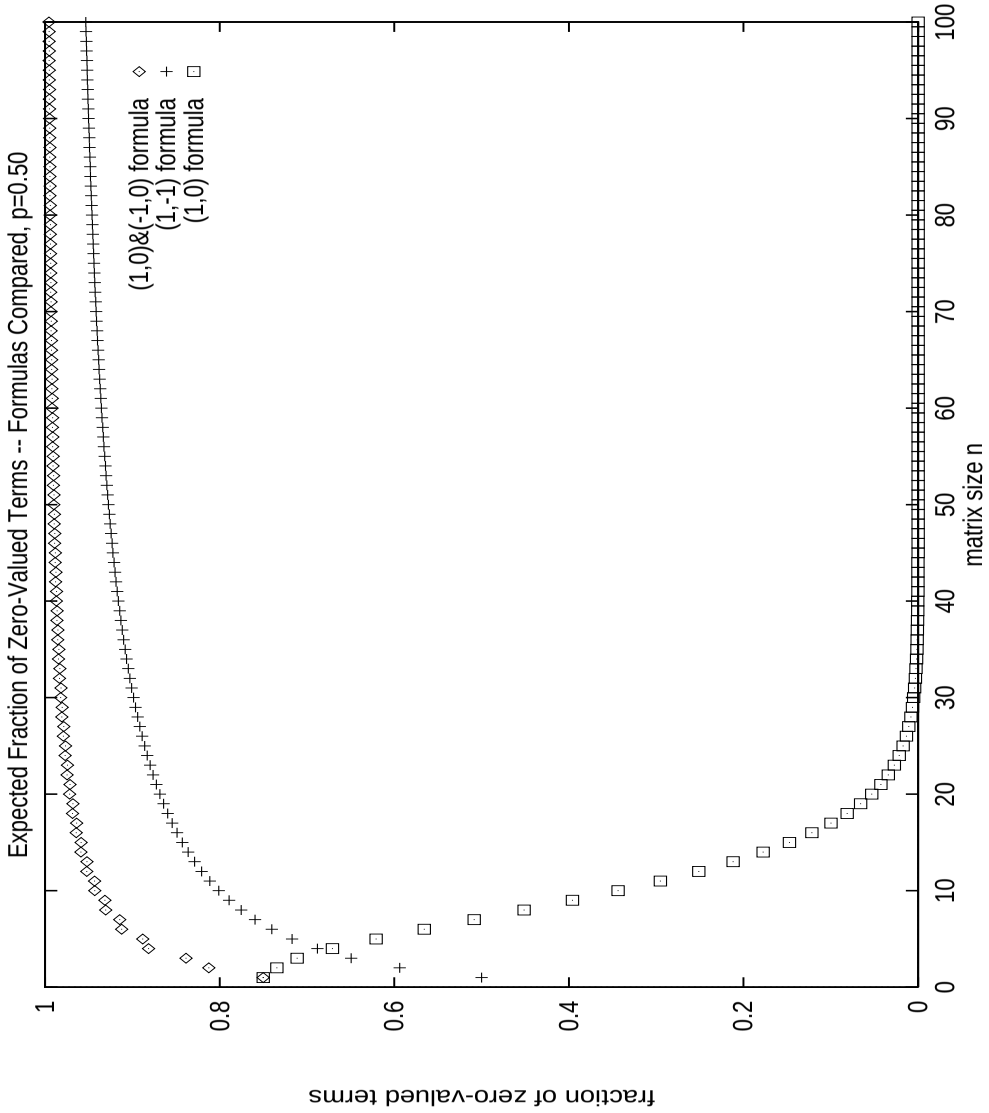

Figure 1: The new formula has more zero-valued terms than the $(\mathbf{1},-\mathbf{1})$ formula.

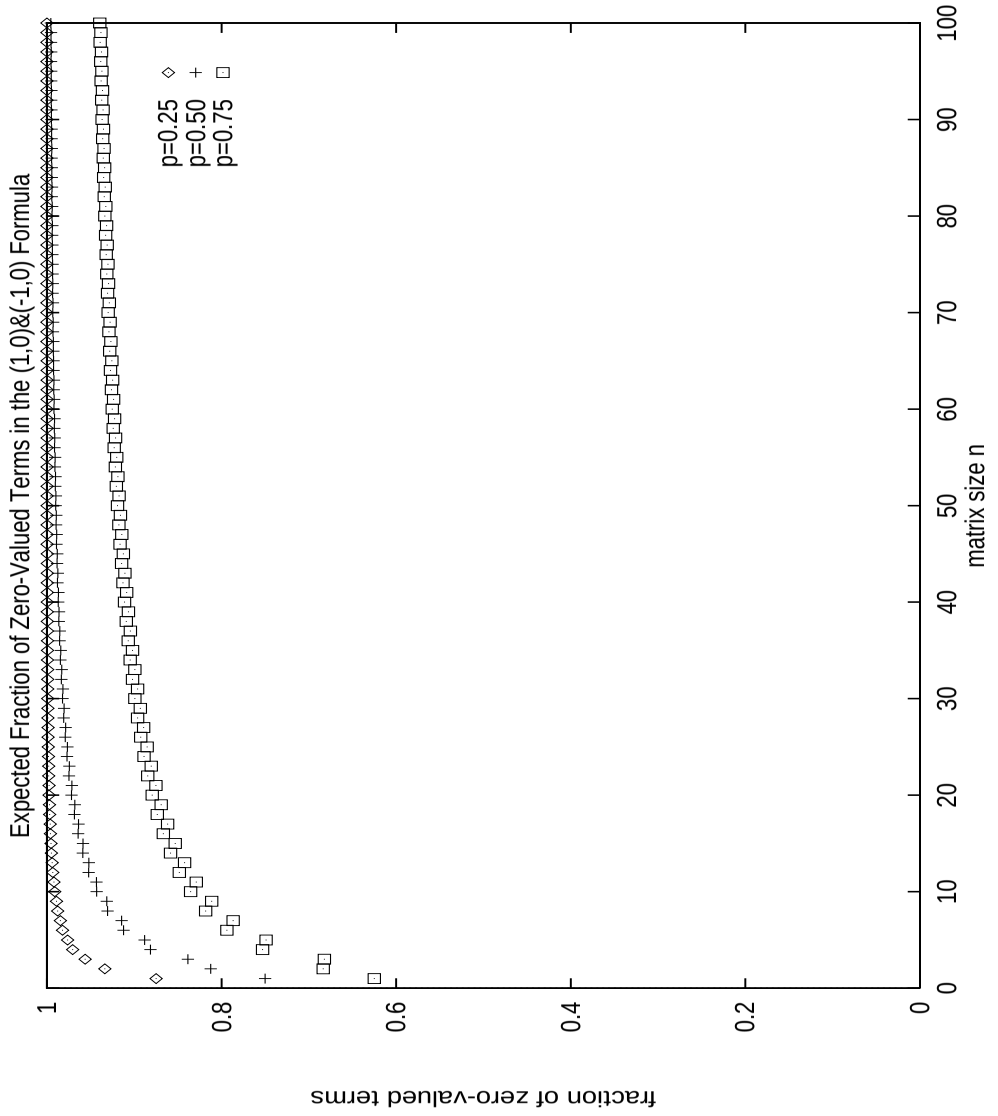

Figure 2: The expected fraction of zero-valued terms in the new formula is nearly 100% for sparse matrices ($p = 0.25$).

5 Eliminating Sets of Zero Terms to Speed Computation

The following algorithm computes the permanent by evaluating the finite-differences in the expression

$$\text{per } A = D_1(u_1, v_1) \cdots D_n(u_n, v_n) P(\mathbf{x}) \tag{17}$$

Matrix A, matrix size n, and finite difference parameters $\mathbf{u}$ and $\mathbf{v}$ are global variables. The function call F(1,$\mathbf{0}$) evaluates the permanent. Each function call F(j,$\mathbf{r}$) evaluates $D_j \cdots D_n P(\mathbf{x})$. Variable $\mathbf{r}$ accumulates row sums as the variables x_j receive assignments. In each function call,

$$r_i = \sum_{k=1}^{j-1} a_{ij} x_j \tag{18}$$

We use $\mathbf{a}^j$ to denote column j of A.

F(j,$\mathbf{r}$):
{
if $j = n+1$ then return $\prod_{i=1}^{n} r_i$
else return $\frac{1}{u_j - v_j}[\text{F}(j+1, \mathbf{r} + u_j \mathbf{a}^j) - \text{F}(j+1, \mathbf{r} + v_j \mathbf{a}^j)]$
}

Each function call F(j,$\mathbf{r}$) assigns a value to x_j, and the descendent function calls assign values to $x_{j+1}, \ldots, x_n$. If, for some row, the partial row sum r_i is zero, and the entries in columns j through n are all zero, then the row sum will remain zero in all descendent function calls. Hence, all descendent function calls will return zero. The following algorithm avoids these descendent function calls and returns zero instead. Each global variable f_i stores the column of the rightmost nonzero element in row i.

G(j,$\mathbf{r}$):
{
if, for some row i, $f_i = j - 1$ and $r_i = 0$, then return 0
else if $j = n+1$ then return $\prod_{i=1}^{n} r_i$
else return $\frac{1}{u_j - v_j}[\text{G}(j+1, \mathbf{r} + u_j \mathbf{a}^j) - \text{G}(j+1, \mathbf{r} + v_j \mathbf{a}^j)]$
}

The computational savings scale exponentially with the level of recursion at which zero terms are collected. We encourage efficient collection of zero terms by permuting the columns of A to pack nonzero entries to the left, using the following procedure. First, choose a row with a minimum number of nonzero

p	% zero-valued terms	% function calls eliminated
0.25	99.38	98.08
0.50	94.10	81.22
0.75	82.84	52.49

Table 1: Matrix size 10 – percentages of terms with value zero and percentages of function calls eliminated. Each value is the average over 100 random matrices.

p	% zero-valued terms	% function calls eliminated
0.25	99.82	99.25
0.50	97.23	85.79
0.75	87.47	56.35

Table 2: Matrix size 20 – percentages of terms with value zero and percentages of function calls eliminated. Each value is the average over 100 random matrices.

entries. Permute the columns of A to pack the row's nonzero entries into the leftmost columns. Then choose a row with a minimum (but positive) number of nonzero entries in the remaining columns, and permute the remaining columns to pack those nonzero entries to the left. Repeat until all columns have been packed (or until columns with only zero entries remain, in which case the permanent is zero.)

Tables 1 and 2 show the results of using the preprocessing procedure with function G and the new formula. Our procedure is very effective for sparse matrices. For example, with matrix size $n = 20$ and entry probability $p = 0.25$, only $100\% - 99.25\% = 0.75\%$ of the function calls are executed, speeding up the computation by a factor of $\frac{1}{0.0075} \approx 133$. Also, note that the fraction of function calls avoided increases as the matrix size increases.

6 Discussion

Some parameter settings may produce even more zero-valued terms. We have found that the number of zero-valued terms can be increased by tailoring the finite-difference parameters to problem instances [1], and there is much more work to be done in this area. Furthermore, some settings may allow computational advantages by means other than producing zero-valued terms.

Our procedure to collect sets of zero-valued terms is crude and simple. The computational savings are significant for sparse matrices, but not so for dense matrices. More sophisticated or clever methods may lead to improvements in this area.

Finite-difference formulas may yield good estimators for the permanent. Formulas with few large terms and many small terms may be efficiently estimated by computing the large terms and inferring the sum of the small terms from a sample. In the formula introduced in this paper, terms with many variables x_j assigned zero and terms with about the same number of positive and negative assignments are likely to have small values. (For information on other approaches to estimating the permanent, see [5, 6, 7].)

Finally, finite-difference formulas can be developed for other problems. For example, finite-difference formulas have been used to count paths and cycles in graphs [2]. For counting problems, all that is needed is a generating function with degree equal to the number of variables and with the multilinear term corresponding to the objects to be counted. Once finite-difference formulas are developed, the challenge is to find parameter settings that produce reductions in computation.

7 Acknowledgements

We thank an anonymous referee and editor David Gries for advice regarding content and presentation. The first author thanks the second author for advice and guidance.

References

[1] E. Bax. Tailoring the permanent formula to problem instances. CalTech-CS-TR-96-17

[2] E. Bax and J. Franklin. A finite-difference sieve to count paths and cycles by length. *Information Processing Letters*. 60 (1996) 171-176.

[3] E. Bax and J. Franklin. A finite-difference sieve to compute the permanent. CalTech-CS-TR-96-04.

[4] E. Bax and J. Franklin. Expected fraction of zero-valued terms in a finite-difference formula for the permanent. CalTech-CS-TR-97-02.

[5] M. R. Jerrum and A. Sinclair. Approximating the permanent. *SIAM Journal on Computing*. 18(6):1149-1178. December 1989.

[6] M. Jerrum and U. Vazirani. A mildly exponential approximation algorithm for the permanent. *Algorithmica*. 16(1996):392-401.

[7] N. Karmarkar, R. Karp, R. Lipton, L. Lovász, and M. Luby, A Monte Carlo algorithm for estimating the permanent. *SIAM Journal on Computing*. 22(2):284-293. April 1993.

[8] H. J. Ryser. *Combinatorial Mathematics*. The Mathematical Association of America 1963, Ch. 2.

[9] L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*. 8(1979):189-201.