Computing the Permanent

Patrick Lei

UMass CS 741

October 22, 2019

1 Basic Notions and Notation

Recall that the determinant is given by the formula

$$\det A = \sum_{\sigma \in S_n} \operatorname{sign}(\sigma) \prod_{i=1}^{n} A_{i,\sigma(i)}.$$

Similarly, the permanent is given by

$$\operatorname{perm} A = \sum_{\sigma \in S_n} \prod_{i=1}^{n} A_{i,\sigma(i)}.$$

Remark 1. Given a graph G, the permanent $\operatorname{perm} A$ of its adjacency matrix A is the number of vertex-disjoint cycle covers of G.

Next, we define some complexity classes.

Definition 2. A counting-TM is a standard NTM with an auxillary output device that magically[1] prints in binary the number of accepting computations on the given input. It has time complexity $f(n)$ if the longest accepting computation runs in time $f(n)$.

Definition 3. $\#P$ is the class of functions that can be computed by counting TMs of polynomial time complexity.

Here we will use the notion of an oracle reduction, where oracles can be any poly-time computable function. Note that for NP-complete problems, it is relatively easy to prove $\#P$-completeness for their counting versions. However, there are problems in P whose counting versions are $\#P$-complete.

Example 4. We can compute in polynomial time whether or not a formula is a tautology (look for something of the form $x \vee \overline{x}$), but finding the number of assignments that make a CNF-formula ϕ false is $\#P$-complete (doing this computes $\#SAT$).

Example 5. The permanent corresponds to counting perfect matchings, so the corresponding decision problem is in P (Ford-Fulkerson, from CS311).

2 Hardness Results

We begin with some lemmas.

Lemma 6. *There is a polynomial-time computable function f from CNF-formulas to matrices with entries in $\{-1, 0, 1, 2, 3\}$ such that for all formulas,*

$$\operatorname{perm}(f(\phi)) = 4^{t(\phi)} s(\phi),$$

where $t(\phi) = 2\#\{\text{number of occurrences of literals}\} - \#\{\text{clauses}\}$ and $s(\phi)$ is the number of satisfying assignments.

Proof. We will constructed a weighted directed graph G whose adjacency matrix has the desired permanent. We will assume that each clause in the formula ϕ has three literals. We will construct a track T_k for each variable x_k, an interchange R_i for each clause C_i, and for each literal $y_{i,j}$ that is either $x_k, \overline{x_k}$, a junction $J_{i,k}$ where R_i, T_k meet. Interchanges have internal junctions, and all junctions are given by the following adjacency matrix:

$$\begin{pmatrix} 0 & 1 & -1 & -1 \\ 1 & -1 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 0 & 1 & 3 & 0 \end{pmatrix}.$$

[1] Valiant actually uses this word.

In addition each junction has external connections only via nodes 1, 4 and not 2, 3. The construction of a track and an interchange are left to the images below, and the technical details of verifying this construction are left to Valiant.[2]

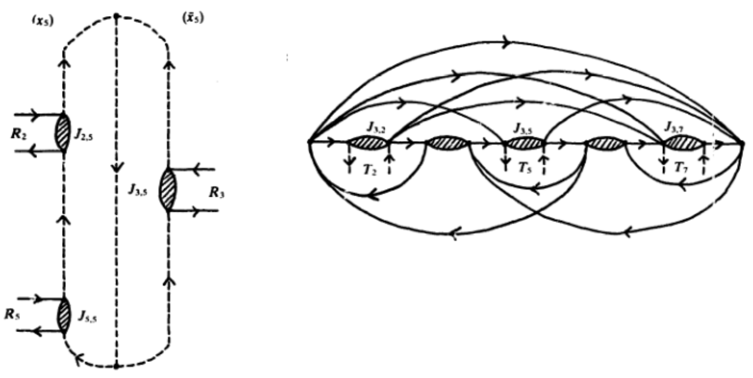

Figure 1: A track and an interchange.

□

Lemma 7. *The proof that SAT is NP-complete can be modified to preserve the number of accepting computations.*

Lemma 8. *There is a polynomial time in m, n reduction h from $n \times n$ matrices with entries in $\{0, \dots, m\}$ to matrices (of polynomial size) with entries in $\{0, 1\}$ such that*

$$\operatorname{perm} h(A) = \operatorname{perm} A$$

for all matrices A.

Proof. We replace every edge of positive weight k with a subgraph with k self-loops. The diagram for $k = 5$ is below.

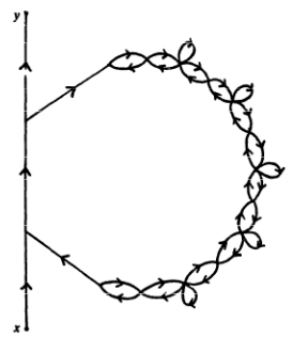

Fig. 2.

Figure 2: Edge Replacement

□

Proposition 9. *Computing the permanent of a $(0, 1)$-matrix mod $r < dn \log n$ is #P-hard for some value of d.*

Proof. Note that if the matrix A has $|A_{ij} \leq| M$, the permanent has size at most $M^n n!$. Therefore we compute the permanent mod p_i for a set of distinct primes $\{p_1, \dots, p_k\}$ whose product is at least $2M^n n!$. Then for some constant d' it suffices to have $p_i < d' n \log(Mn)$ (Hardy and Wright is cited here). By Lemma 8, we can transform each matrix $A \mod p_i$ into a $(0, 1)$-matrix with the same permanent. The result follows from Lemmas 6, 7. □

[2]He says this proof is adapted from earlier work, and then he refers to this paper in another paper of his for this result.

Theorem 10. *Computing the permanent of a $(0,1)$-matrix is $\#P$-complete.*

Proof. By Proposition 9, computing the permanent is $\#P$-hard. On the other hand, deciding whether a vertex-disjoint cycle cover of a graph exists is in NP, so computing the permanent is in $\#P$. $\square$

Theorem 11. *For any K not a power of 2, computing the permannent mod K of a $(0,1)$-matrix is UP-hard. Here UP is the class of problems with a unique accepting computation on a poly-time NTM.*

Proof. In the construction of Lemma 6, the permanent is either 0 or a power of 4. Thus it is divisible by K iff M does not accept x. $\square$

3 A Positive Result

We conclude with a positive result.

Theorem 12. *The permanent of an $n \times n$ matrix mod 2^k can be computed in time $O(n^{4k-3})$ for $k \geq 2$ and in time $O(n^2.81)$ for $k = 1$.*

Proof. Consider two types of matrices: Those with h pairs of equal rows and those with all zero entries $g+1$ below the diagonal. Let F_h denote the complexity of computing the permanent of the first type and G_g analogously for the second type.

To compute the permanent of the second type of matrix, we simply compute left-to-right with a dynamic programming algorithm. For each $i = 1, \dots, n-g$, we compute for each $I \in \binom{[i+g]}{i}$ the permanent of rows I and the first i columns. At each stage, there are fewer than n^g submatrices, so we reach the $g+1$ full columns in time $O(n^{g+1})$. Then the last $g+1$ columns can be evaluated in time $O(n^{g+1})$, so $G_g = O(n^{g+1})$.

For the first type, we use a modified version of Gaussian elimination. If we replace A_i with $A_i + A_j$, then we have $\operatorname{perm} A' = \operatorname{perm} A + \operatorname{perm} A''$, where A'' is obtained by replacing row i with row j. Then note that because A'' has two equal rows, its permanent is even. Similarly, we obtain $F_k = 0$. Now we simply eliminate the bottom $n - 2h$ rows so that the matrix is now of the second type. This consists of $O(n^2)$ elimination except with another additive term in each step. Thus we have

$$F_h = O(n^3) + O(n^2)F_{h+1} + G_{2h},$$

which gives us $F_0 = n^{4k-3}$ for $k \geq 2$. For $k = 1$, we obtain $O(n^3)$ Gaussian elimination. $\square$

4 Other Permanent Results

We give a collection of other results relating to the complexity of computing the permanent.

Theorem 13 (Allender, 1997)**.** *The permanent cannot be computed in uniform TC^0.*

Theorem 14 (Jerrum-Sinclair, 1989)**.** *The permanent of a $(0,1)$-matrix has a fully polynomial randomized approximation scheme.*

In 1979, Valiant showed that any polynomial with a formula of size s is the projection of an $(s+2) \times (s+2)$ determinant.[3]. The best lower bound for the formula size of the permanent is quadratic (obtained as a lower bound for the size of the matrix B giving $\operatorname{perm}_n = \det B$).

References

[1] Valiant, L.G. *The Complexity of Computing the Permanent.* Theoretical Computer Science, vol 8, pp. 189-201 (1979).

[3] This was later improved to $s+1$