A hybrid algorithm for computing permanents of sparse matrices

Heng Liang a,*, Songqi Huang b, Fengshan Bai a

a Department of Mathematical Sciences, Tsinghua University, Beijing 100084, PR China
b Zhengzhou Institute of Light Industry, Zhengzhou 450002, PR China

Abstract

The permanent of matrices has wide applications in many fields of science and engineering. It is, however, a \#P-complete problem in counting. The best-known algorithm for computing the permanent, which is due to Ryser [Combinatorial Mathematics, The Carus Mathematical Monographs, vol. 14, Mathematical Association of America, Washington, DC, 1963], runs $O(n^{2^{n-1}})$ in time. It is possible to speed up algorithms for matrices with special structures, which arise commonly in applications. Most algorithms discussed before focus on 0,1 matrix. In this paper, a hybrid algorithm is proposed. It is efficient for sparse matrices.

© 2004 Elsevier Inc. All rights reserved.

* Supported by National Science Foundation of China G10228101 and National Key Basic Research Special Fund G1998020306.

* Corresponding author.

E-mail addresses: liangh01@mails.tsinghua.edu.cn (H. Liang), hsq02@mails.tsinghua.edu.cn (S. Huang), fbai@math.tsinghua.edu.cn (F. Bai).
1. Introduction

The permanent of an \( n \times n \) matrix \( A = [a_{ij}] \) is defined as

\[
\text{Per}(A) = \sum_{\sigma \in A_n} \prod_{i=1}^{n} a_{i\sigma(i)},
\]

where \( A_n \) denotes the set of all possible permutations of \( \{1, 2, \ldots, n\} \). Valiant [18] proves that computing the permanents of matrices is a \#P-complete problem in counting. The best-known algorithm on precise evaluation of the permanent, due to Ryser [16] and later improved by Nijenhuis and Wilf [15], runs \( O(n2^{n-1}) \) in time.

There are wide applications of the permanent both inside and outside mathematics. The permanent of a matrix is expected to be computed in computational molecular chemistry [7,17] with high precision. In Bose-Einstein statistics, the permanent is used to symmetrize the bosonic density matrix [8]. The permanent is also an important problem in combinatorial counting and graph theory [14], as well as numerical solution of polynomial systems [12]. Its computational methods therefore attract more and more attentions recently [3,4,9,11]. For the reason of the intrinsic difficulty of the problem, researches are mostly focused on approximation methods. However, matrices with special structures, such as 0,1 and/or sparse matrices with certain structures, are of great interests in many applications. At the same time, the precise evaluation of permanent is often necessary. By making use of the structural properties, it is possible to make the calculation faster. For 0,1 matrices, there exist efficient precise methods, including Kallman’s method [10] and finite-difference methods [1,2]. In this paper, a hybrid algorithm on the permanent are presented. It also works for general nonnegative sparse matrices.

In the next section, we briefly review Ryser–Nijenhuis and Wilf method (R–NW) [15,16] and the direct expansion method (DEM) [13]. In Section 3, an improved direct expansion method (IDEM) is proposed. Then a hybrid algorithm is constructed by combining IDEM with R–NW in Section 4. Numerical results are presented in Section 5. It shows that the hybrid algorithm is very efficient, if the number of nonzero entries of an \( n \times n \) matrix is less than \( 5n \).

2. Precise algorithms for permanent

We prescribe some terminologies used in this paper. Let \( |S| \) denote the number of elements of the set \( S \). Let \( A(i,:) \) denote the \( i \)th row of \( A \), and \( A(:,j) \) denote the \( j \)th column of \( A \). Let \( H(i,:) \) denote the set of columns indices of nonzero entries in \( A(i,:) \) and \( H(:,j) \) denote the set of rows indices of nonzero entries in \( A(:,j) \). For a tree, let \( \text{child}(D) \) denote the set of child nodes of node \( D \).
2.1. Ryser–Nijenhuis and Wilf method (R–NW)

Ryser [16] gives a formula for the permanent of matrix \( A \) as

\[
\text{Per}(A) = (-1)^n \sum_S (-1)^{|S|} \prod_{j \in S} a_{ij}.
\]

(2)

Here \( A \) is the \( n \times n \) matrix and \( S \) runs over all possible subsets of \( \{1, 2, \ldots, n\} \). This method is based on the inclusion and exclusion principle.

The computational cost of Ryser method is \( O(n^2 \cdot 2^{n-1}) \). Nijenhuis and Wilf [15] improve Ryser method from the viewpoint of effective implementation.

Note that for each subset \( S \), one has to calculate

\[
f(S) = \prod_{i=1}^{n} \hat{\lambda}_i(S),
\]

where

\[
\hat{\lambda}_i(S) = \sum_{j \in S} a_{ij} \quad (i = 1, 2, \ldots, n).
\]

Arranging the sequence of all subsets according to Hamilton walk on an \((n - 1)\)-cube (Chapter 1 of [15]), the current subset \( S \) differs from its predecessor \( S' \) only by a single element, say \( j \). Thus

\[
\hat{\lambda}_i(S) = \hat{\lambda}_i(S') \pm a_{ij} \quad (i = 1, 2, \ldots, n).
\]

This reduces the number of additions in a subset \( S \) from \( n \cdot |S| \) to \( n \). So the total computational cost becomes \( O(n \cdot 2^{n-1}) \).

2.2. Direct expansion method for sparse matrix (DEM)

Mittal and Al-Kurdi [13] present an algorithm based on the numerical structure of a matrix defined in [6]. An expansion tree is introduced first. The root node of the tree is denoted by \( O \), and every other node represents an nonzero element of the matrix. A node with depth \( k \) belongs to \( H(k, :) \), so it can be denoted by the column index of the element. All simple downward paths of the tree must be depth \( n \) and they are one-to-one correspondent to all nonzero terms of (1). For example, if \((O, i_1, i_2, \ldots, i_n)\) is a simple downward path, then \( a_{1i_1}a_{2i_2}\ldots a_{ni_n} \) is a nonzero term of (1). Those trees are constructed efficiently with methods in [13], so that all nonzero terms in (1) are found.

The example in Fig. 1 will make the idea clear. Let

\[
A = \begin{pmatrix}
a_{11} & 0 & a_{13} & 0 \\
0 & a_{22} & a_{23} & 0 \\
0 & 0 & a_{33} & a_{34} \\
a_{41} & a_{42} & a_{43} & a_{44}
\end{pmatrix}.
\]
The expansion tree of matrix $A$ is shown in this figure, where the number marked nearby a node is the column index of the element. Hence we have

$$\text{Per}(A) = a_{11}a_{22}a_{33}a_{44} + a_{11}a_{22}a_{34}a_{43} + a_{11}a_{23}a_{34}a_{42} + a_{13}a_{22}a_{34}a_{41}. $$

The corresponding method can be described as follows.

*Direct expansion method (DEM)* [13]

Arranging the rows of $A$ such that $|H(1,:)| \leq |H(2,:)| \leq \cdots \leq |H(n,:)|$. Let $\beta_1 = \text{child}(O)$; i.e. $\beta_1 = H(1,:)$. For $i = 2, 3, \ldots, n$

- For each $\beta_{i-1} \in \beta_{i-1}$, we have $(\beta_{i-1}, \beta_{i-1})$ such that $\beta_{i,k} \in \text{child}(\beta_{i-1})$ $\forall k < i$;
- Let child($\beta_{i-1}$) = $H(i,:)\setminus\{\beta_{i-1}, \ldots, \beta_{i-1}\}$;
- $\beta_i = \cup_{\beta_{i-1} \in \beta_{i-1}} \text{child}(\beta_{i-1})$;

End

$$P = \sum_{\sigma \in \Omega} a_{1\sigma(1)} \cdots a_{n\sigma(n)},$$ where $\Omega$ is the set of all possible downward paths.

The DEM is essentially a direct expansion method in nature owing to that all nonzero terms in (1) have to be considered.

**Theorem 2.1.** Algorithm DEM corrects evaluates the permanent of $n \times n$ matrix $A$ in $O(n!\prod_{i=1}^{n} |H(i,:)|)$ operations.
3. An improved direct expansion method (IDEM)

Consider the expansion by pair property

\[
\text{Per}
\begin{pmatrix}
  a & b & x^T \\
  y_1 & y_2 & Z
\end{pmatrix}
= \text{Per}
\begin{pmatrix}
  0 & 0 & x^T \\
  y_1 & y_2 & Z
\end{pmatrix}
+ \text{Per}(ay_2 + by_1 Z),
\tag{3}
\]

where \(a\) and \(b\) are scalars, \(x^T\) is an \((n - 2)\)-dimensional row vector, \(y_1\) and \(y_2\) are both \((n - 1)\)-dimensional column vectors, and the left is \((n - 1) \times (n - 2)\) matrix \(Z\). The expansion appears in [8] and is used to establish an approximate algorithm for permanent. We now improve the direct expansion method based on this relation. The algorithm is as follows.

**Improved direct expansion method (IDEM)**

\[
P = \text{IDEM}(A);
\]

Find \(i\) and \(j\) such that 

\[
|H(i,:)| = \min_{1 \leq k \leq n} \{|H(k,:)|, \ |H(:,j)| = \min_{1 \leq k \leq n} \{|H(:,k)|};
\]

if \(|H(:,j)| < |H(i,:)|\), then exchange \(A(:,1)\) and \(A(:,j)\), let \(A = A^T\);
else exchange \(A(1,:)\) and \(A(i,:);\)

Permute the column of \(A\) to move all nonzero entries in the first row to the left;

Use (3) to get two matrices \(A_1, A_2, P = \text{IDEM}(A_1) + \text{IDEM}(A_2).\)

**Theorem 3.1.** Let \(\lceil x \rceil\) denote the smallest integer such that \(\lceil x \rceil \geq x\). Then algorithm IDEM correctly evaluates the permanent of \(n \times n\) matrix \(A\) in \(O(n^2 \prod_{i=1}^{n} \lceil \frac{|H(i,:)|}{2} \rceil)\) operations.

**Proof.** \(\text{Per}(A)\) can be divided to the sum of permanents of at most \(\lceil \frac{|H(1,:)|}{2} \rceil (n - 1) \times (n - 1)\) matrices. And during the process of recursion, the permanent of a \(k \times k\) matrix may be divided to the sum of \(\lceil \frac{|H(n-k+1,:)|}{2} \rceil (k - 1) \times (k - 1)\) sub-terms at most. So there are not more than \(\prod_{i=1}^{n} \lceil \frac{|H(i,:)|}{2} \rceil\) monomials and getting each monomial contains at most

\[
3 \cdot [(n - 1) + (n - 2) + \cdots + 1] = \frac{3}{2} \cdot n(n - 1)
\]

additions and multiplications operations. Hence the conclusion is proved. \(\square\)

Essentially, the IDEM eliminates two entries each time and the DEM just one. Hence the IDEM may be faster than DEM exponentially. The numerical experiments confirm the theoretic result.
We randomly generate eight very sparse $30 \times 30$ matrices $A_1, \ldots, A_8$ and their permanents are all nonzero. The values of permanents and CPU times by DEM and IDEM are listed in Table 1.

As the values of permanents grow larger, the DEM slows down dramatically. Hence the improvements leading by the IDEM is distinct.

### 4. A hybrid algorithm

Note that the improved direct expansion method is a recursive algorithm. It is faster than R–NW when the matrix is very sparse. Hence we can combine IDEM with R–NW to establish hybrid algorithm. First, we recursively call IDEM, and the sub-matrices may become denser. When the order of submatrices is reduced to some extend or the sub-matrix is dense to some extend, we evaluate it by R–NW. A hybrid algorithm can be established.

**Algorithm H**

\[
P = H(A);
\]

\[
n \leftarrow \text{the order of } A;
\]

\[
s \leftarrow \text{the minimal number of non-zeros entries in one row or column of } A;
\]

if \( n > 2 \) and \( s < 5 \), then divide \( A \) to \( A_1, A_2 \) using IDEM, then \( P = H(A_1) + H(A_2) \);

else return R–NW(A).

**Theorem 4.1.** Algorithm H evaluates the permanent of an \( n \times n \) matrix \( A \) in \( O(n2^{n-1}) \) operations.

**Proof.** We call the matrix, which has been called recursively, the medial matrix. Let the running time for the medial matrix with order \( k \) is \( T(k) \) and the running time of R–NW is \( R(k) \). Either \( T(k) = R(k) = O(k2^{k-1}) \) when \( s > 4 \), or \( T(k) \leq [s/2] \cdot [T(k-1) + 3(k-1)] \leq 2 \cdot [T(k-1) + 3(k-1)] \). Then we assume that we have had \( T(i) = O(i2^{i-1}) \) when \( i < k \). We can get \( T(k) = R(k) \) or
\[ T(k) \leq 2 \cdot [T(k-1) + 3(k-1)] = 2 \cdot O((k-1)2^{k-2}) + 6(k-1) = O(k2^{k-1}). \]

So, we have \( T(n) = O(n2^{2n-1}) \). \qed

Theorem 4.1 shows that the computational complexity of the hybrid Algorithm H is the same as that of Ryser method. Hence it seems no improvement. Computational cost of Ryser method is fixed as long as the matrix size \( n \) fixed. It is robust but takes no benefits from the special structure of matrices. The computational complexity analysis of the hybrid Algorithm H obtained by the Theorem 4.1 above, is the worst case. It saves drastically by using structural information.

5. Numerical results

Let \( n \) denote the order of the matrix and \( m \) denote the number of nonzero entries in the matrix. All test matrices are 0,1 valued and generated randomly. All the methods used in this section do not have any restriction in entry valued. All the numerical computations in this paper are carried on a Pentium PCs (750 MHZ). Matlab6.5 is used as the programming language for simplicity.

Table 2 shows the CPU times for the three methods with different densities in the case of \( n = 15 \). And we have the similar result when \( n \) is larger than 15. When \( m \leq 5n \), the time of hybrid methods will be lower than that of R–NW drastically. Hybrid methods are always better than IDEM.

Then we give the CPU times of R–NW, Algorithm H for \( n = 10, 15, 20, \ldots \) under the different sparsity, i.e. the condition of \( m = 4n, 5n, 6n \) respectively. See Table 3. If the matrix is so sparse that its nonzero entries are not more than \( 4n \), the permanent of \( 40 \times 40 \) matrix can be obtained in just 1 min by H. As \( m \) reaches to around \( 5n \), H still keeps its advantage clearly. And as for \( m = 6n \), it seems to be a critical density that R–NW and H are comparative.

The permanent of adjacency matrices of fullerenes is an interested parameter in chemical graph theory [5,17]. Table 4 gives the computational results on \( C_{30} \).

<table>
<thead>
<tr>
<th>( m )</th>
<th>( 2n )</th>
<th>( 3n )</th>
<th>( 4n )</th>
<th>( 5n )</th>
<th>( 6n )</th>
<th>( 7n )</th>
<th>( 8n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>R–NW</td>
<td>1.68</td>
<td>1.68</td>
<td>1.68</td>
<td>1.68</td>
<td>1.68</td>
<td>1.68</td>
<td>1.68</td>
</tr>
<tr>
<td>IDEM</td>
<td>0.01</td>
<td>0.01</td>
<td>0.03</td>
<td>0.26</td>
<td>4.47</td>
<td>41.1</td>
<td>108.3</td>
</tr>
<tr>
<td>H</td>
<td>&lt;0.01</td>
<td>&lt;0.01</td>
<td>0.01</td>
<td>0.06</td>
<td>0.781</td>
<td>1.63</td>
<td>1.65</td>
</tr>
<tr>
<td>Per</td>
<td>2</td>
<td>63</td>
<td>893</td>
<td>21,299</td>
<td>853,152</td>
<td>7,304,392</td>
<td>59,686,500</td>
</tr>
</tbody>
</table>
Both theoretic and computational results show that Algorithm H keeps the advantages of both R–NW and IDEM. It runs faster than Ryser’s method if the matrix contains rows or columns whose numbers of nonzero entries are less than 5, otherwise it is the same as R–NW. Larger fullerenes, such as $C_{60}$, can be computed by the hybrid algorithm.

<table>
<thead>
<tr>
<th>Table 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computational tests for different sparsity</td>
</tr>
<tr>
<td>Sparsity</td>
</tr>
<tr>
<td>$m = 4n$</td>
</tr>
<tr>
<td>R–NW</td>
</tr>
<tr>
<td>H</td>
</tr>
<tr>
<td>$m = 5n$</td>
</tr>
<tr>
<td>R–NW</td>
</tr>
<tr>
<td>H</td>
</tr>
<tr>
<td>$m = 6n$</td>
</tr>
<tr>
<td>R–NW</td>
</tr>
<tr>
<td>H</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computational result of $C_{30}$</td>
</tr>
<tr>
<td>Per</td>
</tr>
<tr>
<td>R–NW</td>
</tr>
<tr>
<td>29,621</td>
</tr>
</tbody>
</table>

Both theoretic and computational results show that Algorithm H keeps the advantages of both R–NW and IDEM. It runs faster than Ryser’s method if the matrix contains rows or columns whose numbers of nonzero entries are less than 5, otherwise it is the same as R–NW. Larger fullerenes, such as $C_{60}$, can be computed by the hybrid algorithm.

**Acknowledgments**

The authors thank referees for their careful reading and helpful suggestions and comments.

**References**