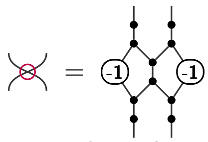
EITQ TD: Counting Complexity

Renaud Vilmart
renaud.vilmart@inria.fr

October 23, 2024

1 Perfect Matchings in Planar Graphs

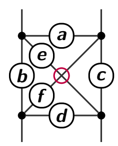
Question 1. Give a tensor that represents $|0\rangle$ (i.e. a tensor with 1 output edge, which should **never** be in a perfect matching). Give a tensor that represents $|1\rangle$ (i.e. a tensor with 1 output edge, which should **always** be in a perfect matching).

Question 2. Check that 

indeed has interpretation $|i, j\rangle \mapsto (-1)^{ij}|j, i\rangle$.

Hint: you can check all 4 cases where the input edges are assumed to be in a perfect matching or not.

Question 3. What is the interpretation of the usual swap? Is it a matchgate? Is it a *planar* matchgate?

Question 4. Compute the interpretation of 

. Show that <u>almost</u> any **planar** matchgate can be represented this way (up to a global scalar). Show that $X \otimes X$ (with X the usual Pauli gate) is a planar matchgate. Show that we can use $X \otimes X$ to deal with the "almost" above.

2 Detecting Cycles

Let $G = (V, E)$ be a **directed** graph.

Question 1. Consider a vertex $v \in V$ with only outgoing edges. How many cycles can go through this node? Same question for a vertex with only incoming edges.

These vertices cannot be part of a cycle.

Question 2. Suppose all the vertices in G have at least 1 incoming edge, and at least 1 outgoing edge. Does G necessarily have a cycle?

It does. Start building a path from a vertex v. At every step, take a random neighbour of the last vertex in the path, and add it to the path. If that vertex was already in the path, a cycle has been found, the algo can stop. Otherwise continue. As there are finitely many vertices, we will eventually reach a vertex that was already explored.

Question 3. From the previous observations, suggest an algorithm to detect the presence of a cycle in a directed graph. What is the complexity of this algorithm?

Remove leaves or co-leaves as long as the graph has some. At the end, if the graph is empty, it initially had no cycle. If vertices remain, the graph had a cycle.

3 Permanent and Cycle Covers

Let $G = (V, E, w)$ be a weighted **directed** graph, and M_G an adjacency matrix of G. A cycle cover of G is a subset C of edges E of G, such that for each vertex $v \in V$, there is exactly one edge $(v, \star) \in C$ and exactly one edge $(\star, v) \in C$.

We call #CC the function that computes the (weighted) number of cycle covers of a graph. Our goal in this exercise is to prove that:

$$\#\mathrm{CC}(G) = \mathrm{PERM}(M_G)$$

Recall the definition of the permanent:

$$\mathrm{PERM}(M) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} m_{i,\sigma(i)}$$

for all $n \times n$ matrices $M = (m_{i,j})$, with S_n the set of permutations over $\{1, ..., n\}$.

Question 1. Show the Laplace expansion formula: for any i:

$$\mathrm{PERM}(M) = \sum_{j=1}^{n} m_{i,j}\, \mathrm{PERM}(M_i^j)$$

where M_i^j is matrix M without row i and column j (called minor of M).

First notice that for a given i, in each term there is a single j such that $m_{i,j}$ is a factor. We can hence partition the terms into those that have $m_{i,1}$ as a factor, those that have $m_{i,2}$ as a factor, ... Consider the terms with $m_{i,j}$, which we can factorise with

$$m_{i,j} \sum_{\substack{\sigma \in S_n \\ \sigma(i)=j}} \prod_{k \neq i} m_{k,\sigma(k)} = m_{i,j}\, \mathrm{PERM}(M_i^j)$$

Question 2. Consider the outgoing edges of a vertex v of G. Reasoning on the (number of) cycles that go through v, determine a way to express the number of cycle covers of G from the number of cycle covers of graphs built from G, with one fewer vertex.

Let $(v, u_1), ..., (v, u_n)$ be the n outgoing edges from v. The number of cycles that go through v are the number of cycles that go through edge (v, u_1) plus the nb of cycles that go through (v, u_2) plus ... plus the nb of cycles that go through (v, u_n). The number of cycles that go through (v, u_i) is the number of cycles in the graph obtained by: 1) removing incoming edges of u_i except (v, u_i), 2) contracting edge (v, u_i) and 3) removing edges (v, u_j) for $j \neq i$. We can hence compute the number of cycle covers in G as a sum of cycle covers in smaller graphs obtained from G.

Question 3. Relate this with the Laplace expansion formula on M_G, and show that $\#\mathrm{CC}(G) = \mathrm{PERM}(M_G)$.

When expressed in terms of adjacency matrices, the above decomposition is exactly the Laplace expansion formula. The base case being true ($\#\mathrm{CC}(G) = \mathrm{PERM}(M_G)$ when G contains a single vertex), this proves the result by induction.

4 Permanent and Bipartite Perfect Matchings

Let $G = (V_0, V_1, E, w)$ be a weighted **bipartite** graph with $|V_0| = |V_1|$, and M_G be a biadjacency matrix of G. Our goal here is to prove that:

$$\#\mathrm{PM}(G) = \mathrm{PERM}(M_G)$$

Question 1. Considering a vertex $v \in V_0$, and reasoning on the different ways to cover v by a perfect matching, determine a way to compute $\#\mathrm{PM}(G)$ from the number of perfect matchings of simpler graphs derived from G.

Let $\{u_1, ..., u_n\}$ be the neighbours of v. The different perfect matchings of G can be partitioned into the ones that contain (v, u_1), the ones that contain (v, u_2), ..., the ones that contain (v, u_n). The number of perfect matchings that contain (v, u_i) is exactly the number of perfect matchings in the graph obtained by removing vertices v and u_i (and associated edges). Notice that the obtained graph is still bipartite. We can hence compute the number of perfect matchings in G as a sum of perfect matchings in smaller bipartite graphs obtained from G.

Question 2. Relate this to the Laplace expansion formula on M_G, and show that $\#\mathrm{PM}(G) = \mathrm{PERM}(M_G)$.

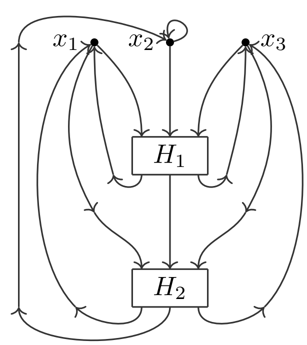
Similarly as the cycle covers case, when expressed in terms of biadjacency matrices, the above decomposition is exactly the Laplace expansion formula. The base case being true ($\#\mathrm{CC}(G) = \mathrm{PERM}(M_G)$ when G contains only 2 vertices), this proves the result by induction.

5 Cycle Covers and SAT Solutions

The objective here is to find a polynomial reduction from #3-SAT to #CC. For every 3-SAT formula φ, we want to build a graph G_φ such that it is easy to recover #3-SAT(φ) from #CC(G_φ). Consider we have a gadget (piece of graph) H that represents a clause. It has 3 incoming edges, and 3 outgoing edges. We build G_φ as follows:

- create one vertex v_i for each variable x_i of φ
- create one clause-gadget H_i for each clause C_i of φ
- for each variable x_i, create two "cycles" that go through v_i. One represents the "True" valuation for x_i, and the other the "False" valuation. If $C_i = \ell_{i_1} \vee \ell_{i_2} \vee \ell_{i_3}$, then if $\ell_{i_1} = x_{i_1}$ the **"True"** cycle of v_{i_1} will enter gadget H_i through the first incoming edge, and exit the gadget through the 1st outgoing edge; if $\ell_{i_1} = \neg x_{i_1}$ the **"False"** cycle of v_{i_1} will enter gadget H_i through the first incoming edge, and exit the gadget through the 1st outgoing edge. We do the same for all clauses, and all literals in the clauses.

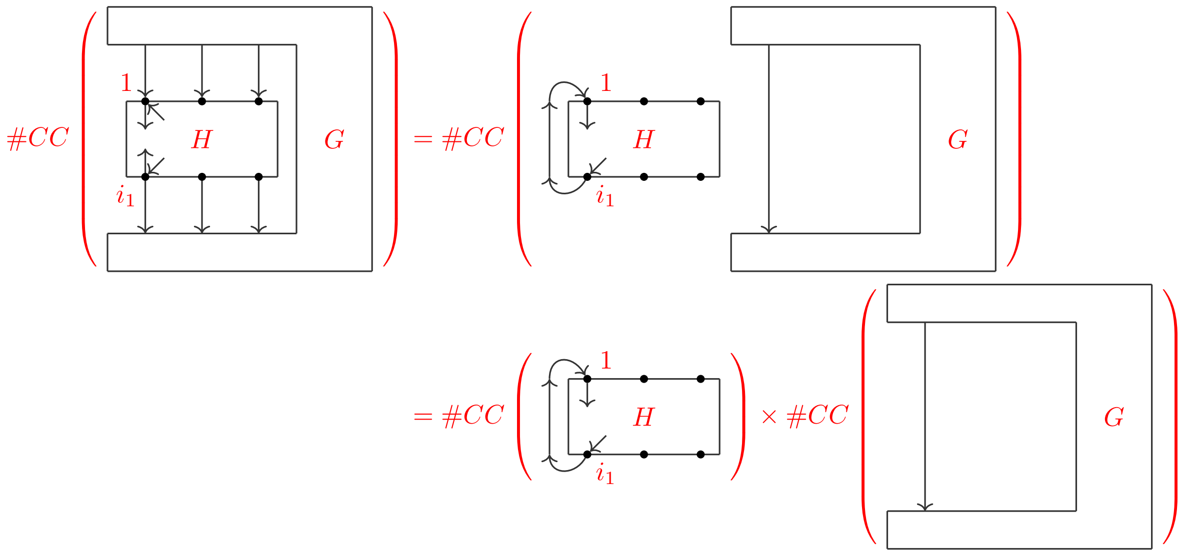
For instance, if $\varphi = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$, graph G_φ looks as follows:



Suppose the gadget H has adjacency graph M, with incoming wires to vertices (columns) 1, 2 and 3, and outgoing edges from vertices i_1, i_2 and i_3. We ideally want that incoming edge 1 is in the cycle cover iff outgoing edge 1 is in the cycle cover (and similarly for incoming/outgoing edges 2 and 3). But we also want that at least one of the incoming/outgoing edges is in the cycle cover (to represent clauses). We will use negative weights to make sure that the unwanted configurations cancel out.

Question 1. When none of the incoming/outgoing edges are in the cycle cover, you want the contribution to be 0. What does that mean for the permanent of M? When only the first incoming/outgoing pair of edges is in the cycle cover, the contribution is some non-zero constant c. What does that mean for the permanent of the matrix $M^1_{i_1}$? Express similarly all constraints on the permanent of minors of M.

When none of the incoming/outgoing edges of H are in the cycle cover, the overall number of cycle covers is the number of cycle covers inside H times the number of cycle covers of the rest of the graph. We can force this to be 0 by imposing that the contribution of H is 0, that is $\mathrm{PERM}(H) = 0$.
When there is a single cycle going through 1 and i_1, we are in the following situation:



where we assumed the only incoming edge of H used in a cycle covers was that of vertex 1, and the only outgoing edge of H used in the cycle covers was that of vertex i_1. We are now interested in the contribution of the gadget H to the number of cycle covers. If $i = i_1$, the only cycle that can go through 1 is the self loop, and the contribution of H is simply the number of cycle covers in H without vertex 1, which is nothing but $\mathrm{PERM}(M^1_{i_1})$. If $i \neq i_1$, we can contract the edge (i_1, i) without changing the number of cycle covers (we already removed other incoming edges from 1 and outgoing edges from i_1). The resulting quantity is again $\mathrm{PERM}(M^1_{i_1})$. Ideally, this number would be 1, so as to directly compute the number of solutions to φ by computing the number of cycle covers in G_φ. If it is instead some constant c, this is enough. Hence, we want $\mathrm{PERM}(M^1_{i_1}) = c$. Similarly, we want $\mathrm{PERM}(M^2_{i_2}) = c$, $\mathrm{PERM}(M^3_{i_3}) = c$, $\mathrm{PERM}(M^{12}_{i_1 i_2}) = c$, $\mathrm{PERM}(M^{13}_{i_1 i_3}) = c$, $\mathrm{PERM}(M^{23}_{i_2 i_3}) = c$, $\mathrm{PERM}(M^{123}_{i_1 i_2 i_3}) = c$, and all other permanents of minors of M on columns in $\{1, 2, 3\}$ and rows $\{i_1, i_2, i_3\}$ to be zero.

Question 2. Show that the following matrix satisfies the above constraints with constant $c = 12$, and with $i_1 = 5$, $i_2 = 4$ and $i_3 = 3$:

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

This can (and should) be checked with the help of a computer.

Question 3. Conclude by showing that #3-SAT $\preccurlyeq_{\#}$ $\#\mathrm{CC}_{\{-1,1,2,3\}}$.

With m the number of clauses in φ, we actually compute $\#CC(G_\varphi) = 12^m \#SAT(\varphi)$, so we have to divide the result of #CC by 12^m to get the answer to #3-SAT. This step is polynomial-time. Moreover, G_φ is obtained in polynomial time in function of φ. We hence have a polynomial counting reduction.

6 #P-completeness of Counting Perfect Matchings

Question 1. Assuming that #3-SAT is **#P-complete**, putting everything back together, show that #PM is **#P-complete**.

From the above, we have #3-SAT $\preccurlyeq_{\#}$ #CC$_{\{-1,1,2,3\}}$ $\preccurlyeq_{\#}$ PERM$_{\{-1,1,2,3\}}$ $\preccurlyeq_{\#}$ #PM$_{\{-1,1,2,3\}}$. We can also get rid of weights -1, 2 and 3 as explained in the slides of the lecture. So we also have #PM$_{\{-1,1,2,3\}}$ $\preccurlyeq_{\#}$ #PM. Reductions being transitive, we get #3-SAT $\preccurlyeq_{\#}$ #PM. Assuming that #3-SAT is **#P-complete**, this makes #PM a **#P-hard** problem. This problem in also in **#P**. Indeed, we can build a non-deterministic Turing machine that decides PM. Hence, #PM $\in$ **#P-complete**.