THE PERMANENT, GRAPH GADGETS AND COUNTING SOLUTIONS FOR CERTAIN TYPES OF PLANAR FORMULAS

CHRISTIAN SCHRIDDE

Abstract. In this paper, we build on the idea of Valiant [13] and Ben-Dor/Halevi [3], that is, to count the number of satisfying solutions of a boolean formula via computing the permanent of a specially constructed matrix. We show that the Desnanot-Jacobi identity (DJi) prevents Valiant’s original approach to achieve a parsimonious reduction to the permanent over a field of characteristic two. As the next step, since the computation of the permanent is \#P-complete, we make use of the equality of the permanent and the number of perfect matchings in an unweighted graph’s bipartite double cover. Whenever this bipartite double cover (BDC) is planar, the number of perfect matchings can be counted in polynomial time using Kasteleyn’s algorithm [7]. To enforce planarity of the BDC, we replace Valiant’s original gadgets with new gadgets and describe what properties these gadgets must have. We show that the property of circular planarity plays a crucial role to find the correct gadgets for a counting problem. To circumvent the DJi-barrier, we switch over to fields \( \mathbb{Z}/p\mathbb{Z} \), for a prime \( p > 2 \).

With this approach we are able to count the number of solutions for \textsc{forest-3sat} formulas in randomized polynomial time. Finally, we present a conjecture that states which kind of generalized gadgets can not be found, since otherwise one could prove \( \text{RP} = \text{NP} \). The conjecture establishes a relationship between the determinants of the minors of a graph \( G \)’s adjacency matrix and the circular planar structure of \( G \)’s BDC regarding a given set of nodes.

1. Introduction

Given a \((n \times n)\)-matrix \( A = a_{ij}, 1 \leq i, j \leq n \), its determinant is defined as

\[
\det(A) = \sum_{\sigma \in S_n} \text{sgn}(\sigma) \prod_{i=1}^{n} a_{i,\sigma_i},
\]

and the permanent is defined as

\[
\text{perm}(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} a_{i,\sigma_i}.
\]

The difference in notation is minimal. The determinant considers the sign of the permutations \( \sigma \) whereof the permanent does not. This little difference is responsible for the huge gap between the necessary resources that are needed in order to compute the actual value of these functions. The determinant can be evaluated in time that is polynomial in the dimension \( n \), whereas the permanent is only computable in time that is exponential in \( n \), e.g., using the \( O(n^2^n) \) algorithm of Ryser. In the year 1979 Leslie Valiant [13, 14] showed that the permanent is even \#P-complete and he shows the same for any computation of \( \text{perm}(A) \mod p \), \( p > 2 \). In his proof, Valiant created a graph \( G \) from a given boolean formula \( \Phi \) such that the number of \( \Phi \)’s satisfying assignments can be obtained from the permanent of \( G \)’s adjacency matrix by using Proposition 1.3.

Definition 1.1 (Cyclic-cover). A cyclic-cover of a weighted directed graph \( G = (V, E) \) is a subset \( R \subseteq E \) that forms a collection of node-disjoint directed cycles that cover all the nodes of \( G \).

Definition 1.2 (Weight). The weight of a cyclic cover \( R \), denoted by \( W(R) \), is the product of the weights of the edges in \( R \).

Proposition 1.3. Given a graph \( G \) and its adjacent matrix \( A \). Let \( R \) be the set of all cyclic covers of \( G \), then \( \text{perm}(A) = \sum_{R \in R} W(R) \).

The essential ingredients for Valiant’s proof are graph gadgets. These gadgets ensure that the weights of the cyclic covers depend on whether they are part of a satisfying assignment or not. In a subsequent work, Ben-Dor and Halevi [3] presented another, simpler proof that comes along with another type of graph gadget.

Contribution. In this paper, we extend the idea of Valiant and Ben-Dor/Halevi by means of defining new gadgets. In Valiant’s approach, his gadgets are designed to make the permanent of the graph’s adjacency matrix contain the number of satisfying solutions of a boolean formula \( \Phi \). Since the permanent is equal to the determinant
in fields of characteristic two, we first show, that gadgets that would allow to count the number of satisfying solutions (at least of formulas from Unique−SAT) can not be found because of the Desnanot-Jacobi identity.

Since despite the case modulo 2, the permanent can not be evaluated in general, we then turn our attention to the relationship of the permanent and the number of perfect matchings in a graph’s bipartite double cover. We search for gadgets that have the potential to determine satisfiability over \( \mathbb{Z}/p\mathbb{Z} \), for \( p > 2 \), whenever the bipartite double cover is planar. We constructed a new XOR-gadget\(^1\) that has two mandatory properties:

- It is an unweighted graph, i.e. its adjacency matrix is a \((0/1)\)-matrix. This is in contrast to the XOR-gadget of Valiant (and the CLAUSE-gadget of Ben-Dor and Halevi) which contains values from \{-1, 0, 1, 2, 3\}. Having a \((0/1)\)-graph is mandatory to apply Proposition\(^2\).
- It fulfills a weakened set of permanent rule\(^3\) over a small finite group rather than the integers.

In [\cite{13}, Lemma 3.3] Valiant showed how to transform a matrix \( A \) with positive integers into another matrix \( A' \) that only has values from \{0, 1\} and \( \text{perm}(A) = \text{perm}(A') \). Since his XOR-gadget does contain edges with weight \(-1\), he could not apply the transformation. However, Ben-Dor and Halevi\(^4\) showed the following reduction chain:

\[ \text{IntPerm} \propto \text{NoNegPerm} \propto \text{2PowersPerm} \propto 0/1-\text{Perm} \]

whereof the permanent value is preserved along the chain. The drawback of their approach is, that this reduction chain increases the number of nodes heavily, which reduces the chances to reach planarity, which is mandatory to apply the Kasteleyn’s counting algorithm. They did not consider to look at the planarity of their resulting graph nor they tried to further work with it.

Although so far our new XOR-gadget already gives the theoretical potential to determine satisfiability, it turns out that most of the time Kasteleyn’s algorithm can not be applied\(^5\). The reason is, that the BDC is almost always non-planar. Consequently, we tried to find methods that increases the chance that the BDC will be planar. Even though the bipartite double cover operation changes planarity in both ways, it turns out that planar graphs with low connectivity do more often have a planar bipartite covers. Following this fact, we introduce a method to make \( G \) planar by the help of two further new XOR-gadgets and Lichtenstein’s reduction\(^6\) to pn-planar formulas.

Finally, one notices that the fraction of planar BDCs is still negligible. To circumvent the problem of getting a planar bipartite double cover only accidentally, in the last sections of the paper we turn towards generalization. We describe what gadgets have to be used therewith the bipartite double cover will be always planar if the input is a pn-planar formula. We then show which subclasses of these formulas can be counted with that approach, since appropriate gadgets exists. And, even more important, what prevents the generalized approach from counting solutions of arbitrary pn-planar formulas.

At the end, we formulate a conjecture which states what kind of gadgets could not be found since otherwise one could prove \( \text{RP} = \text{NP} \). The conjecture establishes a relationship between the determinants of the minors of a graph \( G \)'s adjacency matrix and the circular planar structure of \( G \)'s bipartite double cover regarding a certain set of nodes.

1.1. Preliminaries and Definitions.

**Definition 1.4** (Decision problem: SAT). Given a boolean formula, determine its satisfiability. ▶

**Definition 1.5** (Decision problem: Unique−SAT). Given a boolean formula that has at most one satisfying assignment, determine its satisfiability. ▶

**Definition 1.6** (Decision problem: kSAT). Given a boolean formula that is in CNF with at most \( k \) literals per clause (kCNF), determine its satisfiability. ▶

**Definition 1.7** (Decision problem: Unique−kSAT). Given a boolean formula in kCNF that has at most \( k \) satisfying assignment, determine its satisfiability. ▶

**Definition 1.8** (Discrete counting problem: \( \#_p \) kSAT). Given a boolean formula from kSAT, compute the number of satisfying solutions modulo \( p \). ▶

For \( p = 2 \), the \( \#_p \) kSAT problem is equivalent to its parity problem \( \oplus k\text{SAT} \).

**Theorem 1.9** (Valiant-Vazarani [\cite{17}]). If there is a polynomial-time algorithm solving the decision problem Unique−SAT than \( \text{NP} = \text{RP} \).

\(^1\)That is the name of one of the gadgets Valiant uses in his proof.

\(^2\)Valiant introduces a set of permanent rules that an XOR-gadget must fulfill in order to make his proof work.

\(^3\)At least, we draw this conclusion from our implementation, that shows that for large formulas the bipartite double cover is almost always non-planar.

\(^4\)A certain class of boolean formulas that is also NP-complete.
The class Unique-SAT handles boolean formulas that have at most a single satisfying assignment. Surprisingly, this does not help in general to decide their satisfiability. In the proof of Theorem 1.12, Valiant and Vaziran showed, that one can transform an arbitrary boolean formula Φ with n variables, via an efficient computable function f, into another formula such that

\[ \Phi \text{ is satisfiable } \Rightarrow \ Pr[f(\Phi) \text{ has a unique assignment}] \geq \frac{1}{8n} \]

\[ \Phi \text{ is not satisfiable } \Rightarrow \ Pr[f(\Phi) \text{ is satisfiable}] = 0 \]

Formulas from Unique-SAT can represent problems from number theory or cryptology, e.g., it is possible to express the factorization problem as the task to find the solution of a formula that only has one assignment and that involves the binary representation of the two factors.

Definition 1.10 (Incidence graph of a formula). Let Φ be a CNF formula with variables \( v_1, \ldots, v_n \) and clauses \( c_1, \ldots, c_m \). Then the incidence graph \( G_\Phi \) is defined as

\[
G_\Phi = (V, E) \\
V = \{v_1, \ldots, v_n, c_1, \ldots, c_m\} \\
E = \{(v_i, c_j)|v_i \text{ or } \neg v_i \text{ occurs in } c_j\}
\]

Definition 1.11 (PLANAR-3SAT). Given a 3CNF with a planar incidence graph (called “a planar 3CNF formula”), then PLANAR-3SAT is the decision problem to decide its satisfiability. ▽

Definition 1.12 (POSITIVE-NEGATIVE-PLANAR-3SAT). Let Φ be a 3CNF formula with variables \( v_1, \ldots, v_n \) and clauses \( c_1, \ldots, c_m \). The graph \( G_\Phi \) is defined as

\[
G_\Phi = (V, E) \\
V = \{v_1, \ldots, v_n, c_1, \ldots, c_m\} \\
E = E_1 \cup E_2 \\
E_1 = \{(v_i, v_{i+1})|1 \leq i \leq n-1\} \cup (v_n, v_1) \\
E_2 = \{(v_i, c_j)|v_i \text{ or } \neg v_i \text{ occur in } c_j\}
\]

If \( G_\Phi \) is planar, then \( \Phi \) is a planar 3CNF formula. The edge set \( E_1 \) forms a cycle, i.e., it divides the plane into two faces, the inner and the outer face. The restriction in POSITIVE-NEGATIVE-PLANAR-3SAT or short PN-PLANAR-3SAT is that, if \( c_i \) and \( c_j \) contain the same variable but in a negated form, then \( c_i \) and \( c_j \) are in different faces of the graph. PN-PLANAR-3SAT is the decision problem to decide if \( \Phi \) is satisfiable. ▽

The graph from Definition 1.11 is bipartite since it is a mapping between the two vertex sets \( \{v_1, \ldots, v_n\} \) and \( \{c_1, \ldots, c_m\} \), whereas the graph from Definition 1.11 is not bipartite directly, but can be converted into a bipartite graph by removing the edges from \( E_1 \). In his paper, Lichtenstein describes a reduction algorithm, that transforms an arbitrary formula \( \Phi \) to another formula \( \Phi' \), such that the graph \( G_{\Phi'} \) (pn)-planar. This transformation may change the number of satisfying solutions of \( \Phi \), but not \( \Phi \)'s satisfiability.

Definition 1.13. PN-PLANAR-kSAT uses Definition 1.12 but with a kCNF as the input. ▽

Definition 1.14 (FOREST-3SAT). Given a 3CNF formula, who incidence graph is a forest. The FOREST-3SAT is the decision problem to decide if \( \Phi \) is satisfiable. ▽

Surprisingly, there are special classes of satisfiability/counting problems that could be solved in polynomial time. In 2010, Marko Samer and Stefan Szeider 12 already showed that #FOREST-3SAT can be computed in polynomial time using a approach that has nothing to do with graph gadgets. Dan Roth 11 describes aCYCLIC-SAT, which is a special case of monotone 2CNF formulas and showed, that its counting version (#aCYCLIC-2MON-SAT) is polynomial time solvable. NESTED-SAT is handled by Knuth 8 and Kratochvıl and Krivnek 9 and it is shown that NESTED-SAT is actual in P. So far it is not known if the counting version is also efficiently solvable.

Further class of counting algorithms build on planar graphs. The reason is, that in planar graphs some problems can be solved more efficiently than in non-planar graphs. One such problem is to count the number of perfect matchings. Fisher, Kasteleyn and Temperley 2 discovered an algorithm that finds the Pfaffian orientation of a planar graph in polynomial time, which can be used to efficiently compute the number of perfect matchings. As already mentioned in the beginning, it is well known that the permanent of a \((0/1)\)-matrix \( A \) is equal to the number of perfect matchings in the graph that belongs to \( A \)'s biadjacency matrix. Based on this relationship, Valiant invented the hole theory of Holographic Algorithms 15,16 which is extended, e.g., by works of Cai and Lu 5,18. These holographic algorithms lead to polynomial time computable algorithms whereof former only exponential ones
where known, e.g., \#X-MATCHINGS, \#PL-3-NAE-SAT, \#PL-EVEN-LIN-2. In that course a surprising result is that \#PL-RTW-MON-3CNF\(^5\) has been proved to be \(\oplus\)P-complete, but \#\(2^k\)-PI-RTW-MON-3CNF to be polynomial time computable. Later it is shown by Cai et al.\(^5\) that this is actually a special case of \#\(2^k-1\)-PI-RTW-MON-3CNF, which are all polynomial time computable. The idea of holographic algorithms can be seen as a further development of Valiant’s proof idea to reduce counting satisfying assignments to the permanent. He replaced the graph gadgets by a more general type called matchgate but also showed in his paper \(^5\) that there do not exits elementary matchgrid algorithms for 3CNF formulas.

**Notations:** With each graph \(G\), we associated two matrices; first its adjacency matrix \(A_G\) and the bipartite adjacency matrix \(B_G \equiv \begin{pmatrix} 0 & A_G^* \end{pmatrix} \), which is always undirected and loop-free. \(A^t\) is the transposition of the matrix \(A\). Note that \(B_G^1_G = B_G\). The graph of \(B_G\) is denoted by \(\hat{G}\) and is the bipartite double cover graph of \(G\). Building the bipartite double cover of \(G\) can also be written as the tensor graph product \(\hat{G} = G \times K_2\) or as the Kronecker product of the two matrices \(B_G = A_G \otimes A_{K_2}\).

With \(0^{n \times m}\) we denote a zero matrix with dimensions \(n\) and \(m\). The notation \(A_{i_1,\ldots,i_k}^{o_1,\ldots,o_k}\) means the sub-matrix of \(A\) that is created by removing the columns \(i_1,\ldots,i_k\) and the rows \(o_1,\ldots,o_k\). For a boolean formula \(\Phi\), the number of satisfying assignments is denoted as \#\(\Phi\). The term \(\Phi_{x_i=1}\) means the formula \(\Phi\) with \(x_i\) substituted by 1. The graph that is constructed from a boolean formula according to the idea described by Valiant, is called a Valiant graph.

As usual, we use \(1_{\text{statement}}\) to denote 1 if the statement is true, and 0 otherwise. E.g., \(1_{\Phi}\) is sat is equal to 1 if the formula \(\Phi\) is satisfiable and 0 otherwise. We write, e.g. \(X = 3:Z\), if \(X\) is a multiple of 3.

**Definition 1.15** (Parsimonious reduction). \(f\) is reducible to \(g\) if there exists a polynomial time computable function \(\rho: \{0,1\}^* \rightarrow \{0,1\}^*\) such that for every \(x \in \{0,1\}^*, f(x) = g(\rho(x))\).

**Definition 1.16** (Many-one reduction). \(f\) is reducible to \(g\) if there exists a polynomial time computable function \(\rho: \{0,1\}^* \rightarrow \{0,1\}^*\) and \(\tau: N \rightarrow N\) such that for every \(x \in \{0,1\}^*, f(x) = \tau(g(\rho(x)))\).

**Proposition 1.17** (Relationship: Permanent and PerfectMatchings). Let \(G\) be an arbitrary unweighted graph and \(A_G\) its \((0/1)\)-adjacency matrix, then the permanent of \(A_G\) is equal to the number of perfect matchings in the graph \(\hat{G}\) (i.e., the bipartite double-cover).

Proposition 1.17 states a well known relationship between computing the permanent and counting the number of perfect matchings. The later is in general also a \(\text{NP}\)-hard problem, but can be solved efficiently if the graph is planar.

**Theorem 1.18** (Fisher-Kasteleyn-Temperley (FKT) algorithm). If \(A\) is a \((0/1)\)-adjacency matrix of a planar graph \(G\), then the number of perfect matchings in \(\hat{G}\) can be computed in polynomial time.

The FKT algorithm uses the planarity of the graph to obtain an embedding in the plane. From this embedding, the algorithm orients the edges of all faces in a certain way. The orientation of all edges is called a Pfaffian orientation. This orientations permits to count the number of perfect matchings efficiently.

**Proposition 1.19**. The permanent of a \((0/1)\)-matrix \(A\) with associated graph \(G\), can be computed in polynomial time if \(\hat{G}\) is planar.

Proposition 1.19 is just a consequence from Proposition 1.17 and Theorem 1.18. Finally, the planarity of a graph is defined as:

**Definition 1.20** (Planar graphs). A graph \(G = (V,E)\) is called planar if it can be embedding in the plane without two edges cross each other.

**Definition 1.21** (Circular planar graphs). A graph \(G = (V,E)\) is called circular planar according to a boundary \(B \subseteq V\), if it is planar and can be embedding on a disc \(D\) in the plane with all nodes from \(B\) lie on the circle that bounds \(D\) and all other nodes are in the interior of \(D\). The order of a circular planar graph is the order of the boundary nodes in clockwise direction.

**Remark**: The permanent of minors. As also used by Valiant and Ben-Dor/Halevi: if the permanent of a \((0/1)\)-matrix counts the number of cyclic covers, then the permanent of \(A_i^o\) is equal to the number of those ways from node \(i\) to node \(o\), where all nodes not part of the way can be covered by a cyclic cover.

\(^5\)This stands for Planar-ReadTwice-Monotone-3CNF, whereof read twice means that each variable occurs at most two times and monotone means that each variable only occurs positive.
2. Proofs that the permanent is \#P-complete

Since our work heavily depends on the idea of Valiant, we shortly describe the idea of the proof and the used gadgets. Besides the two named proofs of Valiant and Ben-Dor/Halevi, recently, Scott Aaronson [1] provided another proof based on a linear-optic setup and quantum reasoning that shows also that the permanent is \#P-complete.

Whenever a graph is drawn in the rest of the paper, whereof an edge has no shown weight, the weight is equal to 1. Whenever an adjacency matrix is given to a graph, no edge is labeled since the weights could be seen from the matrix.

2.1. Valiant’s proof. Valiant’s idea was, given a boolean 3CNF formula \( \Phi \) with \( n \) variables and \( m \) clauses, to construct a graph \( G \), such that the permanent of \( A_G \) involves \( \#\Phi \). In particular, in his proof the permanent is equal to \( 4^{3m}\#\Phi \). In order to construct the graph, he uses small subgraphs, called gadgets, that have special combinatorial properties. These properties cause that each valid assignment of \( \Phi \) is equivalent to a set of cyclic covers that sum up to a total weight of \( 4^{3m} \). And each assignment that does not satisfy \( \Phi \) is equal to a set of cyclic covers that sum up to a total weight of zero. Since the permanent is the sum over all weights of all cyclic covers, it is \( \text{perm}(A) = 4^{3m}\#\Phi \).

For the rest, we refer to the rewritten proof of Valiant that can be found in Barak and Aroras book [2].

There are in total three gadgets that Valiant uses to construct his graph \( G \).

2.1.1. The CLAUSE-gadget. For each of the \( m \) clauses in \( \Phi \), there is a small subgraph named CLAUSE-gadget.

2.1.2. The VARIABLE-gadget. The variable-gadget is constructed for each variable \( x_i \) in \( \Phi \). For each occurrence of \( x_i \) or \( \neg x_i \) in \( \Phi \), the VARIABLE-gadget has two vertices, whereof the true or the false assignments are on the two different cycles of the gadget. Whenever one of this cycles is used in a cyclic cover the other cyclic is prevented and the nodes must use their self-loop to form a valid cyclic cover. This guarantees, that one literal is either true or false but not both. The shown red lines connect the VARIABLE-gadget to the XOR-gadgets. We write \( G_1(x_i) \) to address the VARIABLE-gadget for variable \( x_i \).
2.1.3. The XOR-gadget. The XOR-gadget is the most important gadget. It is the only part of the whole final graph, that has edges with weights other than 0 or 1. The XOR-gadget connects on the one side to a CLAUSE-gadget and on the other side to a VARIABLE-gadget. As its name suggests, it allows only one side to be taken by a cyclic cover, but not both or not none. The reasoning is that either the VARIABLE-gadget gets the connection, which means that the literal is set to true, or the CLAUSE-gadget gets the connection, which means that that literal in the clause is set to false.

Setting $(i_1, o_1) = (1, 4)$ and $(i_2, o_2) = (4, 1)$ as the valid input/output ways, Valiant showed, that the adjacency matrix $A$ of an XOR-gadget has to fulfill the requirements:

\[
\begin{align*}
(A) \quad 0 &= \text{perm}(A) \\
(C) \quad c &= \text{perm}(A_{i_1, o_1}^{i_1, o_2}) \\
(D) \quad c &= \text{perm}(A_{i_2, o_2}^{i_1, o_1}) \\
(E) \quad 0 &= \text{perm}(A_{i_1, o_2}^{i_1, o_1}) \\
(F) \quad 0 &= \text{perm}(A_{i_2, o_1}^{i_2, o_2})
\end{align*}
\]

In the matrix he chose for his paper, the constant $c$ is equal to 4. If we rephrase the requirements (A)-(F) in words, we get: The cyclic cover can either enter the gadget using node 1 and leave at node 4 or the other way round. But taking both ways, none, or taking the wrong exit yields a weight of 0.

Remark: For those who are familiar with Valiant’s holographic algorithms, one perhaps notices, that these rules are related to the signature of a matchgate. The signature of a matchgate is enumeration of the weights of the matchgate perfect matchings, that arise if all possible combinations of input and output edges are removed. Here it is the permanent value of the gadget that arise if output and input nodes are removed.

All three shown gadgets appear as independent gadgets within the graph. All edges that connect between these gadget (the red edges in the figures) have weight 1, thus do not contribute to the weight of any cyclic-cover.

2.2. Ben-Dor & Halevis proof. In this work, only one gadget is used, which the authors also call CLAUSE-gadget. It represents a clause in the formula, which justifies the name, but actually has more similarity with the properties of Valiant’s XOR-gadget.
They stated the following rules in order to make their proof work.

(A) \(0 = \text{perm}(A)\) 
(B) \(0 = \text{perm}(A^{i,j}_{\omega_1})\) 
(C) \(0 = \text{perm}(A^{i,j}_{\omega_3})\)

(D) \(0 = \text{perm}(A^{i,j}_{\omega_2})\) 
(E) \(0 = \text{perm}(A^{i,j}_{\omega_2})\) 
(F) \(0 = \text{perm}(A^{i,j}_{\omega_1})\)

(G) \(0 = \text{perm}(A^{i,j}_{\omega_2})\) 
(H) \(0 = \text{perm}(A^{i,j}_{\omega_2})\) 
(G) \(0 = \text{perm}(A^{i,j}_{\omega_2})\)

(4) \(0 = \text{perm}(A^{i,j}_{\omega_3})\) 
(J) \(0 = \text{perm}(A^{i,j}_{\omega_3})\) 
(K) \(0 = \text{perm}(A^{i,j}_{\omega_3})\)

(5) \(0 = \text{perm}(A^{i,j}_{\omega_3})\) 
(M) \(c = \text{perm}(A^{i,j}_{\omega_3})\) 
(N) \(c = \text{perm}(A^{i,j}_{\omega_3})\)

(O) \(c = \text{perm}(A^{i,j}_{\omega_3})\) 
(P) \(c = \text{perm}(A^{i,j}_{\omega_3})\) 
(Q) \(c = \text{perm}(A^{i,j}_{\omega_3})\)

In their paper they chose \((i_1, i_2, i_3) = (1, 2, 3)\) and \((\omega_1, \omega_2, \omega_3) = (3, 4, 5)\). The rules can be subsumed shortly by:

Each CLAUSE-gadget must be used and no input-output switching is allowed (i.e., if one enters at \(i_j\) one must leave at \(\omega_j\)), since this yields a permanent of zero.

2.3. A preventing Identity. In the computational complexity lecture notes of Oded Goldreich one can find the following lines regarding Valiant’s proof

“If there is a transformation from \(\Phi\) to \(G\), such that \(\text{perm}(A) = c^m \#\Phi\), then assuming \(\text{NP} \nsubseteq \text{BPP}\), the constant \(c\) must be even.”

This statement refers to the constant \(c\) in the rules for the XOR-gadget (see Rules \(\Box\)) and which is \(c = 4\) in Valiant’s proof. If the constant would be \(c = 1\) one has a parsimonious reduction to the permanent, but this would contradict the assumption \(\text{NP} \nsubseteq \text{BPP}\). In the next lines, we show that this would not only contradict that well believed complexity assumption, but also a known determinant identity. The tool we use is that we can evaluate the permanent modulo 2 since the determinant and the permanent are equal in \(\mathbb{Z}/2\mathbb{Z}\).

One could observe that there is no need to restrict the requirements of the XOR-gadget to such narrow rules like the ones described in Rules \(\Box\). If one is only interested in deciding whether the formula has a satisfying assignment, one bit of information is enough and thus computing modulo 2 is sufficient. This weakening can be translated to the following new rules for the XOR-gadget:

(A) \(0 \equiv \text{perm}(A) \equiv \det(A) \pmod{2}\) 
(B) \(0 \equiv \text{perm}(A^{i,j}_{\omega_1}) \equiv \det(A^{i,j}_{\omega_1}) \pmod{2}\)

(C) \(1 \equiv \text{perm}(A^{i,j}_{\omega_1}) \equiv \det(A^{i,j}_{\omega_1}) \pmod{2}\) 
(D) \(1 \equiv \text{perm}(A^{i,j}_{\omega_2}) \equiv \det(A^{i,j}_{\omega_2}) \pmod{2}\)

(E) \(0 \equiv \text{perm}(A^{i,j}_{\omega_1}) \equiv \det(A^{i,j}_{\omega_1}) \pmod{2}\) 
(F) \(0 \equiv \text{perm}(A^{i,j}_{\omega_2}) \equiv \det(A^{i,j}_{\omega_2}) \pmod{2}\)

Suppose, we are given a boolean formula from \(\text{Unique–3SAT}\). All cyclic covers that do not yield a satisfying assignment sum up to an even weight and thus vanish modulo 2. If there is a satisfying assignment, then this satisfying assignment yields an odd sum, which could be detected by evaluating the determinant modulo 2.

However, regarding the determinant there exists an identity that prevents someone from finding such a gadget.

**Theorem 2.1** (Desnanot-Jacobi identity (DJi)). Let \(M = m_{i,j}, 1 \leq i, j \leq n\) be a square matrix

\[\det(M) \det(M^{i,j}_{\omega_1}) = \det(M^{i,j}_{\omega_1}) \det(M^{i,j}_{\omega_2}) - \det(M^{i,j}_{\omega_2}) \det(M^{i,j}_{\omega_3})\]

(6) 

One finds the Desnanot-Jacobi identity mostly in the literature in connection with Dodgson condensation algorithm \(\Box\). If one applies the DJi to the weakened Rules \(\Box\) for the XOR-gadget, one gets

\(\det(A) \det(A^{i,j}_{\omega_1}) = \det(A^{i,j}_{\omega_1}) \det(A^{i,j}_{\omega_2}) - \det(A^{i,j}_{\omega_2}) \det(A^{i,j}_{\omega_3}) \equiv 0 \cdot 0 \equiv 0 \cdot 1 \cdot 0 \cdot 0 \quad \pmod{2}\)

which is false. Similar, if one applies this to the rules of of Ben-Dor and Halevis CLAUSE-gadget, one gets

\(\det(A) \det(A^{i,j}_{\omega_1}) = \det(A^{i,j}_{\omega_1}) \det(A^{i,j}_{\omega_2}) - \det(A^{i,j}_{\omega_2}) \det(A^{i,j}_{\omega_3}) \equiv 0 \cdot 1 \equiv 1 \cdot 1 \cdot 1 \cdot 0 \cdot 0 \quad \pmod{2}\)

which is also false. Additionally, if one lays down the rules for the EQUALITY-gadget of Blaeser et al. \(\Box\) one gets

\(\det(A) \det(A^{i,j}_{\omega_1}) = \det(A^{i,j}_{\omega_1}) \det(A^{i,j}_{\omega_2}) - \det(A^{i,j}_{\omega_2}) \det(A^{i,j}_{\omega_3}) \equiv 1 \cdot 1 \equiv 0 \cdot 0 \cdot 0 \cdot 0 \quad \pmod{2}\)

which again is false\(\Box\). So, the ability to decide if a formula from \(\text{Unique–3SAT}\) has a satisfying assignment by trying to find a suitable gadget can not succeed.

**Proposition 2.2.** The constant \(c\) in Valiant’s proof can not be odd due to the Desnanot-Jacobi identity.
A DJi-compatible gadget. One could ask the question: What kind of useful gadget is possible, that is compatible with the DJi? The answer is: A PLANARITY-gadget. A PLANARITY-gadget substitutes each edge crossing with a planar subgraph and can be described with the rules

\[
\begin{align*}
(A) \quad 1 &\equiv \text{perm}(A) \equiv \text{det}(A) \pmod{2} \\
(B) \quad 1 &\equiv \text{perm}(A_{i_1,i_2}^{o_1,o_2}) \equiv \text{det}(A_{i_1,i_2}^{o_1,o_2}) \pmod{2} \\
(C) \quad 1 &\equiv \text{perm}(A_{i_1}^{o_1}) \equiv \text{det}(A_{i_1}^{o_1}) \pmod{2} \\
(D) \quad 0 &\equiv \text{perm}(A_{i_2}^{o_2}) \equiv \text{det}(A_{i_2}^{o_2}) \pmod{2} \\
(E) \quad 0 &\equiv \text{perm}(A_{i_1}^{o_1}) \equiv \text{det}(A_{i_1}^{o_1}) \pmod{2} \\
(F) \quad 0 &\equiv \text{perm}(A_{o_1}^{i_1}) \equiv \text{det}(A_{o_1}^{i_1}) \pmod{2}
\end{align*}
\]

which translates into

\[
\text{det}(A)\text{det}(A_{i_1,i_2}^{o_1,o_2}) = \text{det}(A_{i_1}^{o_1})\text{det}(A_{i_2}^{o_2}) - \text{det}(A_{i_1}^{i_1})\text{det}(A_{i_2}^{i_2}) \equiv 1 \cdot 1 \equiv 0 \cdot 0 - 1 \cdot 1 \pmod{2}
\]

and is true.

In the following we introduce the following notation regarding the rules of a gadget.

Definition 2.3 (Signature). For each gadget \(G\) with the input-output pairs \((i_1,o_1)\) and \((i_2,o_2)\) we define the signature \(S_G\) as

\[
S_G \overset{\text{def}}{=} (r_1,r_2,r_3,r_4,r_5,r_6)_p
\]

where the \(r_i\)'s are from

\[
\begin{align*}
(A) \quad \text{perm}(A) &\equiv r_1 \pmod{p} \\
(B) \quad \text{perm}(A_{i_1,i_2}^{o_1,o_2}) &\equiv r_2 \pmod{p} \\
(C) \quad \text{perm}(A_{i_1}^{o_1}) &\equiv r_3 \pmod{p} \\
(D) \quad \text{perm}(A_{i_2}^{o_2}) &\equiv r_4 \pmod{p} \\
(E) \quad \text{perm}(A_{i_1}^{o_1}) &\equiv r_5 \pmod{p} \\
(F) \quad \text{perm}(A_{o_1}^{i_1}) &\equiv r_6 \pmod{p}
\end{align*}
\]

When the signature array has no subscript \(p\) then we refer to the integer values rather than the modulo \(p\) reduced value of the permanent. ▶

3. A planar Valiant graph.

Our progress depends on the observation that the DJi is a determinant identity and a similar identity for the permanent is not known. The CLAUSE-gadget as well as the VARIABLE-gadget are uncritical, since they already are unweighted graphs. But the XOR-gadget has edges with weights different from 0 and 1, which does not allow a planar Valiant graph.

\(G\) has 1,2 as input nodes and 3,4 as its output nodes. The correct path is from 1 to 3 and 2 to 4. Entering in 1 and leaving in 4 (or 2 and 3) yields, according to its signature, a permanent of 0 in \(\mathbb{Z}/3\mathbb{Z}\). Its signature is

\[
S_{G_3} = (6,3,4,4,6,3) = (0,0,1,1,0,0)_3
\]

We now proof the first theorem, which shows that already this new gadget allows to efficiently decide the satisfiability of a formula from Unique-3SAT if \(\tilde{G}\) turns out to be planar.

\[\text{That means, the path that could lead to a satisfying assignment}\]
Theorem 3.1. Let $\Phi$ be a boolean formula from Unique−3SAT with $n$ variables and $m$ clauses and let $G$ be the graph that is constructed from $\Phi$ according to Valiant’s proof, using the gadgets $G_1$ (see Figure 1), $G_2$ (see Figure 2), $G_3$ (see Figure 3). If $G$ is planar, then there exists an algorithm $A$ that decides $\Phi$’s satisfiability in polynomial time $O(m^3)$.

Proof. For the proof, we follow the arguments of Valiant in a close way, but use the notation of an $F$-completion from Ben-Dor and Halevi (see [3], Definition 8). We do not rewrite the whole proof, but only argue along the essential step, that is, where the sum over all cyclic-covers is computed. We describe an algorithm that does:

$A(\Phi)$:

$G \leftarrow \text{buildValiantGraph}(G_1,G_2,G_3,\Phi)$
$s \leftarrow \text{PerfMatch}(G) \pmod{3}$
return $1_{s=0}$

The input of $A$ is a formula $\Phi$ from Unique−3SAT. The only gadget we have substituted so far is the XOR-gadget. In all cases where a cyclic cover in Valiant’s proof contributes a weight of 0, the same cyclic cover in our case contributes a weight that is 0 or a multiple of 3. And in all cases where in Valiant’s proof the cyclic cover contributes a weight of $4^{3m}$, the same cyclic cover in our case contributes a weight that is congruent to 1 modulo 3. Since any cyclic cover must be of one of these types, the final permanent value is the sum over

$$\text{perm}(A) = \sum_{C \in F} W(C) = 3 \cdot Z + 1_{\Phi \text{ is sat}} \cdot (1 + 3 \cdot Z) = 3 \cdot Z + 1_{\Phi \text{ is sat}} \cdot (1 + 3 \cdot Z)$$

After reducing modulo 3 we get

$$\text{PerfMatch}(G) \pmod{3} = 1_{\Phi \text{ is sat}} = \begin{cases} 0 & \Phi \text{ is unsatisfiable} \\ 1 & \Phi \text{ is satisfiable} \end{cases}$$

If $\Phi$ consists of $n$ variables $x_1,\ldots,x_n$ and $m$ clauses. $x_i$ together with $\neg x_i$ occurs $n_i$ times, thus $\sum_{i=1}^{n} n_i \leq 3m$ as well as $n \leq 3m$, since $\Phi$ is a 3CNF. Hence, the graph $G$ consists of

$$m|G_1| + \sum_{i=1}^{n} |G_2(x_i)| + 3m|G_3| = m \cdot 4 + \sum_{i=1}^{n} (2 + 2n_i) + 3m \cdot 6$$
$$= 2n + 2 \sum_{i=1}^{n} n_i + 22m$$
$$< 6m + 6m + 22m$$
$$< 34m$$

nodes. The bipartite double cover operations doubles the number of nodes, so we can bound the number of nodes from above via $|G| \leq 68m$. Finally, Kasteleyn’s algorithm runs in time that is cubic in the number of nodes, which yields a total running time of our algorithm of $O(68^3 m^3)$, or $O(m^3)$ for large $m$.

The following proposition shows the well known consequence between deciding satisfiability and finding a solution.

Corollary 3.2. If one could decide the satisfiability of a formula $\Phi$ from Unique−3SAT that has $n$ distinct variables in time $T$, one can find the solution in time $\Omega(nT)$.

Proof. To find the solution, if it exists, apply to following algorithm, which makes calls to the oracle $O^{\text{Unique−3SAT}}(\Phi)$. That oracle implements the algorithm from Theorem 3.1 and returns 0 or 1 according to $\Phi$’s satisfiability:

1. $v \leftarrow O^{\text{Unique−3SAT}}(\Phi)$.
2. assignment $\leftarrow [0,\ldots,0]$
3. if $v = 0$ then return 'unsatisfiable'.
4. for $i = 1$ to $n$ do
5. $v \leftarrow O^{\text{Unique−3SAT}}(\Phi_{x_i=1})$
6. if $v = 0$ then assignment[$i$] = 0
7. else assignment[$i$] = 1
8. return assignment

At the end, in the array assignment at position $i$, a value of the $i$-th variable is stored that leads to a satisfying assignment. The loop makes $n$ calls to the oracle, and each loop is evaluated in time $T$. □
Theorem 3.3. Let $\Phi$ be random boolean formula from 3SAT with $n$ variables and $m$ clauses and let $G$ be the graph that is constructed from $\Phi$ according to Valiant’s proof, using the gadgets $\Theta_1$ (see Figure 1), $\Theta_2$ (see Figure 2), $\Theta_3$ (see Figure 5). If $\hat{G}$ is planar, then there exists an algorithm $A$ that decides $\Phi$’s satisfiability in randomized polynomial time $O(km^3)$ with an a success probability of

$$\Pr[s \leftarrow A(\Phi, k) | 1_{\Phi \text{ is sat}} = s] = 1 - s \frac{1}{3^k}$$

Proof. The proof is similar to the one of Theorem 3.1 so we can overtake the same algorithm $A$ and its running time. For the proof, one could also use function $f$ from Valiant-Vazarani and convert the formula into a formula from Unique-3SAT. But that will increase unnecessary the number of nodes in the resulting Valiant graph.

The difference here is, that for a formula from 3SAT it could happen, that the number of satisfying solutions is a multiple of 3, which can not be distinguished from the case that the formula is unsatisfiable. Assume the following algorithm $B$:

1. for $i = 1$ to $n$ do
2. $s \leftarrow A(\Phi_{x_i=1})$
3. if $s = 1$ then return ‘satisfiable’
4. return ‘unsatisfiable’

Case 1: $\Phi$ is unsatisfiable. The algorithm will never return ‘satisfiable’ in Step 3, since no substitution of $x_i = 1$ can make the formula satisfiable. So the algorithm always returns the correct result.

Case 2: $\Phi$ is satisfiable. If the algorithm returns ‘satisfiable’, the formula must be satisfiable and we are done. Whenever the algorithm returns ‘unsatisfiable’, each solution set of the formula with $x_i = 1$ must again be a multiple of 3. Since we have a random boolean formula, there is no dependency among the bits of the solutions, the chance that this happens for each $\Phi_{x_i=1}$ is $3^{-1}$. Thus, the chance after $k$-trials not to detect the satisfiability is $3^{-k}$. □

4. Enhancements to reach planarity of $\hat{G}$.

To enhance the previous approach, we try to make the Valiant graph already planar. The reason is that for large graphs with a low connectivity it turns out that already planar graphs do more often have planar bipartite double covers than non-planar graphs. In order to achieve this, we restrict our input to $pn$-planar formulas.

Figure 6. Two different drawings of the incidence graph of the formula $\Phi = (x_1 \lor x_2 \lor x_3) \land (x_1 \lor \neg x_4 \lor x_6) \land (\neg x_1 \lor \neg x_3 \lor x_5) \land (\neg x_4 \lor \neg x_5 \lor x_6)$. On the left in a typical bipartite form and on the right the $pn$-planar embedding (without the edges that connect the variable-nodes).

In Figure 4 an example of a formula with an $pn$-planar embedding is shown. The reason to focus on this kind of formulas is, that they can exactly be retraced with the graph gadgets from Valiant. E.g., the VARIABLE-gadget has two directions. One for the positive literals and one for the negative literals. By the usage of $pn$-planar formulas, it is possible to turn each gadget into the correct face direction to keep planarity.

To make the entire graph planar, all gadgets have to be planar itself. However, the current new XOR-gadget $\Theta_3$ is not planar. Even more, simple planarity is not sufficient. The gadgets must be circular planar with boundaries that consider their input and output nodes as well as the face on which a clause is located. We will describe these said requirements in the next sections in more detail.

8Unfortunately, we can base this statement only on a few calculation we did.
4.0.1. **The RL-XOR-gadget.** The first gadget we found can be seen in Figure 7.

![Figure 7. GRL: The new RIGHT-LEFT-XOR-gadget (RL-XOR-gadget). The red lines are not directly part of the gadget. As it can be seen, the gadget has circular planar embedding according to the nodes 1, 2, 3, 4.](image)

It has 1, 2 as input nodes and 3, 4 as its output nodes and all these nodes are boundary nodes. The correct path is from 1 to 3 and 2 to 4. Its signature is

\( S_{\text{RGL}} = (3, 3, 2, 4, 3, 3) = (0, 0, 2, 1, 0, 0)_3 \)

4.0.2. **The LR-XOR-gadget.** The second gadget we found can be seen in Figure 8.

![Figure 8. GLR: The new LEFT-RIGHT-XOR-gadget (LR-XOR-gadget). The red lines are not directly part of the gadget. As it can be seen, the gadget has circular planar embedding according to the nodes 1, 2, 3, 4.](image)

Equivalent to the RL-XOR-gadget, it has 1, 2 as input nodes and 3, 4 as its output nodes and all these nodes are boundary nodes. The correct path is from 1 to 3 and 2 to 4. Its signature is

\( S_{\text{GLR}} = (12, 3, 5, 7, 3, 3) = (0, 0, 2, 1, 0, 0)_3 \)

5. **Circular planar bipartite double covers and a limiting conjecture**

We now turn towards generalization. The bipartite double cover operation turns out to play a central role and we are mainly interested in the way it changes the planarity of a graph. For our approach, a mandatory requirement is, that the used gadgets itself must turn into planar bipartite double covers in order to have chance to achieve planarity for the whole graph. And simple planarity alone is not enough. Similar to the RL and LR-XOR-gadgets, that achieve circular planarity for their input/output nodes, one now has to take care about the circular planarity of their bipartite double cover regarding the input/output nodes.
Clause 1: \( x_1 \lor x_2 \lor x_3 \)

Clause 2: \( x_1 \lor \neg x_4 \lor x_6 \)

Clause 3: \( \neg x_1 \lor \neg x_3 \lor x_5 \)

Clause 4: \( \neg x_4 \lor \neg x_5 \lor x_6 \)

**Figure 9.** This is a pn-planar formula \( \Phi = (x_1 \lor x_2 \lor x_3) \land (x_1 \lor \neg x_4 \lor x_6) \land (\neg x_1 \lor \neg x_3 \lor x_5) \land (\neg x_4 \lor \neg x_5 \lor x_6) \) with its associated Valiant graph. The clauses 1, 2 and 4 are on face 1 and the clause 3 is on face 2. The dashed connection are the two connection from Figure 4 that actually circumvent the clause nodes 1 and 4.

**Figure 10.** On the left the original example graph, with input and output pairs (3, 1) and (4, 6). On the right the bipartite double cover with corresponding input and output pair (9, 1) and (10, 6). It can be connected in the same way as the original graph.

**Lemma 5.1** (Bipartite connectable graphs). If a planar graph \( G \) can be drawn as a connection of gadgets \( G_i \), then \( \hat{G} \) will be also planar if each gadget itself has a circular planar bipartite double cover \( \hat{G}_i \), with the input and output nodes on the boundary in the same order as in \( G \) itself.

**Proof.** The lemma is immediately clear if one considers the example gadget in Figure 10. The graph on the left side is the original graph \( G \) and on the right side its bipartite double cover. If one builds a graph \( G \) using gadgets \( G_i \) (like shown on the left side in the figure), one simply replaces each of occurrences of \( G_i \) with \( \hat{G}_i \) (the one on the right side in the figure). Since the input and output pairs of \( \hat{G}_i \) are in the same order and also on the boundary, the original planar structure between the gadgets is overtaken and hence \( \hat{G} \) will be planar if \( G \) is. \( \square \)

Lemma 5.1 states that one could draw the bipartite double cover of a graph by simply replacing each gadget by its BDC, if each gadget turns into suitable bipartite double cover as well. Knowing this and remembering that we already use planar formulas, in particular pn-planar formulas, as the input, actually we only need to find the correct gadgets to make the BDC planar. We do not have to care about the structure of the input formulas, but only that the gadgets itself have a appropriate bipartite double cover. And since gadgets could be rather small compared to the overall graph, this seems to be a more accessible approach. However, we will show in the remaining sections, that it is not that easy at all.

First, we take a look at the bipartite double cover of the CLAUSE-gadget, cf. Figure 11.

As it can be seen, the CLAUSE-gadget has a circular planar embedding for its bipartite double cover, but with a wrong order. The order must contain the nodes of the pairs (5, 3), (6, 1) and (7, 2) as successive nodes. Every interleaving destroys the planarity of the graph. So Valiant’s original CLAUSE-gadget does not have the correct order regarding its boundary nodes when building its bipartite double cover.
Figure 11. On the left the CLAUSE-gadget and on the right the bipartite double cover with the boundary nodes on the dashed disc. The connections (7,2) and (5,3) intersect, which destroys the overall planarity of the graph and makes the original CLAUSE-gadget not usable.

Figure 12. The input/output pairs are \((n+3, 2), (n+5, 4), \ldots, (n+m+1, m)\) and \((n+m+3, m+2), (n+m+5, m+4), \ldots, (n+n-1, n-2)\). And they can be drawn at pairs on a circle.

**Remark:** If we talk about boundary nodes and their order, we always only consider the input and output nodes on the boundary like drawn in Figure 11. One could easily also increase the boundary by the nodes 4 and 8, but as they are not used as input and output nodes, we keep them as interior nodes.

**Definition 5.2.** Let \(\mathcal{G}\) be a gadget with input/output nodes \((i_1, o_1)\) and \((i_2, o_2)\). If the bipartite double cover is circular planar according to the nodes \(i_1', i_2', o_1', o_2\), then it is called

- **connectable** if the order is \(i_1', o_1, i_2', o_2\),
- **extendable** if the order is \(i_1', i_2, o_2, o_1\).

Note that an order of \(i_1', i_2', o_1, o_2\) can not exists if the gadget is planar. Regarding the two new XOR-gadgets it holds that, \(\mathcal{G}_{RL}\) is connectable and \(\mathcal{G}_{LR}\) is extendable. They both have a planar bipartite cover, but are not circular planar according to the input/output nodes. The only gadget that turns out to be usable, i.e., that also is circular planar with the correct boundary order, is the VARIABLE-gadget. It has the following bipartite double cover (see Figure 12).

The input/output pairs of this bipartite double cover are clockwise for the TRUE-nodes and anticlockwise for the FALSE-nodes. It can be swapped by swapping the circle horizontally.

The next two Lemmas are well known results regarding bipartite graphs.

**Lemma 5.3.** If \(\mathcal{G}\) is a bipartite graph, then \(\hat{\mathcal{G}}\) is a disconnected graph that consists of two copies of \(\mathcal{G}\).

**Lemma 5.4.** A graph \(\mathcal{G}\) is bipartite if and only if \(\hat{\mathcal{G}}\) is disconnected.

There is a relationship between the signature of \(\mathcal{G}\) and the signature of \(\hat{\mathcal{G}}\).
Lemma 5.5. Let $G$ be a graph with adjacency matrix $A$ that has the signature

\begin{equation}
S_G = (r_1, r_2, r_3, r_4, r_5, r_6)
\end{equation}

Let $B$ be the matrix of the bipartite double cover $\hat{G}$, with $i'_1 = i_1 + \dim(A)$ and $i'_2 = i_2 + \dim(A)$. Then

\begin{equation}
S_{\hat{G}} = (r_1^2, r_1 r_2, r_1 r_3, r_1 r_4, r_1 r_5, r_1 r_6)
\end{equation}

Proof. The proof can be deduced from Ryser permanent formula for a $n \times n$ matrix $B$

\begin{equation}
\perm(B) = (-1)^n \sum_{S \subseteq \{1, \ldots, n\}} (-1)^{|S|} \prod_{i=1}^{n} \sum_{j \in S} a_{ij}
\end{equation}

The dimension of our matrix is even since it is a bipartite matrix, i.e, we write $|B| = 2n$ so $B$ is of the form

\begin{equation}
B = \begin{pmatrix}
0^{n \times n} & A \\
A^t & 0^{n \times n}
\end{pmatrix}
\end{equation}

with $A$ being a $n \times n$ matrix. Further note that $\perm(A) = \perm(A^t)$. With the rows/columns at

\begin{equation}
B = \begin{pmatrix}
0^{n \times n} & A \\
A^t & 0^{n \times n}
\end{pmatrix}
\end{equation}

\begin{equation}
\perm(B) = (-1)^{2n-\tau} \sum_{S \subseteq \{1, \ldots, 2n-\tau\}} (-1)^{|S|} \prod_{i=1}^{2n-\tau} \sum_{j \in S} a_{ij}
\end{equation}

\begin{equation}
\quad = (-1)^{2n-\tau} \sum_{S \subseteq \{1, \ldots, 2n-\tau\}} (-1)^{|S|} \left( \prod_{i=1}^{n-\tau/2} \sum_{j \in S} a_{ij} \cdot \prod_{i=n+1-\tau/2}^{2n-\tau} \sum_{j \in S} a_{ij} \right)
\end{equation}

with $\tau \in \{0, 1\}$ according to the cases if zero, one pair or both pairs of $(i'_1, o_1), (i'_2, o_2)$ have been removed. One can write each subset $S \subseteq \{1, \ldots, 2n\}$ as two unique subsets $S_1, S_2$, with $S_1 \cup S_2 = S$ and $S_1 \subseteq \{1, \ldots, n-\tau\}$ and $S_2 \subseteq \{n+1-\tau, \ldots, 2n-\tau\}$, so $S_1 \cap S_2 = \emptyset$. Since $a_{ij} = 0$ for $1 \leq i, j \leq n$ and $n+1 \leq i, j \leq 2n$ we can write

\begin{equation}
\perm(B) = (-1)^{2n-\tau} \sum_{S_1 \subseteq \{1, \ldots, n-\tau\}, S_2 \subseteq \{n+1-\tau, \ldots, 2n-\tau\}} (-1)^{|S_1|} \left( \prod_{i=1}^{n-\tau} \sum_{j \in S_2} a_{ij} \cdot \prod_{i=n+1-\tau}^{2n-\tau} \sum_{j \in S_1} a_{ij} \right)
\end{equation}

\begin{equation}
\quad = (-1)^{2n-\tau} \sum_{S_1 \subseteq \{1, \ldots, n-\tau\}, S_2 \subseteq \{n+1-\tau, \ldots, 2n-\tau\}} (-1)^{|S_2|} \left( \prod_{i=1}^{n-\tau} \sum_{j \in S_2} a_{ij} \cdot \prod_{i=n+1-\tau}^{2n-\tau} \sum_{j \in S_1} a_{ij} \right)
\end{equation}

\begin{equation}
\quad = (-1)^{2n-\tau} \sum_{S_2 \subseteq \{n+1-\tau, \ldots, 2n-\tau\}} (-1)^{|S_2|} \left( \prod_{i=1}^{n-\tau} \sum_{j \in S_2} a_{ij} \cdot \prod_{i=n+1-\tau}^{2n-\tau} \sum_{j \in S_1} a_{ij} \right)
\end{equation}

\begin{equation}
\quad = \perm(A) \cdot \perm(A^t_\delta)
\end{equation}

\begin{cor}
The following two observations can be made:

1. An XOR-gadget itself could not be bipartite
2. An EQUALITY-gadget itself could be bipartite
\end{cor}
Proof. Let be a bipartite graph \( G = \left( \begin{array}{cc} 0 & A_G \\ A^t_G & 0 \end{array} \right) \). Let \( S_A = (r_1, r_2, r_3, r_4, r_5, r_6) \) be the signature of \( A \). To make \( G \) an XOR-gadget, its signature must be of the form (see Lemma 5.5)

\[
S_\phi = (0, 0, 0, 0, 0, 0) = (r_1^2, r_1 r_2, r_1 r_3, r_1 r_4, r_1 r_5, r_1 r_6) = r_1 S_A
\]

It follows that \( r_1 = 0 \) must be equal to 0 modulo \( p \) and hence also \( r_1 r_3 \) and \( r_1 r_4 \) must vanish modulo \( p \), which contradicts the requirements of an XOR-gadget signature.

The EQUALITY-gadget has the signature

\[
S_\phi = (\neq 0, \neq 0, 0, 0, 0, 0) = (r_1^2, r_1 r_2, r_1 r_3, r_1 r_4, r_1 r_5, r_1 r_6) = r_1 S_A
\]

which leads to the fact that \( r_1 \) and \( r_2 \) must be non equal to zero modulo \( p \) and hence \( r_3, r_4, r_5, r_6 \) must be also non zero, which does not contradict the requirements on an EQUALITY-gadget signature.

As the next step, we present our main conjecture, that states that certain gadgets can not be found.

**Conjecture 5.7.** Let \( G \) be a graph with adjacency matrix \( A \) and the signature

\[
S_G = (r_1, r_2, r_3, r_4, r_5, r_6)_p
\]

for some prime \( p > 2 \). The bipartite double cover graph \( \hat{G} \) can only be circular planar according to the nodes \( (\dim + i_1, \dim + i_2) = (i_1', i_2') \) if the mapping

\[
\mu(r_i) = \begin{cases} 
0, & \text{if } r_i = 0 \\
1, & \text{if } r_i > 0 
\end{cases}
\]

does lead to a valid congruence of the form

\[
\mu(r_1) \cdot \mu(r_2) \equiv \mu(r_3) \cdot \mu(r_4) - \mu(r_5) \cdot \mu(r_6) \pmod{2}
\]

An exception is the case where all \( r_i > 0 \). In that case, a circular bipartite double cover exists, but that is only connectable according to the nodes \( i_1', i_2', o_1, o_2 \).

One could “prove” this conjecture with the sentence “Otherwise \( \text{RP} = \text{NP} \)”. To do so, one would pick a XOR-gadget that contradicts the conjecture and take a formula \( \Phi \) from Unique-3SAT and convert it into a pn-planar formula. Then one would construct the Valiant graph and build, according to Lemma 5.1, the bipartite double cover by simply substituting each gadgets by its bipartite double cover. Hence, the total bipartite double cover has the same planar layout as the Valiant graph and, abstracted, as the formulas incidence graph. After applying Kasteleyn’s algorithm, one could determine \( \Phi \) satisfiability as it is done in the proof of Theorem 3.1.

Note, the conjecture does not exclude the possibility that the bipartite double cover of a gadget will be planar, but only that it can not be circular planar according to all input/output nodes. We found gadgets that have three of the four nodes on the boundary but not all four. In Figure 14, one can see a XOR-gadget that has one pair on the boundary and one pair in the interior, but an XOR-gadget with all four nodes on the boundary can not be found.

**Proposition 5.8.** Since the signature \( S = (0, 0, 0, 0, 0, 0)_p \) does not fulfill the Equation 22, there can not exists an XOR-gadget \( G \), such that \( G \) is circular planar with \( (i_1', i_2') \) and \( (o_1, o_2) \) as boundary nodes.

**Proposition 5.9.** Corollary 5.6 together with Lemma 5.3 add up to the fact, that one can not find an EQUALITY-gadget \( G \) that is itself bipartite (or any other gadget that could exists if \( r_1 \) is equal to one) and is circular planar, since its bipartite double cover would consists of two copies of \( G \) and this would contradict Conjecture 5.7.

Let us turn to the one exception named in the Conjecture 5.7, that is the case when all \( r_i > 0 \). This exception is interesting, since it surprisingly allows the nodes \( (i_1', i_2') \) and \( (o_1, o_2) \) to be on the boundary but puts restrictions to their order. On the first sight, one would argue, that a gadget with such a signature where no input variant vanishes modulo \( p \) is not very useful, since it does not allow to exclude cyclic covers that would potential lead to non satisfying assignments.

Before we proceed, we show the rules to concatenate two gadgets. We then show that if the exceptional gadgets from the conjecture would also be extendable, one could again build gadgets that contradict the conjecture itself.

**Lemma 5.10** (Concatenating gadgets.). Given two gadgets \( G_1 \) and \( G_2 \) with the signatures \( S_{G_1} = (r_1, r_2, r_3, r_4, r_5, r_6) \) and \( S_{G_2} = (r_1, r_2, r_3, r_4, r_5, r_6) \) and the input and output nodes \( (i_1, o_1), (i_2, o_2) \) and \( (i_1, o_1), (i_2, o_2) \). If one concatenates them, by inserting the new edges \( c_1 = (o_1, i_1) \) and \( c_2 = (o_2, i_2) \), one gets a new gadget \( G \) with the signature

\[
S_G = (r_1 r_1, r_2 r_2, r_3 r_3, r_4 r_4, r_5 r_5, r_6 r_6)
\]
and input/output pairs \((i_1, \omega_1), (i_2, \omega_2)\).

**Proof.** The concatenation of two gadgets is shown in Figure 13. A look at the figure immediately illustrates why

![Figure 13](image)

**Figure 13.** This figure illustrates the concatenation principle of two extendable gadgets \(\mathcal{G}_1\) and \(\mathcal{G}_2\). The red edges are their external edges and their internal edges are not visible. The new gadget \(\mathcal{G}\) is constructed by inserting the edges \(c_1\) and \(c_2\), which are then internal edges of the new gadget \(\mathcal{G}\). The new input/output pairs are \((i_1, \omega_1), (i_2, \omega_2)\).

only extendable gadgets can be used to create a new planar gadget. Assume that, e.g., at least one of those gadgets would be connectable. That means, in the case that it would be \(\mathcal{G}_1\), that the order of the boundary nodes would be \(i_1, o_1, i_2, o_2\). The insertion of the new edges \((i_1, \omega_1), (i_2, \omega_2)\) would surround the input edge \(i_2\), which would lead to an edge crossing and hence non-planarity.

Concatenation of two gadgets is in principle the same as to create a small Valiant graph. Like Bon-Dor and Halevi argued in their proof [3], we can argue with the independence of the two gadgets. Let the signature of \(\mathcal{G}\) be \(S_\mathcal{G} = (R_1, R_2, R_3, R_4, R_5, R_6)\), then

1. No external edge is taken and the permanent of \(\mathcal{G}\) is equal to the product of the permanent of \(\mathcal{G}_1\) and \(\mathcal{G}_2\). Hence \(R_1 = r_1\rho_1\).

2. All external edges are taken and thus also the two connecting edges \(c_1\) and \(c_2\) must be taken. It holds

\[
\text{perm}(\mathcal{G}_1) = \text{perm}(\mathcal{G}_2) = \text{perm}(\mathcal{A}_{\mathcal{G}_1}) \text{perm}(\mathcal{A}_{\mathcal{G}_2}) = r_2 \rho_2 = R_2.
\]

3. External edges \(i_1\) and \(\omega_1\) are taken. There are two possibilities to choose the connection edges: either \(c_1\), therefore \(o_1\) and \(i_1\) are also taken, or \(c_2\), thus also \(o_2\) and \(i_2\) are taken. Hence

\[
\text{perm}(\mathcal{G}_1) = \text{perm}(\mathcal{G}_2) = \text{perm}(\mathcal{A}_{\mathcal{G}_1}) \text{perm}(\mathcal{A}_{\mathcal{G}_2}) = r_3 \rho_3 + r_5 \rho_5.
\]

4. External edges \(i_2\) and \(\omega_2\) are taken. There are two possibilities to choose the connection edges: either \(c_2\), therefore \(o_2\) and \(i_2\) are also taken, or \(c_1\), thus also \(o_1\) and \(i_1\) are taken. Hence

\[
\text{perm}(\mathcal{G}_1) = \text{perm}(\mathcal{G}_2) = \text{perm}(\mathcal{A}_{\mathcal{G}_1}) \text{perm}(\mathcal{A}_{\mathcal{G}_2}) = r_4 \rho_4 + r_6 \rho_6.
\]

5. External edges \(i_1\) and \(\omega_2\) are taken. There are two possibilities to choose the connection edges: either \(c_1\), therefore \(o_1\) and \(i_1\) are also taken, or \(c_2\), thus also \(o_2\) and \(i_2\) are taken. Hence

\[
\text{perm}(\mathcal{G}_1) = \text{perm}(\mathcal{G}_2) = \text{perm}(\mathcal{A}_{\mathcal{G}_1}) \text{perm}(\mathcal{A}_{\mathcal{G}_2}) = r_3 \rho_5 + r_5 \rho_4.
\]

6. External edges \(i_2\) and \(\omega_1\) are taken. There are two possibilities to choose the connection edges: either \(c_2\), therefore \(o_2\) and \(i_2\) are also taken, or \(c_1\), thus also \(o_1\) and \(i_1\) are taken. Hence

\[
\text{perm}(\mathcal{G}_1) = \text{perm}(\mathcal{G}_2) = \text{perm}(\mathcal{A}_{\mathcal{G}_1}) \text{perm}(\mathcal{A}_{\mathcal{G}_2}) = r_4 \rho_6 + r_6 \rho_3.
\]
since it represents a EQUALITY-gadget. But in order to be able to extend one gadget with another to form a new gadget, that is still planar, the gadgets must have the property to be extendable. And Conjecture 5.7 states, that all such exceptional gadgets, e.g., a gadget with a signature of \((1,2,2,2,2,2)\), are always only connectable, which makes the resulting new gadget non-planar.

5.1. A solvable counting problem. Question: Is there a counting problem, that can be solved with the gadgets that are realizable? This question aims to an analogous case as to the \#PTW-Mon-3CNF problem, that can “accidentally” be solved using holographic algorithms. The answer is Yes, there is at least one. One could count the solution of formulas from \textsc{Forest-3SAT}, as defined in Definition 1.14. Note, that this is not a new result, since in 2010 Samer and Szeider [12] also showed how to count this efficiently, but using a totally different approach. The reason why this class is countable is, that one does not need circular planarity according to both input and output pairs but only for one of those pairs. The other pair must be located in an arbitrary inner face of the embedding. Since the incidence graph is a forest, all further gadgets could be stored within the previous gadget, i.e., some kind of nesting has to be done. Since such gadgets do not contradict Conjecture 5.7, they can indeed be found. Consider the XOR-gadget in Figure 14 and its bipartite double cover in Figure 15. Its bipartite double cover is connectable over the nodes \((8,3)\) and all further drawing is done within the nested area.

The signature of this XOR-gadget is

\[(23) \quad S_{\mathfrak{G}^*} = (18,3,7,8,6,3) = (0,0,1,2,0,0)\]

The CLAUSE-gadget does not have the same kind of signature like, e.g., the XOR-gadget. It has a smaller kind of signature, that only care about the good cases, i.e., valid paths that can be taken by a cyclic cover. It must not care about what happens if the cyclic cover enters over input node \(i_1\) but leaves about the non associated
output node $o_3$. All this is handled by the XOR-gadget. For a CLAUSE-gadget with two outer edges, one could express this as the signature

$$S = (\neq 0, 0, \neq 0, \neq 0, *, *)$$

whereof * stands for an arbitrary value. Similar arguments hold for the VARIABLE-gadget as well.

Figure 16. The new CLAUSE-gadget $G^{**}$ with its input and output pairs (1, 5), (2, 6) and (3, 7).

Figure 17. The bipartite double cover of the new CLAUSE-gadget with the input output pairs (8, 5), (9, 6) and (10, 7).

**Theorem 5.11.** Let $\Phi$ be random boolean formula from $\#\text{FOREST-3SAT}$ with $n$ variables and $m$ clauses and let $G$ be the graph that is constructed from $\Phi$ according to Valiant’s proof, using the gadgets $G^{**}$ (see Figure 1), $G_2$ (see Figure 2), $G^*$ (see Figure 3). Then there exists an algorithm $A$ that decides $\Phi$’s satisfiability in randomized polynomial time $O(km^3)$ with an a success probability of

$$\Pr[s \leftarrow A(\Phi, k) | \Phi \text{ is sat} = s] = 1 - \frac{1}{3^k}$$

**Proof.** Note that a formula $\Phi$ from $\text{FOREST-3SAT}$ is always planar. Figure 18 shows the idea of how to construct such a graph. W.l.o.g. we assume that the incidence graph of $\Phi$ is connected and we start at an arbitrary clause node. Further note that the orientation of the external edges of each gadget (i.e. clockwise or anticlockwise) does not matter, since we can turn each connecting gadget into the correct orientation by flipping it around. This is due to the fact that the formula is from $\text{FOREST-3SAT}$. Otherwise, we would have to find several gadgets for each type (like the gadget $G_{RL}$ and $G_{LR}$) to take care of the orientations.

We now begin to redraw the incidence graph of the formula $\Phi$ using our new graph gadgets. For the current clause node we insert the new CLAUSE-gadget $G^{**}$. This CLAUSE-gadget has thee direction and in each direction there is a tree. W.l.o.g. we follow one arbitrary direction. In the example graph, e.g., assume we take the direction from $c_1$ to the variable node 1. Next, we pick the XOR-gadget $G^*$, turn it into the right direction and connect it to the CLAUSE-gadget. For the interior input/output pair of the XOR-gadget, we pick the VARIABLE-gadget $G_2$, turn into the right orientation and connect it to represent the variable 1 in the graph. Following the whole formula...
that way and connecting the gadgets as described, allows to obtain a bipartite double cover of $G$ and hence to apply Kasteleyn’s algorithm. Regarding this gadgets the total graph will be of

$$m|G^*| + \sum_{i=1}^{n}|G_2(x_i)| + 3m|G^*| = m \cdot 7 + \sum_{i=1}^{n}(2 + 2n_i) + 3m \cdot 7$$

$$= 2n + 2 \sum_{i=1}^{n}n_i + 28m$$

$$< 6m + 6m + 28m$$

$$< 40m$$

nodes. The bipartite double cover operations doubles the number of nodes, so we can bound the number of nodes from above via $|\hat{G}| \leq 80m$. Finally, Kasteleyn’s algorithm runs in time that is cubic in the number of nodes, which yields a total running time of our algorithm of $O(80^3m^3)$, or $O(m^3)$ for large $m$, for a single evaluation. Applying algorithm $B$ from the proof of Theorem 3.3 yields a total runtime of $O(km^3)$.

![Figure 18](image.png)  

**Figure 18.** On the left is the incidence graph of a small formula from FOREST-3SAT. On the right its drawing using our proof idea. The rectangles illustrates the XOR-gadget that is between each clause and variable gadget regarding the edge it crosses. Because of the forest property, all further gadgets can be nested into the latest XOR-gadget.

6. Concluding remarks

The paper shows up a general approach to count the solutions of a boolean planar formula if one finds appropriate gadgets, i.e. that do not contradict Conjecture 5.7, to build the Valiant graph. Currently, we only showed that $\#$FOREST-3SAT could be solved with this approach, but we only restrict our attention to a graph that is build in the same way as Valiant build its graph in his proof. Are there more special formula classes that could be addressed using another type of Valiant graph?

The conjecture itself is based on many computations and practical verifications and not at least on the very well believed assumption that $\text{RP} \neq \text{NP}$. Proving this conjecture is another open task for the future work in this direction.

7. Acknowledgements

Thanks goes to all my colleagues for fruitful discussion about the topic and to Peter Shor, who pointed out to me the existence of the Desnanot-Jacobi identity by answering one of my questions on cstheory.stackexchange.com.

References

[1] Scott Aaronson. A Linear-Optical Proof that the Permanent is $\#P$-Hard. To appear in the Proceedings of the Royal Society, 2011.
[2] Sanjeev Arora and Boaz Barak. Computational Complexity: A Modern Approach. Cambridge University Press, New York, NY, USA, 1st edition, 2009.
[3] Amir Ben-Dor and Shai Halevi. Zero-one permanent is $\#P$-complete, a simpler proof. In 2nd Israel Symposium on Theory of Computing Systems, pages 108–117, 1993. Natanya, Israel.
[4] Markus Blaeser and Holger Dell. Complexity of the cover polynomial. In 34th International Colloquium on Automata, Languages and Programming (ICALP), volume 4596 of Lecture Notes in Computer Science, pages 801–812, 2007. Wroclaw, Poland.
[5] Jin-Yi Cai and Pinyan Lu. Holographic Algorithms: From Art to Science. In Proceedings of the 39th Annual ACM Symposium on Theory of Computing, STOC ’07, pages 401–410. ACM, 2007. San Diego, California, USA.
[6] Rev. C. L. Dodgson. Condensation of determinants, being a brief method for computing their arithmetical values. *Proceedings of the Royal Society*, 15:150–155, 1966.

[7] P.W. Kasteleyn. Graph theory and crystal physics. *Graph Theory and Theoretical Physics*, pages 43–110, 1967.

[8] Donald Knuth. Nested satisfiability. In *Acta Informatica*, volume 28, pages 1–6. Springer, 1990.

[9] Jan Kratochvíl and Mirko Krivanek. Satisfiability of co-nested formulas. In *Acta Informatica*, volume 30, pages 397–403. Springer, 1993.

[10] D. Lichtenstein. Planar formulae and their uses. *SIAM Journal of Computing*, 11:329–343, 1982.

[11] Dan Roth. On the Hardness of Approximate Reasoning. In *Artificial Intelligence*, volume 82, pages 273–302. Elsevier, 1996.

[12] Marko Samer and Stefan Szeider. Algorithms for propositional model counting. *J. Discrete Algorithms*, 8(1):50–64, 2010.

[13] Leslie G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.

[14] Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal of Computation*, 8(3):410–421, 1979.

[15] Leslie G. Valiant. Accidental Algorithms. In *47th Annual IEEE Symposium on Foundations of Computer Science*, FOCS ’06, pages 509–517, 2006.

[16] Leslie G. Valiant. Holographic algorithms. *SIAM Journal of Computation*, 37(5):1565–1594, 2008.

[17] Leslie G. Valiant and Vijay V. Vazirani. Np is as easy as detecting unique solutions. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, STOC ’85, pages 458–463. ACM, 1985. Providence, Rhode Island, USA.

[18] Jin yi Cai, Pinyan Lu, and Mingji Xia. Holographic Algorithms with Matchgates Capture Precisely Tractable Planar #CSP. In *51st Annual IEEE Symposium on Foundations of Computer Science*, FOCS 2010, FOCS ’10, pages 427–436, 2010. Las Vegas, Nevada, USA.

Federal Office for Information Security, D-53179 Bonn, Germany

E-mail address: ChristianSchridde@googlemail.de