Texts in Computational Complexity: A proof of the $\#\mathcal{P}$-completeness of the permanent

Oded Goldreich
Department of Computer Science and Applied Mathematics
Weizmann Institute of Science, Rehovot, Israel.

November 12, 2005

Main Text

We present the following result of Valiant [2].

Theorem 1 *Counting the number of perfect matchings in a bipartite graph is $\#\mathcal{P}$-complete.*

Equivalently (see Exercise 5), the problem of computing the permanent of matrices with 0/1-entries is $\#\mathcal{P}$-complete. Recall that the permanent of an n-by-n matrix $M = (m_{i,j})$, denoted $\texttt{perm}(M)$, equals the sum over all permutations π of $[n]$ of the products $\prod_{i=1}^{n} m_{i,\pi(i)}$. Theorem 1 is proven by composing two (many-to-one) reductions, asserted in Propositions 2 and 3, respectively. Needless to say, the resulting reduction is not parsimonious.

Proposition 2 *The counting problem of* 3SAT (i.e., $\#R_{\texttt{3SAT}}$) *is reducible to computing the permanent of integer matrices. Furthermore, there exists an* even *integer $c > 0$ and a finite set of integers I such that, on input a 3CNF formula ϕ, the reduction produces an integer matrix with entries in I and a permanent value that equals $c^m \cdot \#R_{\texttt{3SAT}}(\phi)$, where m denotes the number of clauses in ϕ.*

The original proof of Proposition 2 uses $c = 2^{10}$ and $I = \{-1, 0, 1, 2, 3\}$. It follows that, for every integer $n > 1$ that is relatively prime to c, computing the permanent modulo n is NP-hard (see Exercise 6, which also uses Theorem 4). Thus, using the case of $c = 2^{10}$, this means that computing the permanent modulo n is NP-hard for any odd $n > 1$. In contrast, computing the permanent modulo 2 (which is equivalent to computing the determinant modulo 2) is easy (i.e., can be done in polynomial-time and even in $\mathcal{NC}$). Thus, assuming $\mathcal{NP} \not\subseteq \mathcal{BPP}$, Proposition 2 cannot hold for an odd c. We also note that, assuming $\mathcal{P} \neq \mathcal{NP}$, Proposition 2 cannot possibly hold for a set I containing only non-negative integers (see Exercise 7).

Proposition 3 *Computing the permanent of integer matrices is reducible to computing the permanent of 0/1-matrices. Furthermore, the reduction transforms an integer matrix A into a 0/1-matrix A'' such that the permanent of A can be easily computed from A and the permanent of A''.*

Proof of Proposition 2: We will use the correspondence between the permanent of a matrix A and the sum of the weights of the cycle covers of the weighted directed graph represented by the matrix A. A cycle cover of a graph is a collection of simple[1] *vertex-disjoint* directed cycles that

[1] Here a simple cycle is a strongly connected directed graph in which each vertex has a single incoming (resp., outgoing) edge. In particular, self-loops are allowed.

covers all the graph's vertices, and its weight is the product of the weights of the corresponding edges. The SWCC of a weighted directed graph is the sum of the weights of all its cycle covers.

Given a 3CNF formula ϕ, we construct a directed weighted graph G_ϕ such that the SWCC of G_ϕ equals equals $c^m \cdot \#R_{\mathtt{3SAT}}(\phi)$, where c is a universal constant and m denotes the number of clauses in ϕ. We may assume, without loss of generality, that each clause of ϕ has exactly three variables (which are not necessarily distinct).

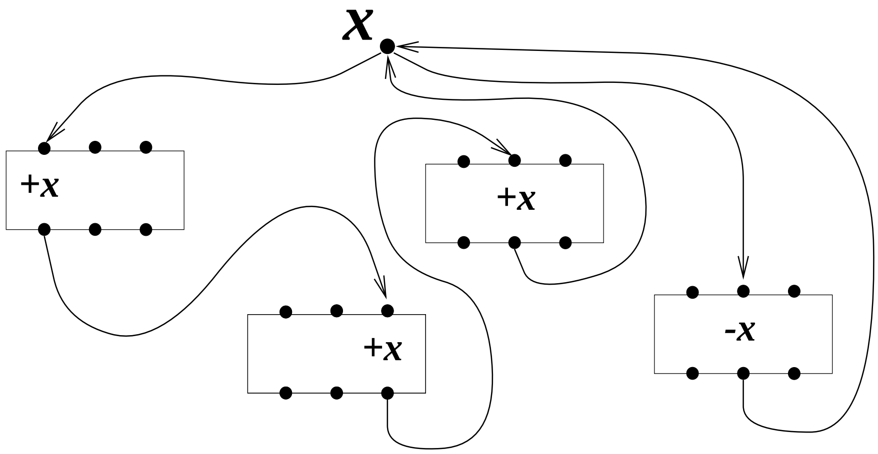
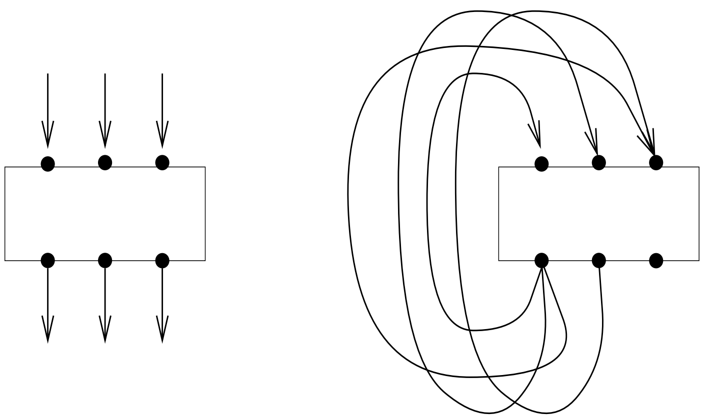


Figure 1: Tracks connecting gadgets for the reduction to cycle cover.

We start with a high-level description (of the construction) that refers to (clause) gadgets, each containing some internal vertices and (weighted) edges, which are unspecified at this point. In addition, each gadget has three pairs of designated vertices, one pair per each literal appearing in the clause, when one vertex in the pair is designated as an entry vertex and the other as an exit vertex. The graph G_ϕ consists of m such gadgets, one per each clause (of ϕ), and n auxiliary vertices, one per each variable (of ϕ), as well as some *additional directed edges*, each having weight 1. Specifically, for each variable, we introduce two tracks, one per each of the possible literals of this variable. The track associated with a literal consists of directed edges that form a simple "cycle" passing through the corresponding vertex as well as through the designated vertices that correspond to the occurrences of this literal in the various clauses. Specifically, for each such occurrence, the track enters the corresponding clause gadget at the entry-vertex corresponding to this literal and exits at the corresponding exit-vertex. (If a literal does not appear in ϕ then the corresponding track is a self-loop on the corresponding variable.) See Figure 1 showing the two tracks of a variable x that occurs positively in three clauses and negatively in one clause. The entry-vertices (resp., exit-vertices) are drawn on the top (resp., bottom) part of each gadget.

For the purpose of stating the desired properties of the clause gadget, we augment the gadget by nine external edges (of weight 1), one per each pair of (not necessarily matching) entry and exit vertices such that the edge goes from the exit-vertex to the entry-vertex (see Figure 2). The three edges that link the designated pairs of vertices that correspond to the three literals are called nice. We say that a collection of edges C (e.g., a collection of cycles) uses the external edges S if the intersection of C with the set of the (nine) external edges equals S. We postulate the following three properties of the clause gadget.

1. The sum of the weights of all cycle covers (of the gadget) that do not use any external edge (i.e., use the empty set of external edges) equals zero.



On the left is a gadget with the track edges adjacent to it (as in the real construction). On the right is a gadget and four out of the nine external edges (two of which are nice) used in the analysis.

Figure 2: External edges for the analysis of the clause gadget

2. Let $V(S)$ denote the set of vertices incident to S, and say that S is nice if it is non-empty and the vertices in $V(G)$ can be perfectly matched using nice edges.[2] Then, there exists a constant c (indeed the one postulated in the proposition's claim) such that, for any nice set S, the sum of the weights of all cycle covers that use the external edges S equals c.

3. For any non-nice set S of external edges, the sum of the weights of all cycle covers that use the external edges S equals zero.

Note that the foregoing three cases exhaust all the possible ones, and that the set of external edges used by a cycle cover must be a matching (i.e., these edges are vertex disjoint). Using the foregoing conditions, it follows that each satisfying assignment of ϕ contributes exactly c^m to the SWCC of G_ϕ (see Exercise 8). It follows that the SWCC of G_ϕ equals $c^m \cdot \#R_{\mathtt{3SAT}}(\phi)$.

Having established the validity of the abstract reduction, we turn to the implementation of the clause gadget. The first implementation is a *Deus ex Machina*, with a corresponding adjacency matrix depicted in Figure 3. Its validity (for the value $c = 12$) can be verified by computing the permanent of the corresponding sub-matrices (see analogous analysis in Exercise 10).

A more structured implementation of the clause gadget is depicted in Figure 4, which refers to a (hexagon) box to be implemented later. The box contains several vertices and weighted edges, but only two of these vertices, called terminals, are connected to the outside (and are shown in Figure 4). The clause gadget consists of five copies of this box, where three copies are designated for the three literals of the clause (and are marked LB1, LB2, and LB3), as well as additional vertices and edges shown in Figure 4. In particular, the clause gadget contains the six aforementioned designated vertices (i.e., a pair of entry and exit vertices per each literal), two additional vertices (shown at the two extremes of the figure), and some edges (all having weight 1). Each designated vertex has a self-loop, and is incident to a single additional edge that is outgoing (resp., incoming) in case the

[2] Clearly, any non-empty set of nice edges is a nice set. Thus, a singleton set is nice if and only if the corresponding edge is nice. The set S of three external edges is nice, because $V(S)$ has a perfect matching using all three nice edges. Thus, the notion of nice sets is "non-trivial" only for sets of two edges. Such a set is nice if and only if $V(S)$ consists of a two pairs of corresponding designated vertices.

The gadget uses eight vertices, where the first six are the designated vertices. The entry-vertex (resp., exit-vertex) associated with the i^{th} literal is numbered i (resp., $i+3$). The corresponding adjacency matrix follows.

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

Note that the edge $3 \to 6$ can be contracted, but the resulting 7-vertex graph will not be consistent with our (inessentially stringent) definition of a gadget by which the six designated vertices should be distinct.

Figure 3: A Deus ex Machina clause gadget for the reduction to cycle cover.

vertex is an entry-vertex (resp., exit-vertex) of the gadget. The two terminals of each box that is associated with some literal are connected to the corresponding pair of designated vertices (e.g., the outgoing edge of `entry1` is incident at the right terminal of the box `LB1`). Note that the five boxes reside on a directed path (going from left to right), and the only edges going in the opposite direction are those drawn below this path.

In continuation to the foregoing, we wish to state the desired properties of the box. Again, we do so by considering the augmentation of the box by external edges (of weight 1) incident at the specified vertices. In this case (see Figure 5), we have a pair of anti-parallel edges connecting the two terminals of the box as well as two self-loops (one on each terminal). We postulate the following three properties of the box.

1. The sum of the weights of all cycle covers (of the box) that do not use any external edge equals zero.

2. There exists a constant b (in our case $b = 4$) such that, for each of the two anti-parallel edges, the sum of the weights of all cycle covers that use this edge equals b.

3. For any (non-empty) set S of the self-loops, the sum of the weights of all cycle covers (of the box) that use S equals zero.

Note that the foregoing three cases exhaust all the possible ones. It can be shown that the conditions regarding the box imply that the construction presented in Figure 4 satisfies the conditions that were postulated for the clause gadget (see Exercise 9). Specifically, we have $c = b^5$. As for box

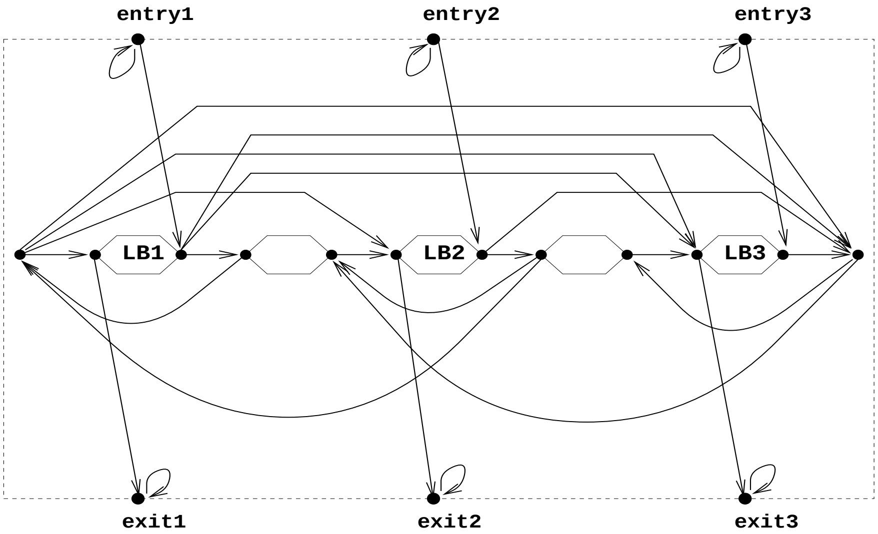
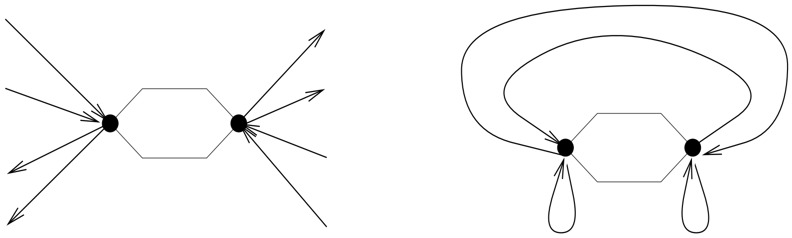


Figure 4: A structured clause gadget for the reduction to cycle cover.



On the left is a box with potential edges adjacent to it (as in the gadget construction). On the right is a box and the four external edges used in the analysis.

Figure 5: External edges for the analysis of the box

itself, a smaller *Deus ex Machina* is provided by the following 4-by-4 adjacency matrix

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

where the two terminals correspond to the first and the fourth vertices. Its validity (for the value $b = 4$) can be verified by computing the permanent of the corresponding sub-matrices (see Exercise 10). ■

Proof of Proposition 3: The proof proceeds in two steps. In the first step we show that computing the permanent of integer matrices is reducible to computing the permanent of non-negative matrices. This reduction proceeds as follows. Given an n-by-n integer matrix $A = (a_{i,j})_{i,j}$, let $\|A\|_\infty = \max_{i,j}(|a_{i,j}|)$ and $Q_A = 2(n!) \cdot \|A\|_\infty^n + 1$. The reduction constructs the matrix $A' = (a_{i,j} \bmod Q_A)_{i,j}$ (i.e., the entries of A' are in $\{0, 1, ..., Q_A - 1\}$), and outputs $v \stackrel{\text{def}}{=} \texttt{perm}(A') \bmod Q_A$ if $v < Q_A/2$ and $-(Q_A - v)$ otherwise. The key observation is that $\texttt{perm}(A) \equiv \texttt{perm}(A') \pmod{Q_A}$

while $|\texttt{perm}(A)| \leq (n!) \cdot \|A\|_\infty^n < Q_A/2$. Thus, $\texttt{perm}(A') \bmod Q_A$ (which is in $\{0,1,...,Q_A-1\}$) determines $\texttt{perm}(A)$. We note that $\texttt{perm}(A')$ is likely to be much larger than $Q_A > |\texttt{perm}(A)|$; it is merely that $\texttt{perm}(A')$ and $\texttt{perm}(A)$ are equivalent modulo Q_A.

In the second step we show that computing the permanent of non-negative matrices is reducible to computing the permanent of 0/1-matrices. In this reduction, we view the computation of the permanent as the computation of the sum of the weights of the cycle covers (SWCC) of the corresponding weighted directed graph (see proof of Proposition 2). Thus, we reduce the computation of the SWCC of directed graphs *with non-negative weights* to the computation of the SWCC of *unweighted directed graphs with no parallel edges* (which correspond to 0/1-matrices). The reduction is via local replacements that preserve the value of the SWCC. These local replacements combined the following two local replacements (which preserve the SWCC):

1. Replacing an edge of weight $w = w_1 \cdots w_t$ by a path of length t (i.e., $t-1$ internal nodes) with the corresponding weights $w_1, ..., w_t$, and self-loops (with weight 1) on all internal nodes.

   Note that a cycle-cover that uses the original edge corresponds to a cycle-cover that uses the entire path, whereas a cycle-cover that does not use the original edge corresponds to a cycle-cover that uses all the self-loops.

2. Replacing an edge of weight $w = w_1 + \cdots + w_t$ by t parallel 2-edge paths such that the first edge on the i^{th} path has weight w_i, the second edges has weight 1, and the intermediate node has a self-loop (with weight 1). (Paths of length two are used because parallel edges are not allowed.)

   Note that a cycle-cover that uses the original edge corresponds to a collection of cycle-covers that use one out of the t paths (and the self-loops of all other intermediate nodes), whereas a cycle-cover that does not use the original edge corresponds to a cycle-cover that uses all the self-loops.

In particular, writing the positive integer w, having binary expansion $\sigma_{|w|-1} \cdots \sigma_0$, as $\sum_{i:\sigma_i=1}(1+1)^i$, we may apply the additive replacement (for the sum over $\{i : \sigma_i = 1\}$), next the product replacement (for each 2^i), and finally the additive replacement (for $1+1$). Applying this process to the matrix A' obtained in the first step, we efficiently obtain a matrix A'' with 0/1-entries such that $\texttt{perm}(A') = \texttt{perm}(A'')$. Combining the two reductions (steps), the proposition follows. ■

Notes

The counting class $\#\mathcal{P}$ was introduced by Valiant [2], who proved that computing the permanent of 0/1-matrices is $\#\mathcal{P}$-complete (i.e., Theorem 1).

Our presentation of Theorem 1 is based both on Valiant's paper [2] and on subsequent studies (most notably [1]). Specifically, the high-level structure of the reduction presented in Proposition 2 as well as the "structured" design of the clause gadget is taken from [2], whereas the Deus Ex Machina gadget presented in Figure 3 is based on [1]. The proof of Proposition 3 is also based on [1] (with some variants). Turning back to the design of clause gadgets we regret not being able to cite and/or use a systematic study of this design problem.

On the hardness of unique solution problems. In the main text, we refer to the following version of the Valiant-Vazirani Theorem, which is stated next. For a binary relation R, we denote $R(x) = \{y : (x,y) \in R\}$, and say that x has a unique solution $|R(x)| = 1$. We say that a many-to-one reduction f of R' to R is parsimonious if for every x it holds that $|R(x)| = |R'(f(x))|$. We denote by $\mathcal{PC}$ the class of search problems that correspond to $\mathcal{NP}$; that is, $R \in \mathcal{PC}$ if there exists a polynomial p such that for every $(x, y) \in R$ it holds that $|y| \leq p(|x|)$ and membership in R can be decided in polynomial-time.

Theorem 4 *Let $R \in \mathcal{PC}$ and suppose that every search problem in $\mathcal{PC}$ is parsimoniously reducible to R. Then solving the search problem of R* (resp., deciding membership in $S_R = \{x : |R(x)| \geq 1\}$) *is reducible in probabilistic polynomial-time to finding unique solutions for R* (resp., the promise problem $(\mathtt{US}_R, \overline{S}_R)$, where $\mathtt{US}_R = \{x : |R(x)| = 1\}$ and $\overline{S}_R = \{x : |R(x)| = 0\}$).

Exercise 5 (computing the permanent of integer matrices) Prove that computing the permanent of matrices with 0/1-entries is computationally equivalent to computing the number of perfect matchings in bipartite graphs.
(Hint: Given a bipartite graph $G = ((X, Y), E)$, consider the matrix M representing the edges between X and Y (i.e., the (i, j)-entry in M is 1 if the i^{th} vertex of X is connected to the j^{th} entry of Y), and note that only perfect matchings in G contribute to the permanent of M.)

Exercise 6 (computing the permanent modulo 3) Combining Proposition 2 and Theorem 4, prove that for every integer $n > 1$ that is relatively prime to c, computing the permanent modulo n is NP-hard under randomized reductions. Since Proposition 2 holds for $c = 2^{10}$, hardness holds for every odd integer $n > 1$.

Guideline: Applying the reduction of Proposition 2 to the promise problem of deciding whether a 3CNF formula has a unique satisfiable assignment or is unsatisfiable. Use the fact that n does not divide any power of c.

Exercise 7 (negative values in Proposition 2) Assuming $\mathcal{P} \neq \mathcal{NP}$, prove that Proposition 2 cannot hold for a set I containing only non-negative integers. Note that the claim holds even if the set I is not finite (and even if I is the set of all non-negative integers).

Guideline: A reduction as in Proposition 2 provides a Karp-reduction of `3SAT` to deciding whether the permanent of a matrix with entries in I is non-zero. Note that the permanent of a *non-negative* matrix is non-zero if and only if the corresponding bipartite graph has a perfect matching.

Exercise 8 (high-level analysis of the permanent reduction) Establish the correctness of the high-level reduction presented in the proof of Proposition 2. That is, show that if the clause gadget satisfy the three conditions postulated in the said proof, then each satisfying assignment of ϕ contributes exactly c^m to the SWCC of G_ϕ and unsatisfying assignments have no contribution.

Guideline: Cluster the cycle covers of G_ϕ according to the set of track edges that they use (i.e., the edges of the cycle cover that belong to the various tracks). (Note the correspondence between these edges and the external edges used in the definition of the gadget's properties.) Using the postulated conditions (regarding the clause gadget) prove that, for each such set T of track edges, if the sum of the weights of all cycle covers that use the track edges T is non-zero then the following hold:

1. The intersection of T with the set of track edges incident at each specific clause gadget is non-empty. Furthermore, if this set contains an incoming edge (resp., outgoing edge) of some entry-vertex (resp., exit-vertex) then it also contains an outgoing edge (resp., incoming edge) of the corresponding exit-vertex (resp., entry-vertex).
2. If T contains an edge that belongs to some track then it contains all edges of this track. It follows that, for each variable x, the set T contains the edges of a single track associated with x.

3. The tracks "picked" by T correspond to a single truth assignment to the variables of ϕ, and this assignment satisfies ϕ (because, for each clause, T contains an external edges that correspond to a literal that satisfies this clause).

It follows that each satisfying assignment of ϕ contributes exactly c^m to the SWCC of G_ϕ.

Exercise 9 (analysis of the implementation of the clause gadget) Establish the correctness of the implementation of the clause gadget presented in the proof of Proposition 2. That is, show that if the box satisfy the three conditions postulated in the said proof, then the clause gadget of Figure 4 satisfies the conditions postulated for it.

Guideline: Cluster the cycle covers of a gadget according to the set of non-box edges that they use, where non-box edges are the edges shown in Figure 4. Using the postulated conditions (regarding the box) prove that, for each set S of non-box edges, if the sum of the weights of all cycle covers that use the non-box edges S is non-zero then the following hold:

1. The intersection of S with the set of edges incident at each box must contain two (non-selfloop) edges, one incident at each of the box's terminals. Needless to say, one edge is incoming and the other outgoing. Referring to the six edges that connects one of the six designated vertices (of the gadget) with the corresponding box terminals as connectives, note that if S contains a connective incident at the terminal of some box then it must also contain the connective incident at the other terminal. In such a case, we say that this box is picked by S,
2. Each of the three (literal-designated) boxes that is not picked by S is "traversed" from left to right (i.e., the cycle cover contains an incoming edge of the left terminal and an outgoing edge of the right terminal). Thus, the set S must contain a connective, because otherwise no directed cycle may cover the leftmost vertex shown in Figure 4. That is, S must pick some box.
3. The set S is fully determined by the non-empty set of boxes that it picks.

The postulated properties of the clause gadget follow, with $c = b^5$.

Exercise 10 (analysis of the design of a box for the clause gadget) Prove that the 4-by-4 matrix presented in Eq. (1) satisfies the properties postulated for the "box" used in the second part of the proof of Proposition 2. In particular:

1. Show a correspondence between the conditions required of the box and conditions regarding the value of the permanent of certain sub-matrices of the adjacency matrix of the graph.

   (Hint: For example, show that the first condition correspond to requiring that the value of the permanent of the entire matrix equals zero. The second condition refers to sub-matrices obtained by omitting either the first row and fourth column or the fourth row and first column.)

2. Verify that the matrix in Eq. (1) satisfies the aforementioned conditions (regarding the value of the permanent of certain sub-matrices).

Prove that no 3-by-3 matrix (and thus also no 2-by-2 matrix) can satisfy the aforementioned conditions.

References

[1] A. Ben-Dor and S. Halevi. In *2nd Israel Symp. on Theory of Computing and Systems*, IEEE Computer Society Press, pages 108-117, 1993.

[2] L.G. Valiant. The Complexity of Computing the Permanent. *Theoretical Computer Science*, Vol. 8, pages 189–201, 1979.