Computational Complexity Theory

Lecture 26-27: 0/1-Perm is #P-complete

Department of Computer Science,
Indian Institute of Science

Recap: Class #P

- Definition. We say a function $f: \{0,1\}^* \rightarrow \mathbb{N}$ is in #P if there's a poly-time TM M and a polynomial function $p: \mathbb{N} \rightarrow \mathbb{N}$ such that for every $x \in \{0,1\}^*$,

$$f(x) = \left|\{u \in \{0,1\}^{p(|x|)} : M(x, u) = 1\}\right| .$$

- Observation. Problems #SAT, #HAMCYCLE, #PerfectMatching, #CYCLE, #PATH and #SPANTREE are in #P.
- In fact, with every language in NP we can associate a counting problem that is in #P.

Recap: #P-completeness

- Definition. A function $f: \{0,1\}^* \rightarrow \mathbb{N}$ is in #P-complete if f is in #P and for every $g \in$ #P, we have $g \in FP^f$ i.e., g is poly-time Cook/Turing reducible to f.

- In other words, for every $x \in \{0,1\}^*$, we can compute $g(x)$ in polynomial time using *oracle access* to f.

- Observation. If a #P-complete language is in FP then #P = FP.

0/1-Permanent is #P-complete

- Theorem. *(Valiant 1979)* 0/1-Perm is #P-complete.

- It implies that #PerfectMatchings is #P-complete.

- If $X = (x_{ij})_{i,j\in n}$ then $\mathrm{Perm}(X) = \sum_{\sigma\in S_n} \prod_{i\in[n]} x_{i\,\sigma(i)}$.

- Note. If B_G is the biadjacency matrix of a bipartite graph G, then $\mathrm{Perm}(B_G)$ = #PerfectMatchings(G).

0/1 matrix

0/1-Permanent is #P-complete

- Theorem. *(Valiant 1979)* 0/1-Perm is #P-complete.

- Theorem. *(Jerrum, Sinclair, Vigoda 2001)* Permanent of a square matrix with non-negative entries has a FPRAS.

0/1-Permanent is #P-complete

- Theorem. *(Valiant 1979)* 0/1-Perm is #P-complete.
- Proof. 0/1-Perm is in #P. (Why?)

0/1-Permanent is #P-complete

- Theorem. *(Valiant 1979)* 0/1-Perm is #P-complete.

- Proof. We'll show that #3SAT $\in$ $FP^{0/1\text{-Perm}}$.

- In fact, we'll give a poly-time "Karp-like" reduction from #3SAT to 0/1-Perm, i.e., we'll give a poly-time computable function that maps a 3CNF ϕ to a 0/1-matrix A_ϕ s.t. $\#\phi$ is efficiently computable from A_ϕ.

- This means only <u>one query</u> to the 0/1-Perm oracle is required.

Graph theoretic interpretation of Perm

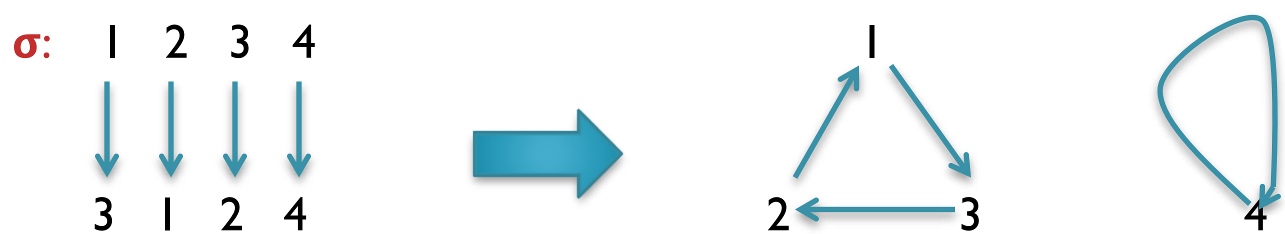
- Let $A = (a_{ij})_{i,j \in r}$, where $a_{ij} \in \mathbb{R}$.
- Then, $\mathrm{Perm}(A) = \sum_{\sigma \in S_r} \prod_{i \in [r]} a_{i\,\sigma(i)}$.
- Let G be the weighted digraph on r vertices with adjacency matrix A, i.e., the edge (i, j) in G has weight a_{ij}.

Graph theoretic interpretation of Perm

- Let $A = (a_{ij})_{i,j \in r}$, where $a_{ij} \in R$.
- Then, $\mathrm{Perm}(A) = \sum_{\sigma \in S_r} \prod_{i \in [r]} a_{i\,\sigma(i)}$.

- Let G be the weighted digraph on r vertices with adjacency matrix A, i.e., the edge (i, j) in G has weight a_{ij}.

- Every permutation $\sigma\colon [r] \rightarrow [r]$ can be expressed (<u>uniquely</u>) as a product of disjoint <u>cycles</u>.



Graph theoretic interpretation of Perm

- Definition. A *cycle cover* of a digraph G is a subgraph of G having in-degree and out-degree of every vertex exactly 1, i.e., the subgraph is a disjoint union of cycles covering all the vertices of G.

- *Weight* of a cycle cover C, denoted wt(C), is defined as the product of the weights of the edges in C.

Graph theoretic interpretation of Perm

- **Definition.** A *cycle cover* of a digraph G is a subgraph of G having in-degree and out-degree of every vertex exactly 1, i.e., the subgraph is a disjoint union of cycles covering all the vertices of G.

- *Weight* of a cycle cover C, denoted $wt(C)$, is defined as the product of the weights of the edges in C.

- **Observation.** $$\mathrm{Perm}(A) = \sum_{\substack{C:\ C \text{ is cycle} \\ \text{cover of } G}} wt(C) .$$

Every "contributing" permutation σ corresponds to a cycle cover C and vice versa.

Graph theoretic interpretation of Perm

- Definition. A <u>*cycle cover*</u> of a digraph G is a subgraph of G having in-degree and out-degree of every vertex exactly 1, i.e., the subgraph is a disjoint union of cycles covering all the vertices of G.

- <u>*Weight*</u> of a cycle cover C, denoted $wt(C)$, is defined as the <u>product</u> of the weights of the edges in C.

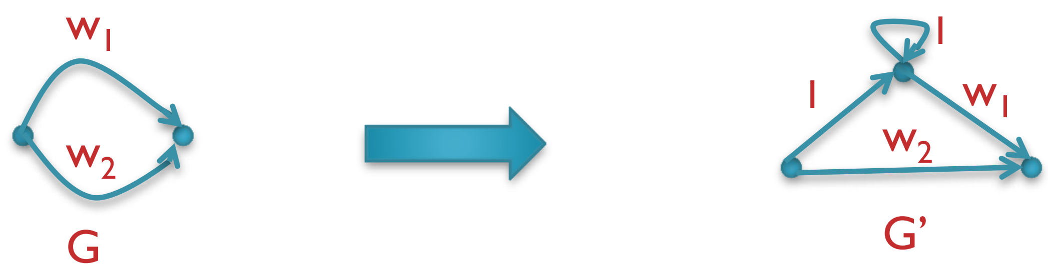
We can denote A as A_G, the adjacency matrix of G

- Observation. $$\mathrm{Perm}(A) = \sum_{C:\ C \text{ is cycle cover of } G} wt(C).$$

Every "contributing" permutation σ corresponds to a cycle cover C and vice versa.

Graph with parallel edges

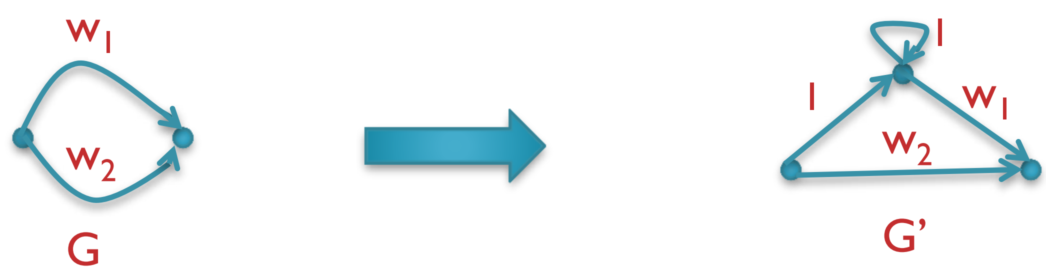
- Note. We can talk about "adjacency matrix" of a graph G that has <u>parallel edges</u> by defining a new graph G':



- Denote the adjacency matrix of a graph H (without parallel edges) by A_H. Then, A_G is defined as $A_{G'}$.

Graph with parallel edges

- **Note.** We can talk about "adjacency matrix" of a graph G that has <u>parallel edges</u> by defining a new graph G':



- Denote the adjacency matrix of a graph H (without parallel edges) by A_H. Then, A_G is defined as $A_{G'}$.

- **Observation.** $$\sum_{\substack{C:\ C \text{ is cycle} \\ \text{cover of } G}} wt(C) = \sum_{\substack{C:\ C \text{ is cycle} \\ \text{cover of } G'}} wt(C).$$

0/1-Permanent is #P-complete

- Theorem. *(Valiant 1979)* 0/1-Perm is #P-complete.
- Proof. Let ϕ be a 3CNF that has n variables and m clauses. Assume that every clause has <u>exactly</u> 3 literals.

- Step 1: From ϕ we'll form a graph $H = H_\phi$ that has edge weights in $\{-1, 0, 1, 2, 3\}$ such that

$$\mathrm{Perm}(A_H) = \sum_{C:\ C \text{ is cycle cover of } H} \mathrm{wt}(C) = 4^{3m}.\#\phi\,. \qquad \textbf{... Eqn (1)}$$

0/1-Permanent is #P-complete

- Theorem. *(Valiant 1979)* 0/1-Perm is #P-complete.
- Proof. Let ϕ be a 3CNF that has n variables and m clauses. Assume that every clause has <u>exactly</u> 3 literals.

- Step 1: From ϕ we'll form a graph $H = H_\phi$ that has edge weights in $\{-1, 0, 1, 2, 3\}$ such that

$$\mathrm{Perm}(A_H) = \sum_{C:\ C \text{ is cycle cover of } H} \mathrm{wt}(C) = 4^{3m}.\#\phi . \qquad \text{... Eqn (1)}$$

- Note. Eqn (1) <u>doesn't</u> give a FPRAS for #3SAT as the FPRAS for Perm is for matrices with <u>*non-negative entries*</u>.

0/1-Permanent is #P-complete

- Theorem. *(Valiant 1979)* 0/1-Perm is #P-complete.
- Proof. Let ϕ be a 3CNF that has n variables and m clauses. Assume that every clause has <u>exactly</u> 3 literals.

- Step 2: We'll process H further to get a new graph $G = G_\phi$ with edge weights in $\{0,1\}$ such that $\#\phi$ can be efficiently computed from $\mathrm{Perm}(A_G)$.

- However, unlike Eqn (1), we won't get an "precise" equation relating $\mathrm{Perm}(A_G)$ and $\#\phi$.

Step 1: Construction of H

- Convention. In the figures, edges without labels have weight 1, and missing edges have weight 0.

- H will be constructed using 3 kinds of *gadgets* (graphs):

Step 1: Construction of H

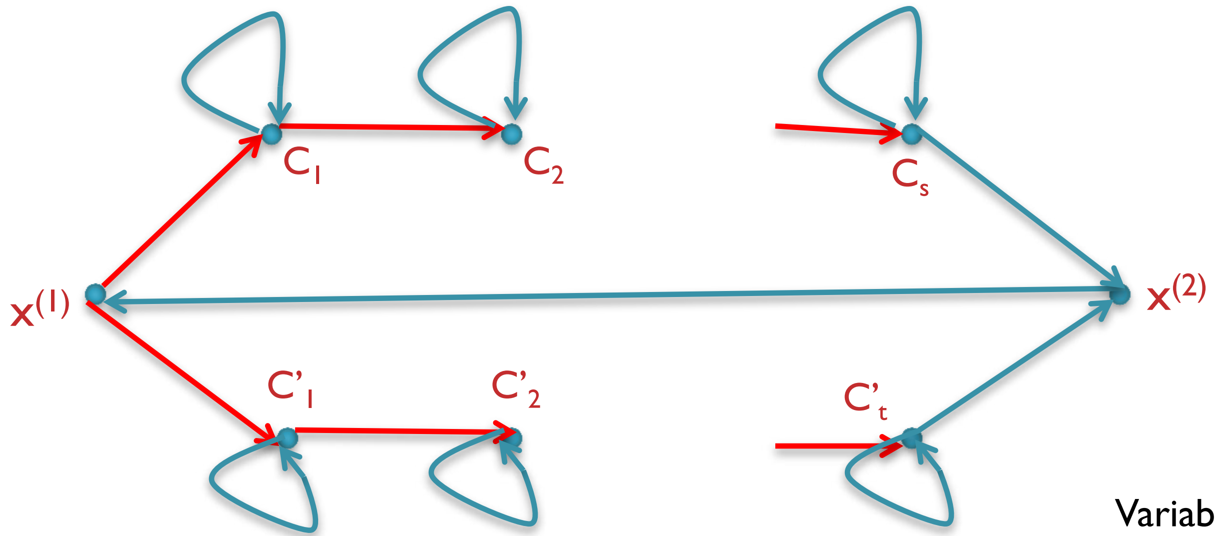
- Convention. In the figures, edges without labels have weight 1, and missing edges have weight 0.

- H will be constructed using 3 kinds of *gadgets* (graphs):
  - ➢ Variable gadgets (there will be n of them),
  - ➢ Clause gadgets (there will be m of them), and
  - ➢ XOR gadgets.

- XOR gadgets are cleverly constructed 4-vertex graphs which will be used to connect variable gadgets with clause gadgets.

A variable gadget

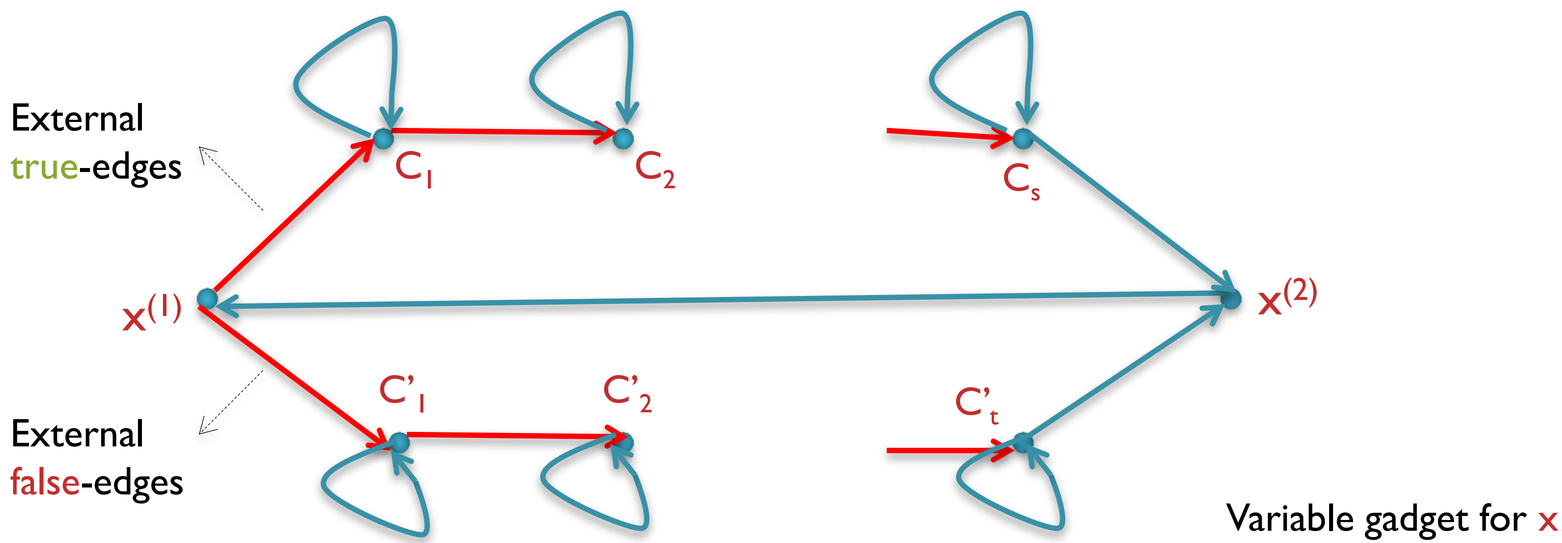
- Let x be a variable. C_1, …, C_s be the clauses in which x appears, and C'_1,…,C'_t the clauses in which ¬x appears.



Variable gadget for x

A variable gadget

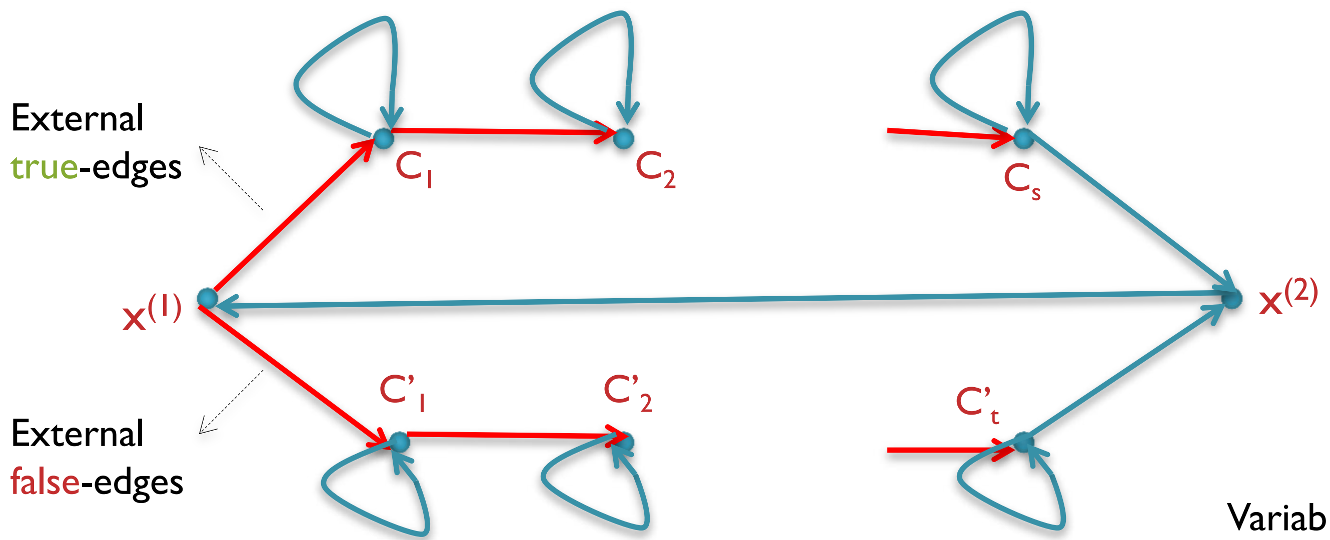
- Let x be a variable. $C_1, \ldots, C_s$ be the clauses in which x appears, and $C'_1,\ldots,C'_t$ the clauses in which ¬x appears.



- The external edges (i.e., the red edges) will <u>not</u> be present in H, they will be used to connect to the Clause gadgets via the XOR gadgets.

A variable gadget

- Let x be a variable. C_1, …, C_s be the clauses in which x appears, and $C'_1,\ldots,C'_t$ the clauses in which ¬x appears.

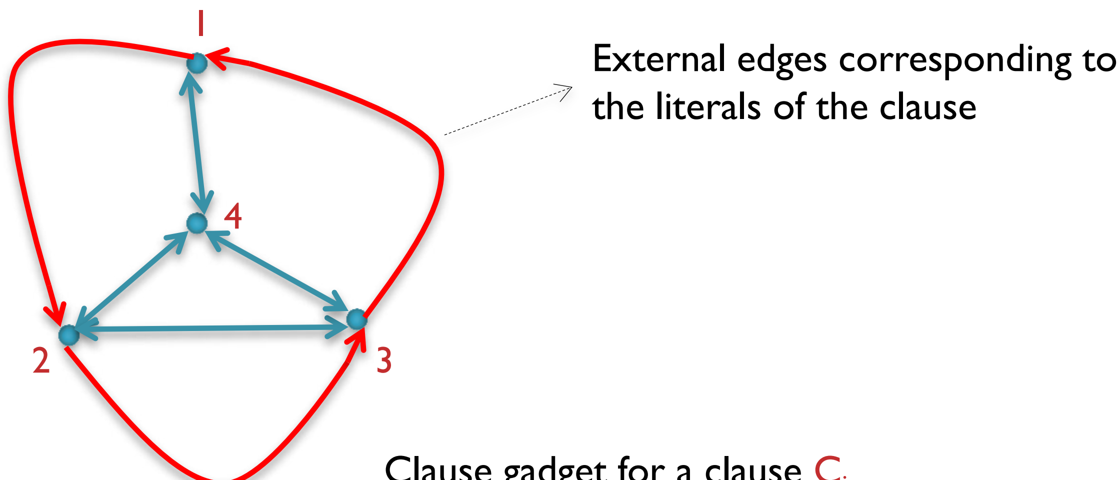


Variable gadget for x

- Observation 1. A variable gadget has exactly 2 cycle covers corresponding to 0/1 assignment to the variable.

A clause gadget

- Has 4 vertices and 3 external edges (i.e., red edges) corresponding to the 3 literals of the clause.

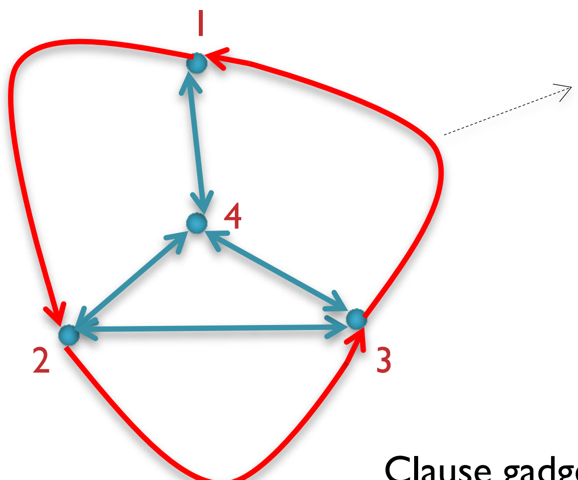


Clause gadget for a clause C_i

- External edges will <u>not</u> be present in H, they will be used to connect to the Variable gadgets via the XOR gadgets.

A clause gadget

- Has 4 vertices and 3 external edges (i.e., red edges) corresponding to the 3 literals of the clause.



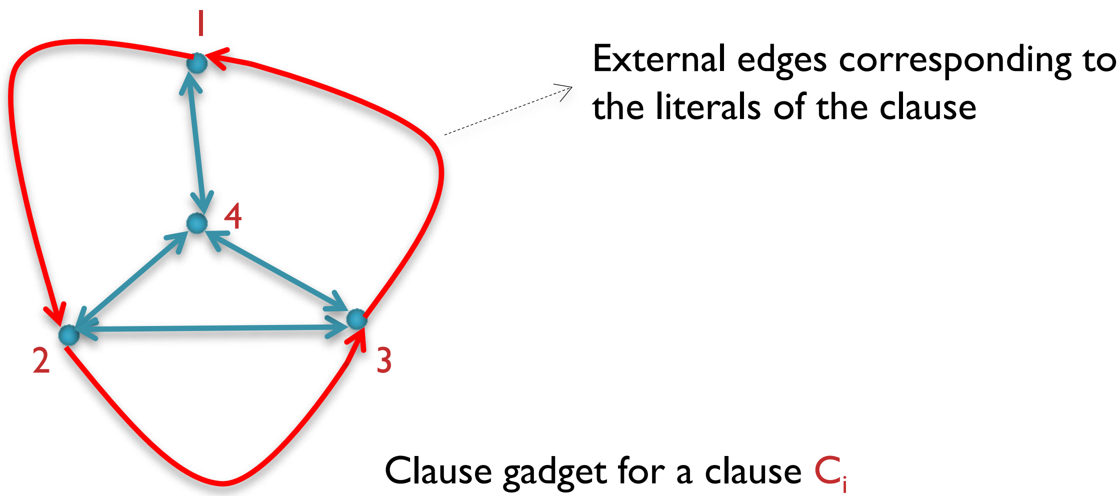
External edges corresponding to the literals of the clause

Clause gadget for a clause C_i

- Observation 2a. The only possible cycle covers of a clause gadget are those that <u>exclude</u> at least one external edge.

A clause gadget

- Has 4 vertices and 3 external edges (i.e., red edges) corresponding to the 3 literals of the clause.



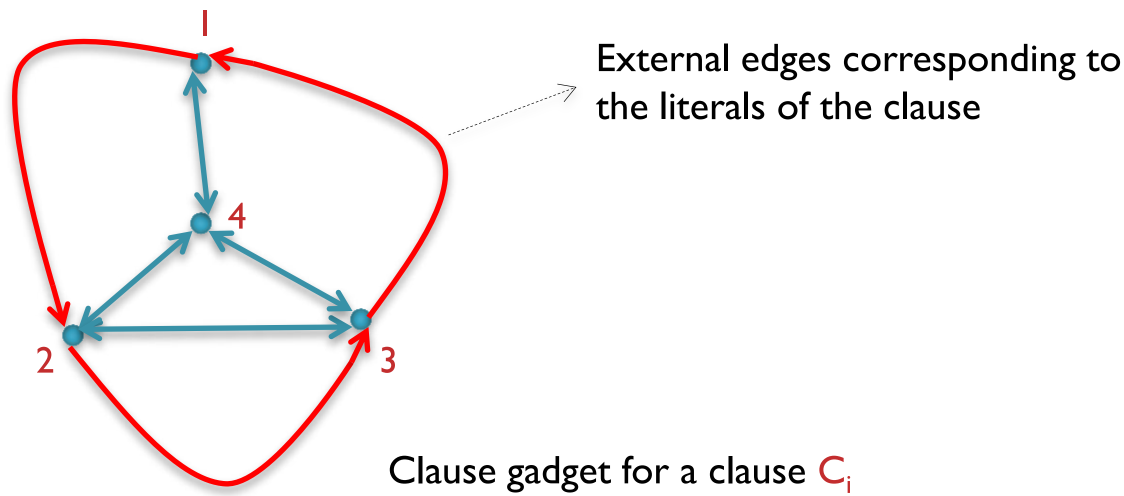
Clause gadget for a clause C_i

- Observation 2a. The only possible cycle covers of a clause gadget are those that <u>exclude</u> at least one external edge.

Excluding an external edge will indicate that the corresponding literal is set to 1.

A clause gadget

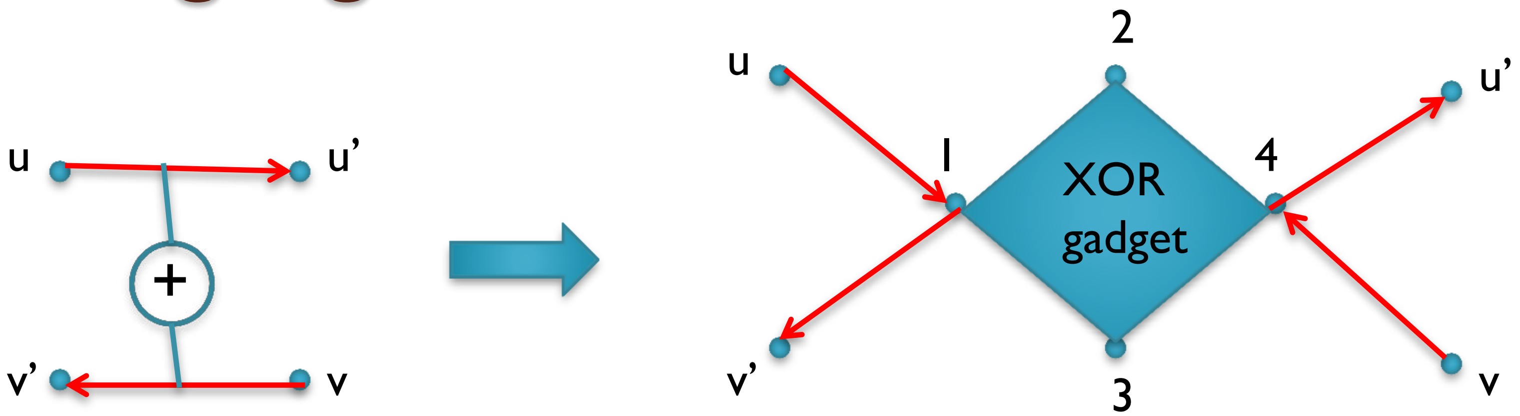
- Has 4 vertices and 3 external edges (i.e., red edges) corresponding to the 3 literals of the clause.



Clause gadget for a clause C_i

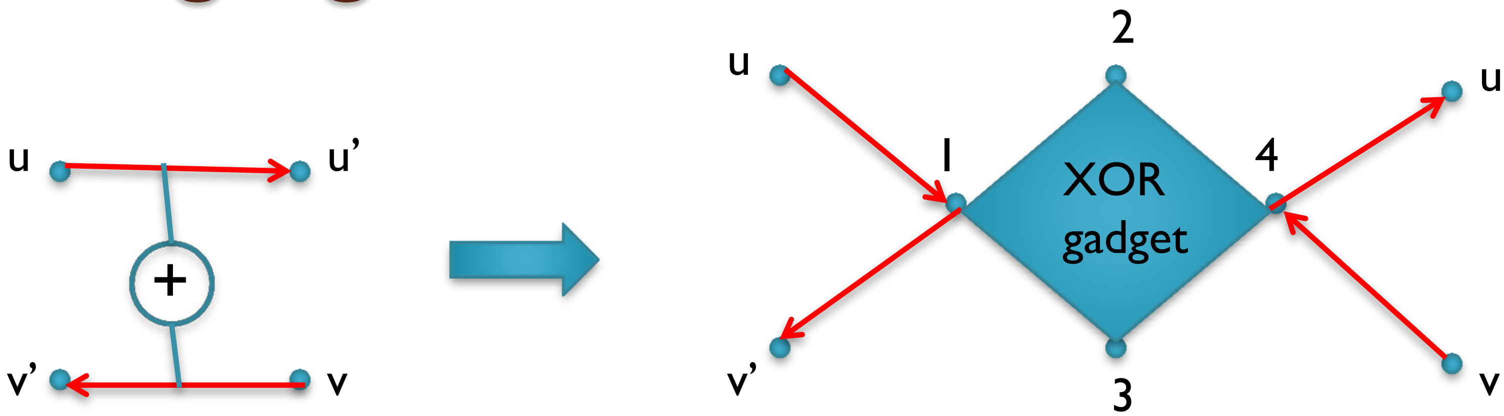
- Observation 2b. For any given proper subset of the 3 external edges, there's a unique cycle cover (of weight 1) that contains them.

XOR gadget



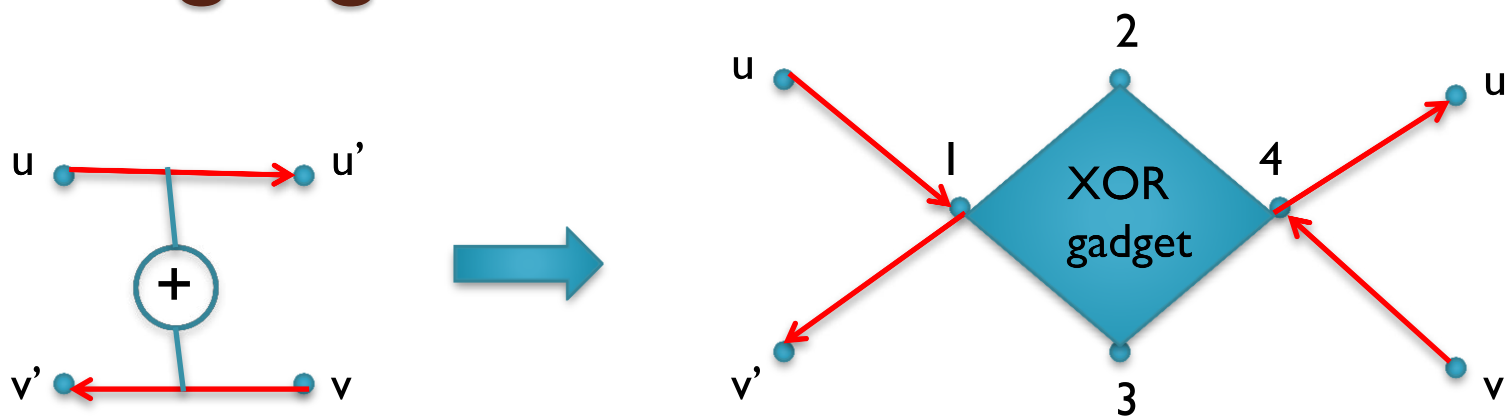
- We'll construct an XOR gadget such that the following features are satisfied:

XOR gadget



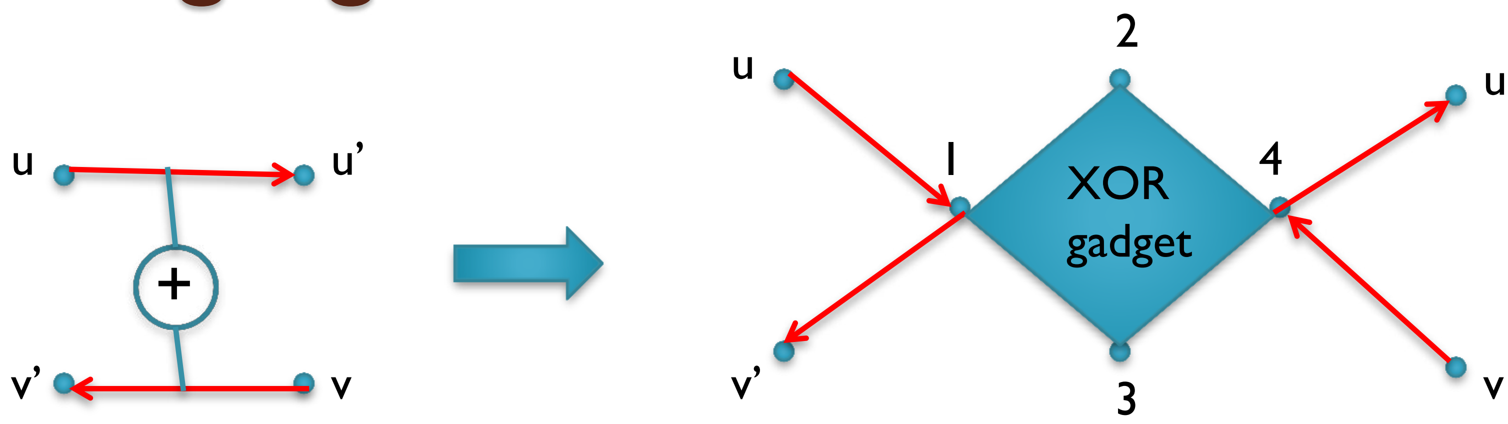
- We'll construct an XOR gadget such that the following features are satisfied:
  - Feature 1: Consider cycle covers of H that contain a <u>fixed</u> set of edges outside the XOR gadget but contain <u>none</u> of (u,1), (1,v'), (v,4), (4,u'). The sum of the weights of all such cycle covers is 0.

XOR gadget



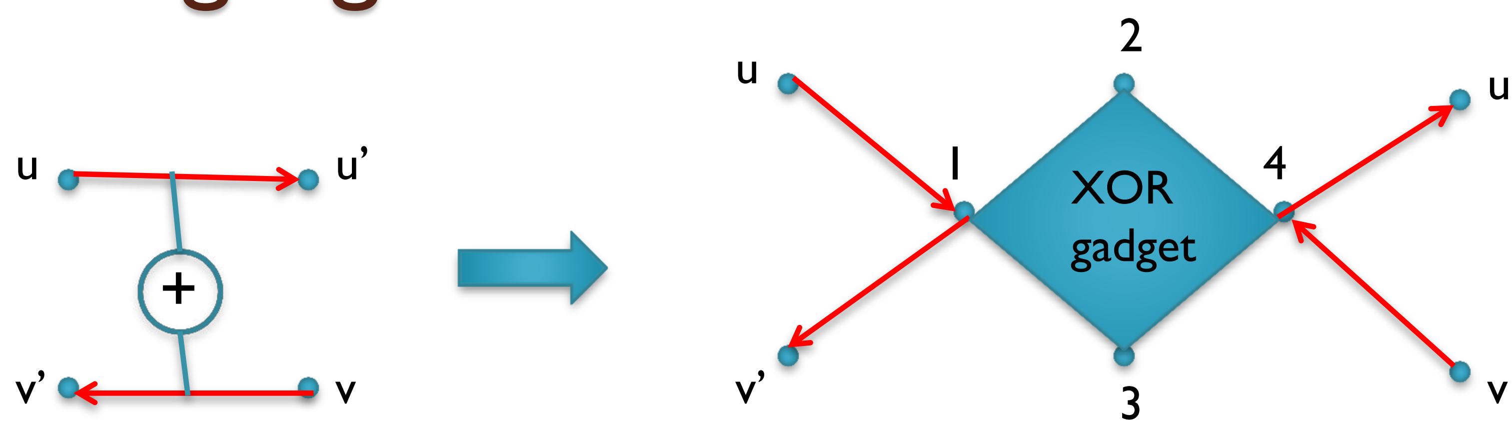
- We'll construct an XOR gadget such that the following features are satisfied:
  - Feature 2: Consider cycle covers of H that contain a <u>fixed</u> set of edges outside the XOR gadget including <u>*at least one*</u> of the pairs ((u,1), (1,v')) and ((v,4), (4,u')). The sum of the weights of all such cycle covers is 0.

XOR gadget



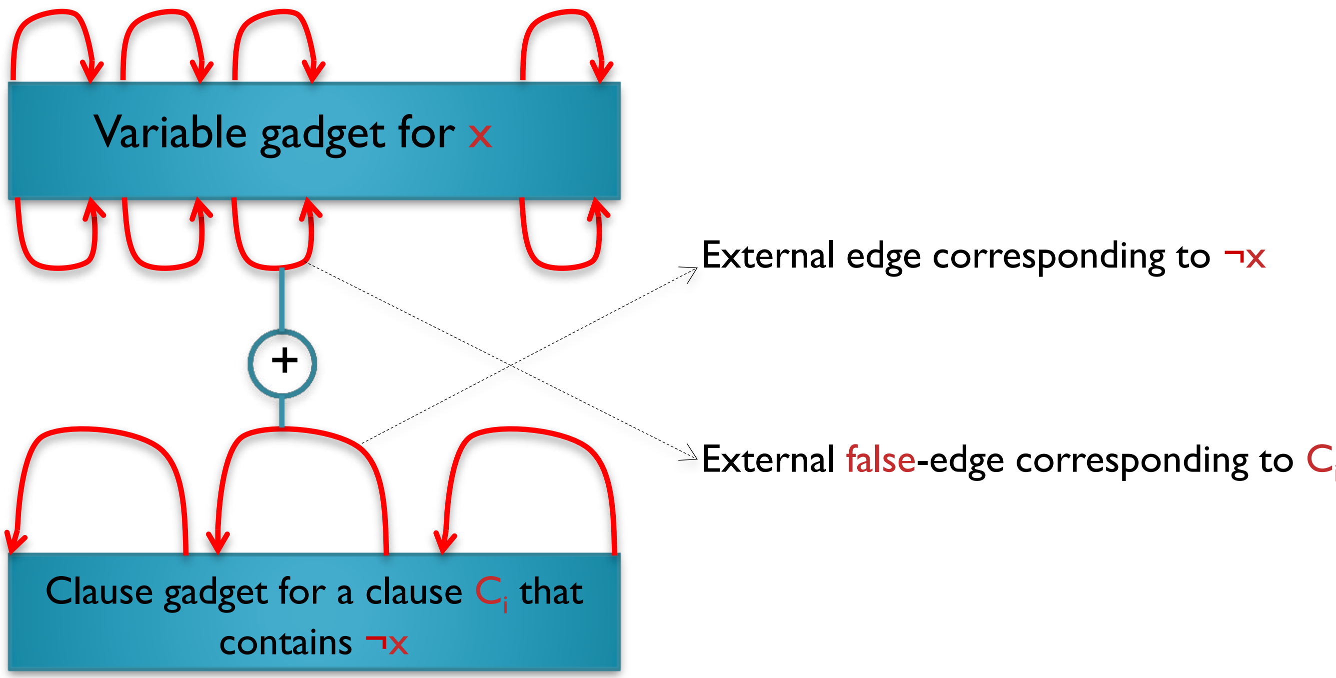
- We’ll construct an XOR gadget such that the following features are satisfied:
  - Feature 3: Consider cycle covers of H that contain a fixed set of edges outside the XOR gadget including (u,1), (4,u’) but not (v,4), (1,v’). The sum of the weights of all such cycle covers is 4.(product of the weights of the fixed set of edges).

XOR gadget



- We'll construct an XOR gadget such that the following features are satisfied:
    - Feature 4: Consider cycle covers of H that contain a fixed set of edges outside the XOR gadget including (v,4), (1,v') but not (u,1), (4,u'). The sum of the weights of all such cycle covers is 4.(product of the weights of the fixed set of edges).

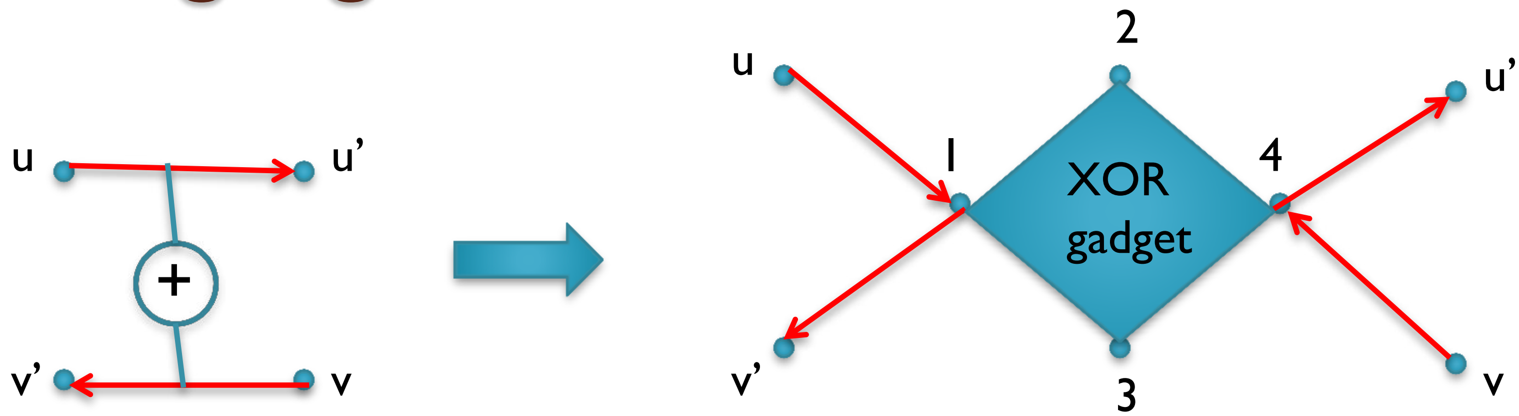
Construction of H



- Size(H) = poly(n,m).

- There are 3m XOR gadgets in H. Every cycle cover of H "touches" the 3m XOR gadgets.

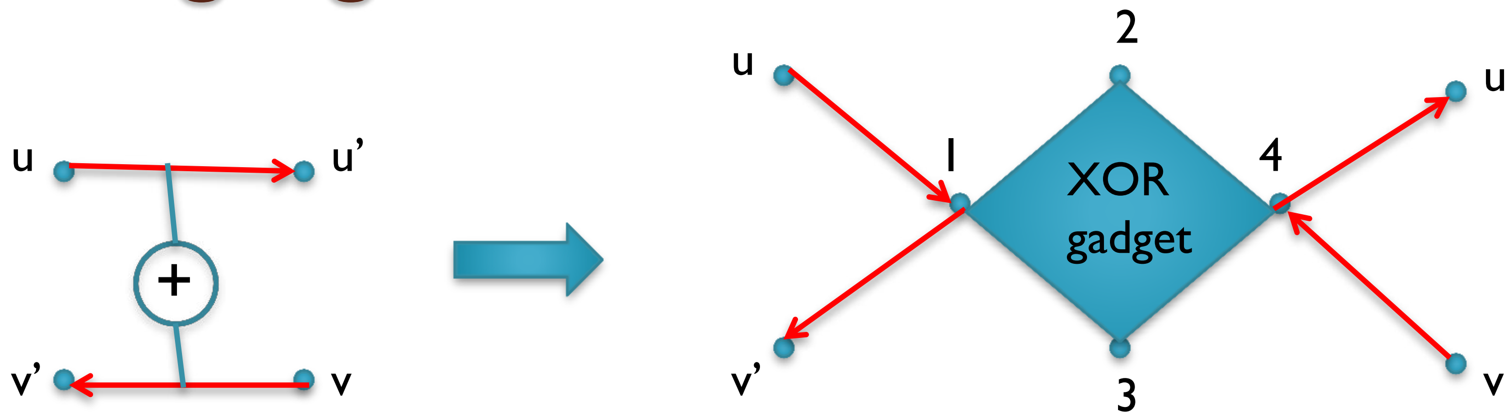
XOR gadget



- An XOR gadget can be "touched" in 4 possible ways:
  a. None of (u,1), (1,v'), (v,4), (4,u'),
  b. At least one of the pairs ((u,1),(1,v')) & ((v,4),(4,u')),
  c. Only (u,1), (4,u'),
  d. Only (v,4), (1,v').

Call these the "<u>touching patterns</u>" of an XOR gadget.

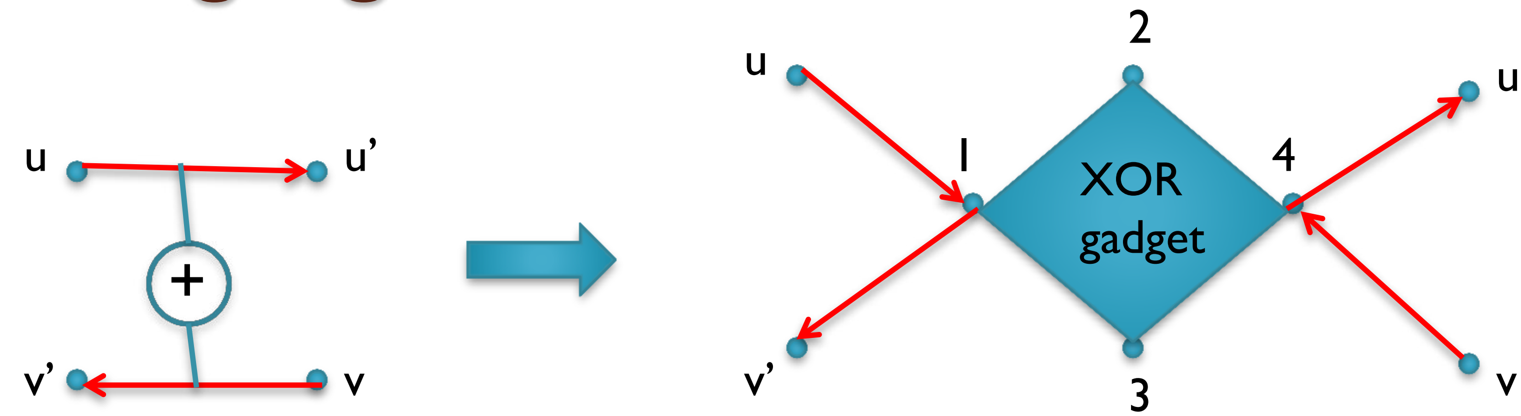
XOR gadget



- Every cycle cover of H can be mapped to a specific choice of the "touching patterns" of the 3m XOR gadgets.

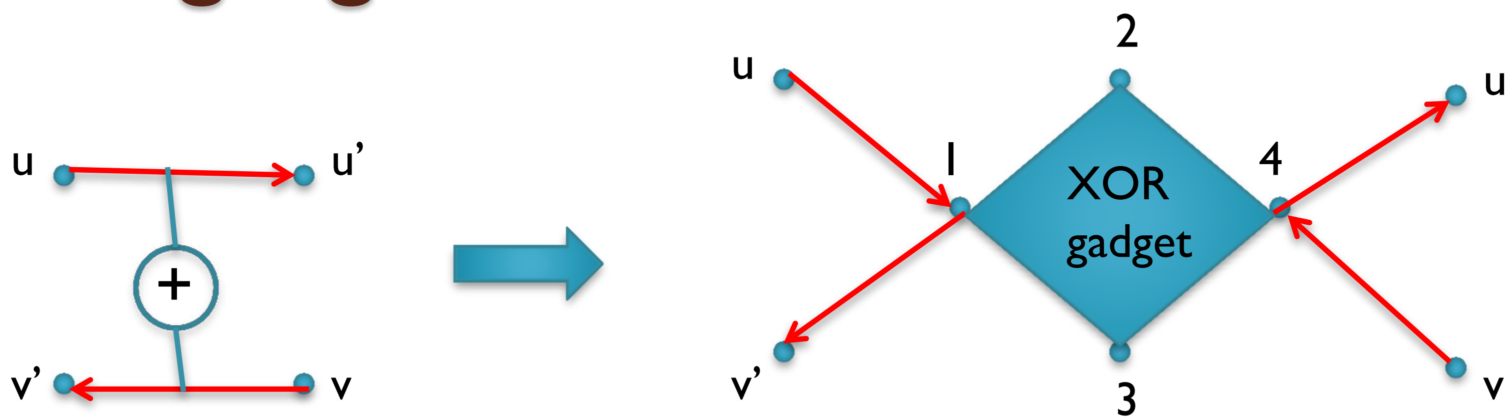
- Now, let us examine the sum of the weights of all the cycle covers of H.

XOR gadget



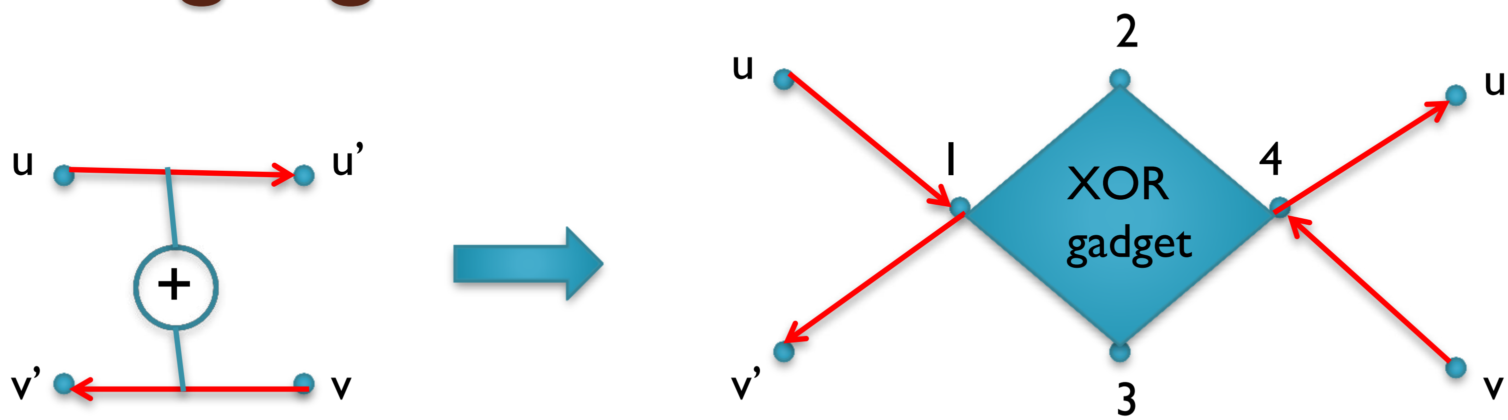
- Claim 1a. Cycle covers, which map to a <u>specific</u> choice of the "touching patterns" of the XOR gadgets s.t. the "touching pattern" of <u>at least one</u> of the XOR gates is of type a, <u>**do not**</u> contribute to the final sum.
- Proof. Follows from Feature 1. *(Homework)*

XOR gadget



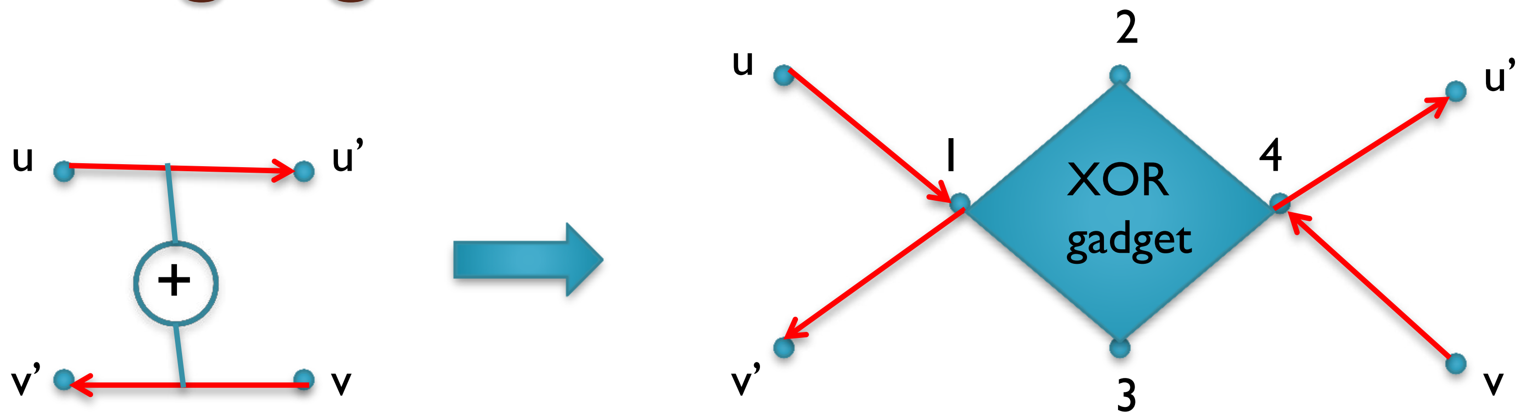
- Claim 1b. Cycle covers, which map to a <u>specific</u> choice of the "touching patterns" of the XOR gadgets s.t. the "touching pattern" of <u>at least one</u> of the XOR gates is of type b, <u>**do not**</u> contribute to the final sum.
- Proof. Follows from Feature 2. *(Homework)*

XOR gadget



- Claim 1c. Cycle covers, which map to a <u>specific</u> choice of the "touching patterns" of the XOR gadgets s.t. the "touching pattern" of <u>every</u> XOR gate is of type c or d, <u>together</u> contribute 4^{3m} or 0 to the final sum.
- Proof. Follows from Feature 3 & 4, and Observations 2a, 2b & 1. *(Homework)*

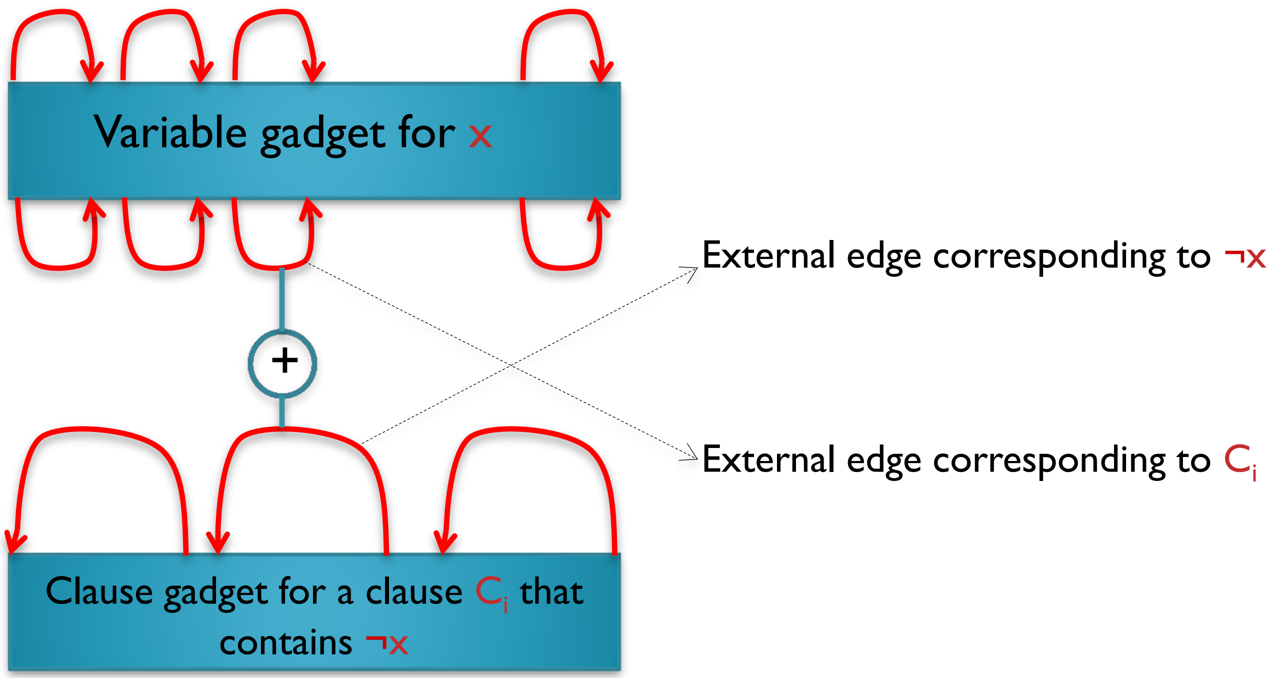
XOR gadget



- Claim 1a, 1b and 1c justify the name of the “XOR” gadget.

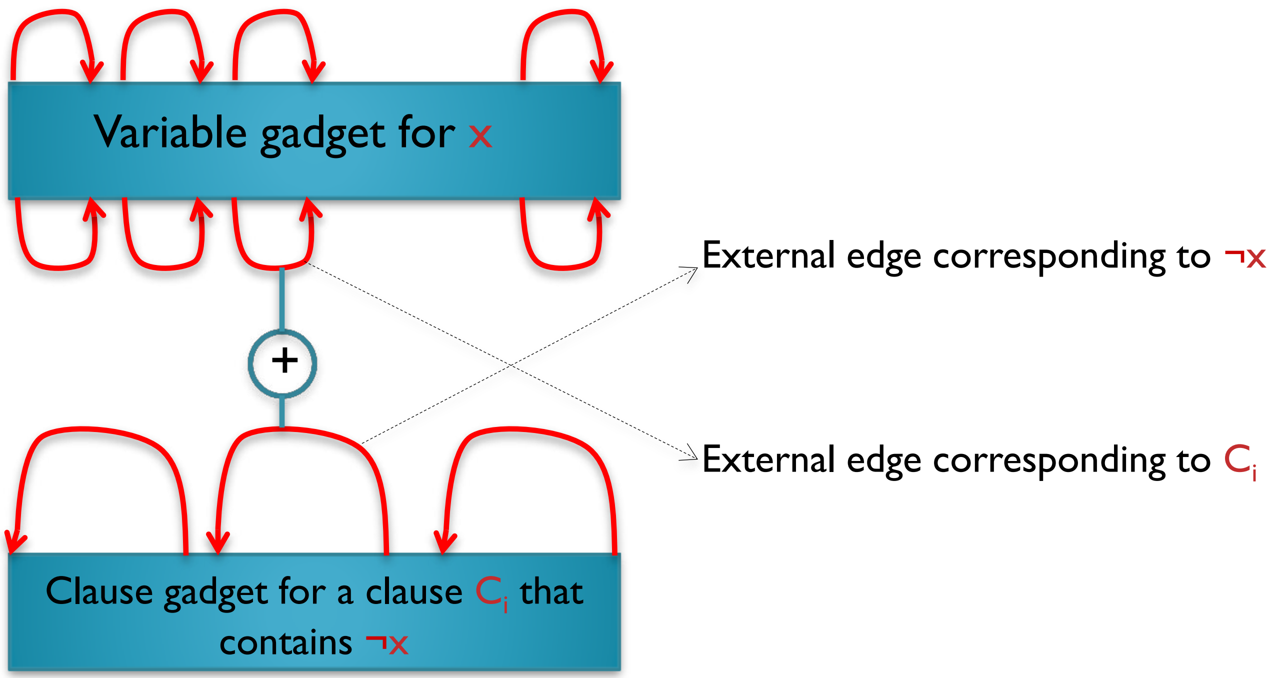
- The XOR gadget ensures that either the “edge” (u,u’) or the “edge” (v,v’) is taken in a <u>potentially</u> contributing choice of the “touching patterns” of the XOR gadgets.

Construction of H



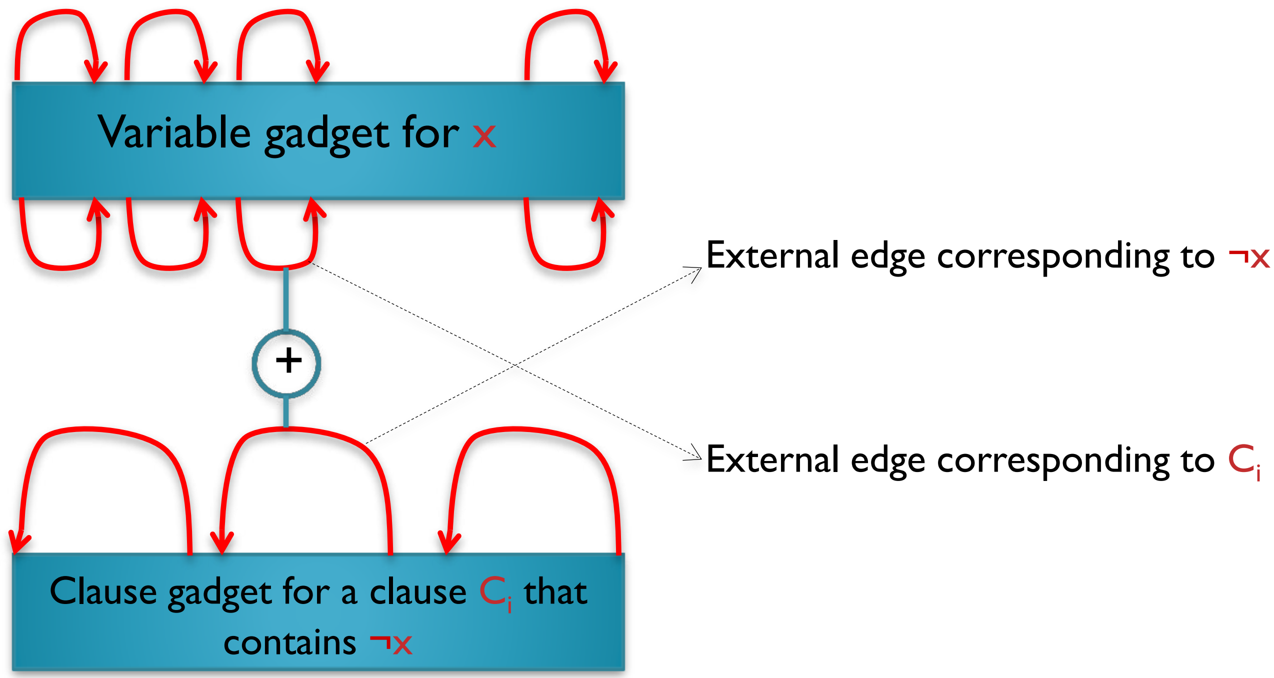
- Observation 3. Every <u>potentially</u> contributing choice of the "touching patterns" of the XOR gadgets can be mapped to a <u>unique</u> choice of the cycle covers of the variable gadgets. *(Homework)*

Construction of H



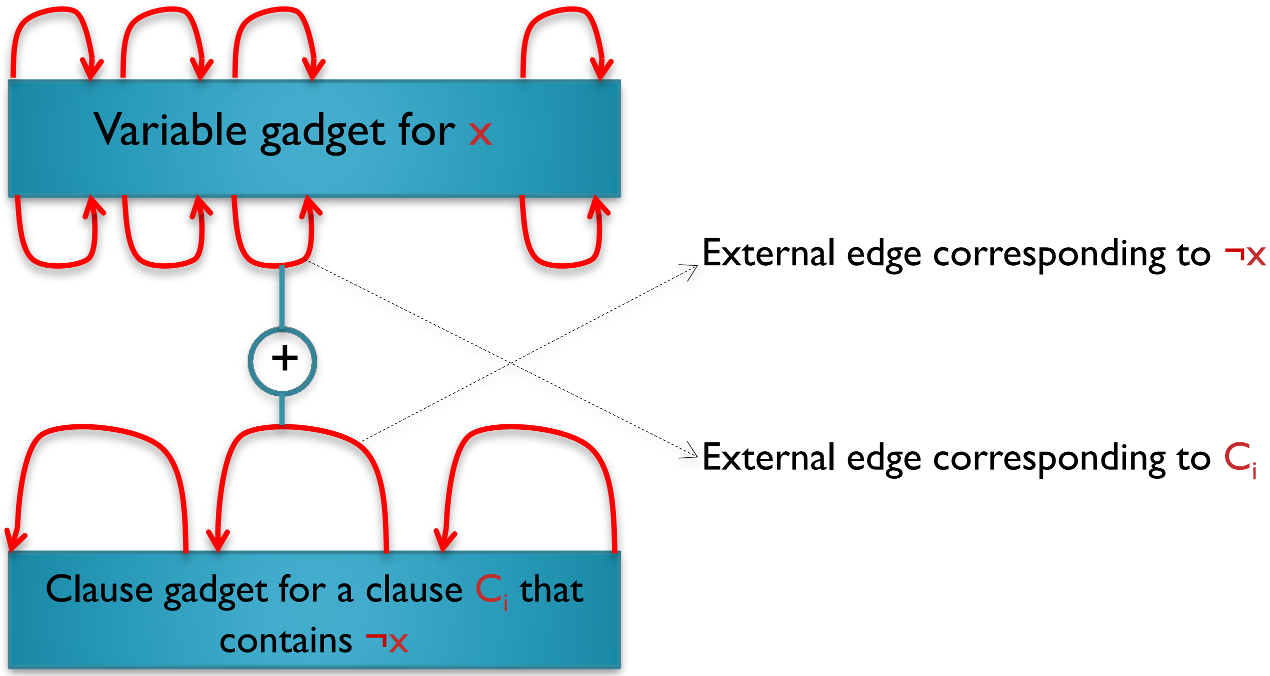
- Recall (from Observation 1) that a variable gadget has exactly 2 cycle covers corresponding to 0/1 assignment to the variable.

Construction of H



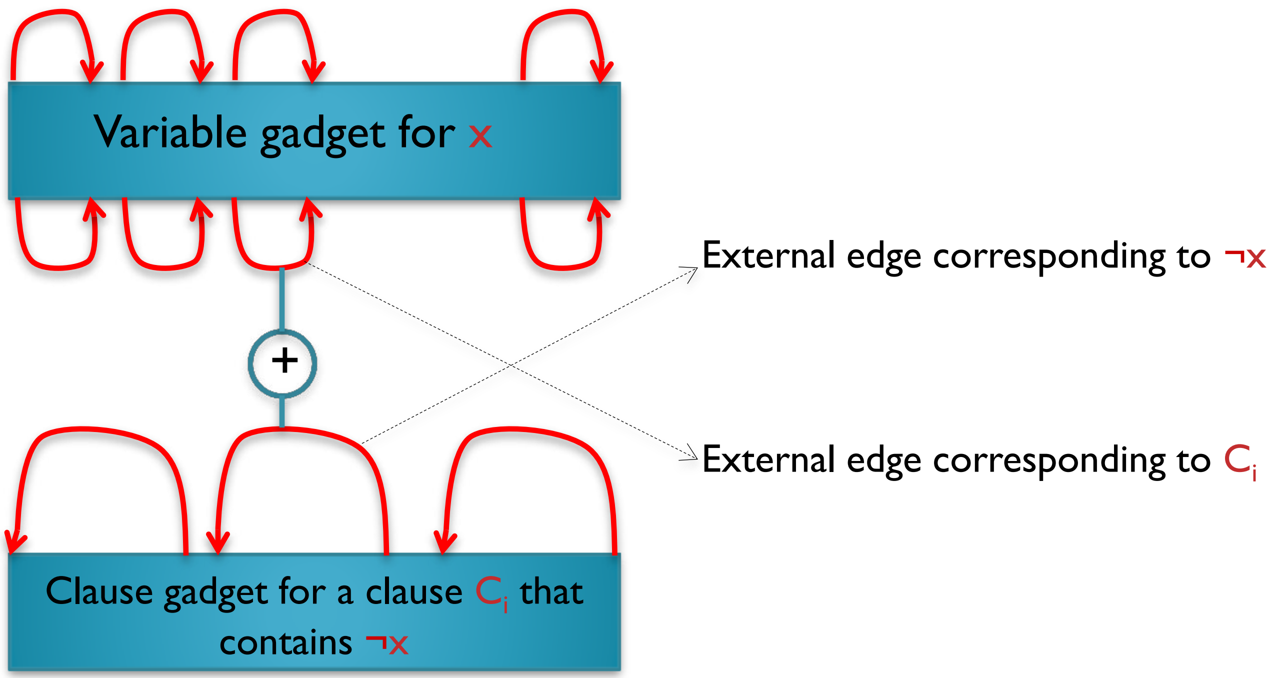
- Observation 3. (put differently) Every <u>potentially</u> contributing choice of the "touching patterns" of the XOR gadgets can be mapped to a <u>unique</u> 0/1 assignment to the variables.

Construction of H



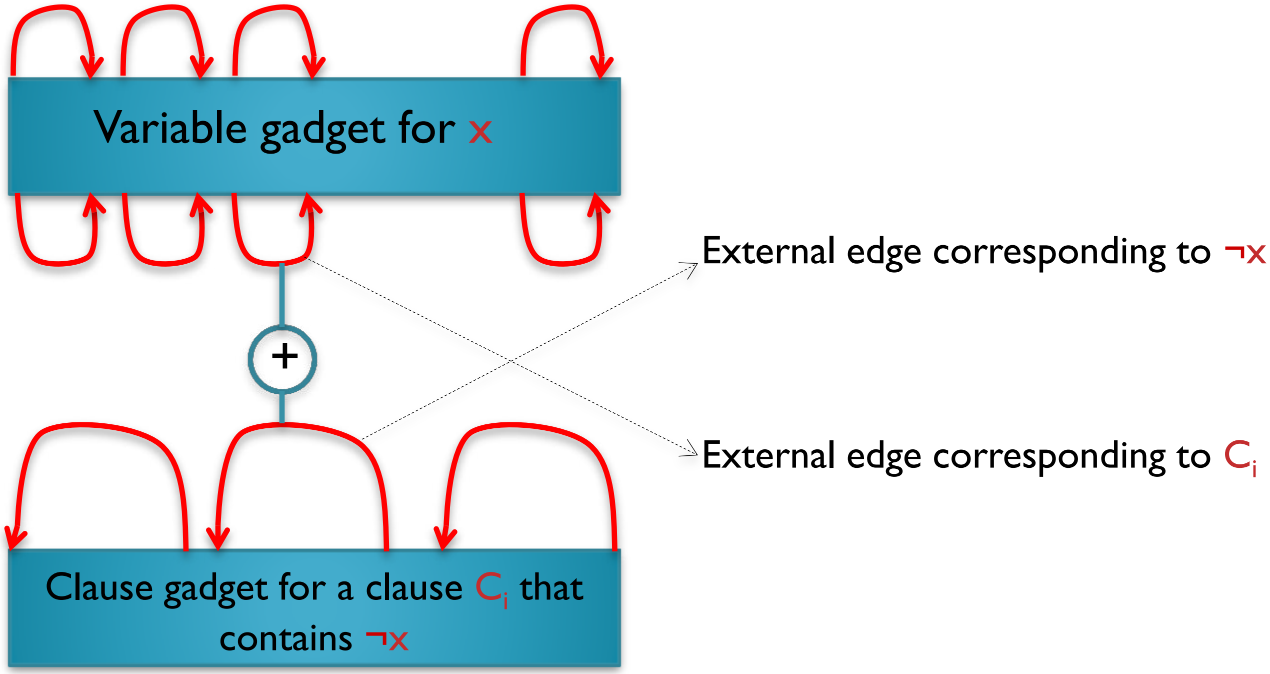
- Which of these 0/1 assignments to the variables correspond to <u>actually</u> contributing choice of the "touching patterns" of the XOR gadgets?

Construction of H



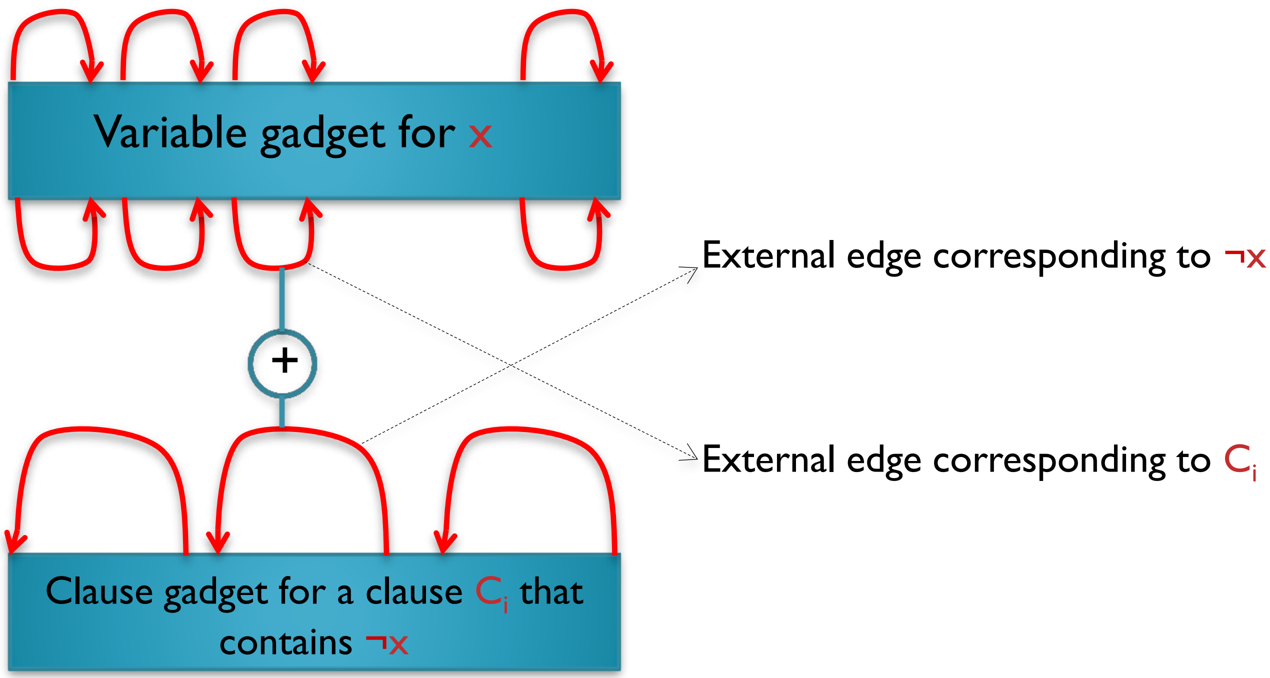
- Which of these 0/1 assignments to the variables correspond to <u>actually</u> contributing choice of the "touching patterns" of the XOR gadgets?
- Answer. Exactly the satisfying assignments of ϕ. *(Why?)*

Construction of H



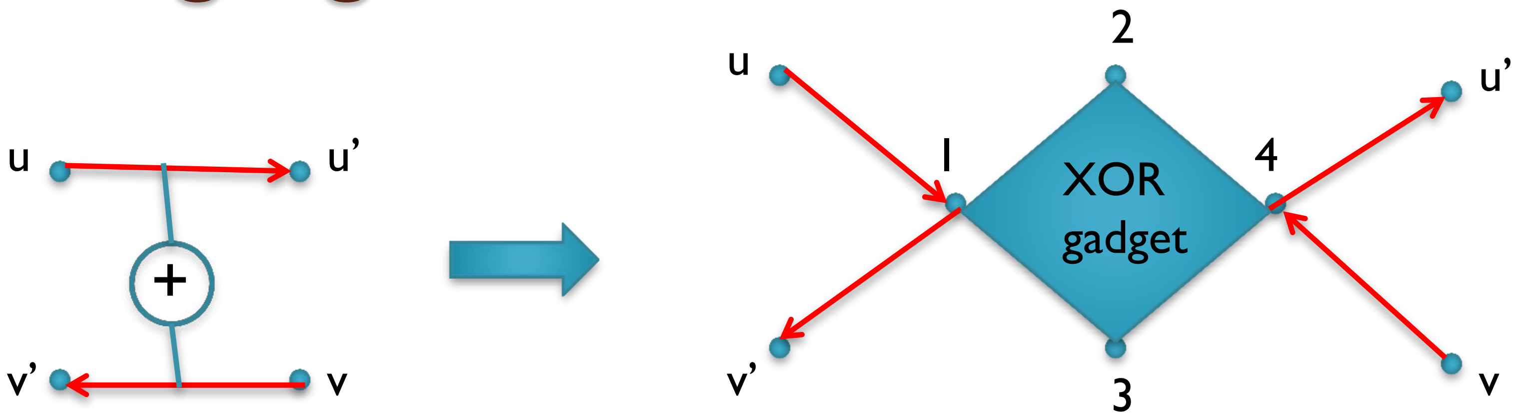
- Hence, the sum of the weighted cycle covers of H is 4^{3m}. #ϕ.
- In other words, $Perm(A_H) = 4^{3m}$. #ϕ. This concludes Step I of the proof of the Theorem.

Construction of H



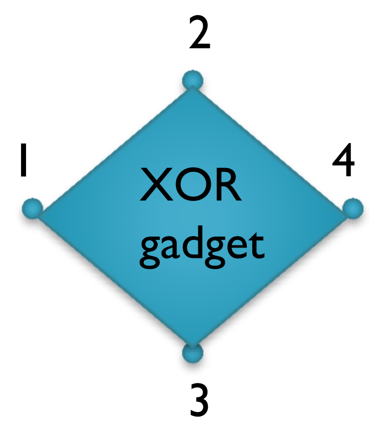
- Hence, the sum of the weighted cycle covers of H is 4^{3m}. #ϕ.
- In other words, $Perm(A_H) = 4^{3m}$. #ϕ. This concludes Step 1 of the proof of the Theorem. (Wait! How do we construct the XOR gadget?)

XOR gadget



- Let $X = (x_{i,j})_{4x4}$ be the adj. matrix of the XOR gadget.
- We need to pick $x_{i,j}$ in a way such that Feature 1, 2, 3 and 4 are satisfied.

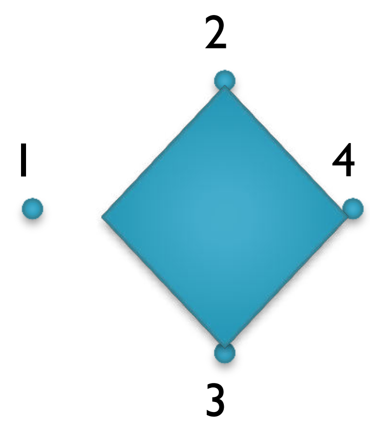
XOR gadget



- Let $X = (x_{i,j})_{4x4}$ be the adj. matrix of the XOR gadget.
- We need to pick $x_{i,j}$ in a way such that Feature 1, 2, 3 and 4 are satisfied.

- Condition 1. Feature 1 implies $\mathrm{Perm}(X) = 0$.

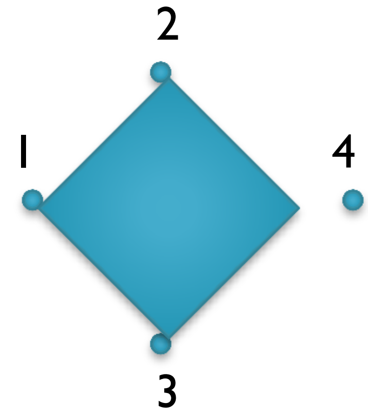
XOR gadget



- Let $X = (x_{i,j})_{4x4}$ be the adj. matrix of the XOR gadget.
- We need to pick $x_{i,j}$ in a way such that Feature 1, 2, 3 and 4 are satisfied.

- Condition 2. Feature 2 implies $\mathrm{Perm}(X_{\{2,3,4\}}) = 0$, where $X_{\{2,3,4\}}$ is the submatrix of X restricted to the rows and columns that are indexed by 2, 3 and 4.

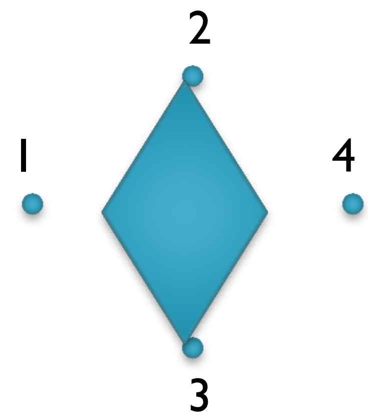
XOR gadget



- Let $X = (x_{i,j})_{4x4}$ be the adj. matrix of the XOR gadget.
- We need to pick $x_{i,j}$ in a way such that Feature 1, 2, 3 and 4 are satisfied.

- Condition 2. Feature 2 implies $\text{Perm}(X_{\{1,2,3\}}) = 0$, where $X_{\{1,2,3\}}$ is the submatrix of X restricted to the rows and columns that are indexed by 1, 2 and 3.

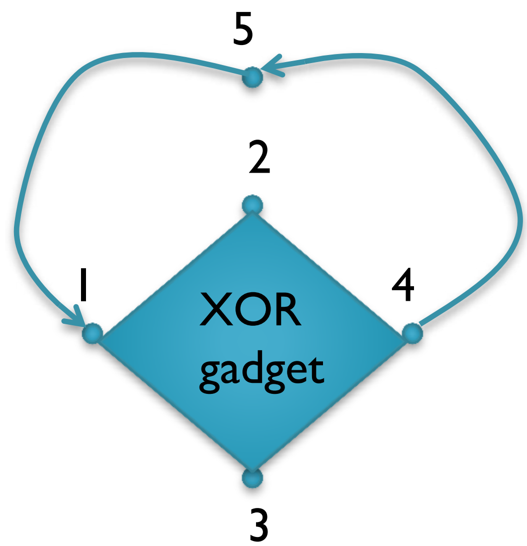
XOR gadget



- Let $X = (x_{i,j})_{4x4}$ be the adj. matrix of the XOR gadget.
- We need to pick $x_{i,j}$ in a way such that Feature 1, 2, 3 and 4 are satisfied.

- Condition 2. Feature 2 implies $\mathrm{Perm}(X_{\{2,3\}}) = 0$, where $X_{\{2,3\}}$ is the submatrix of X restricted to the rows and columns that are indexed by 2 and 3.

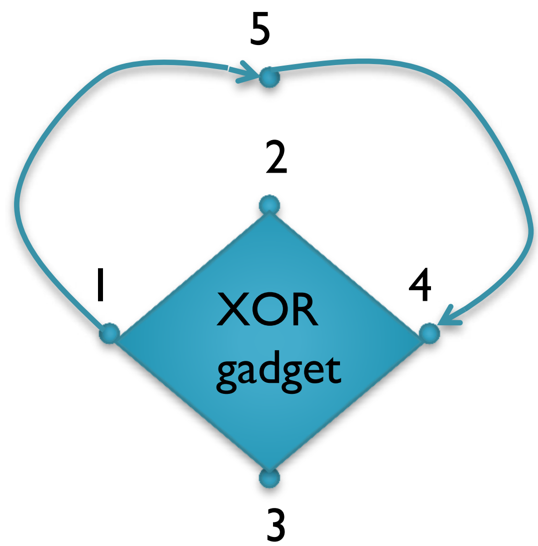
XOR gadget



- Let $X = (x_{i,j})_{4x4}$ be the adj. matrix of the XOR gadget.
- We need to pick $x_{i,j}$ in a way such that Feature 1, 2, 3 and 4 are satisfied.

- Condition 3. Feature 3 implies $Perm(Y) = 4$, where Y is the adjacency matrix of the above 5-vertex graph.

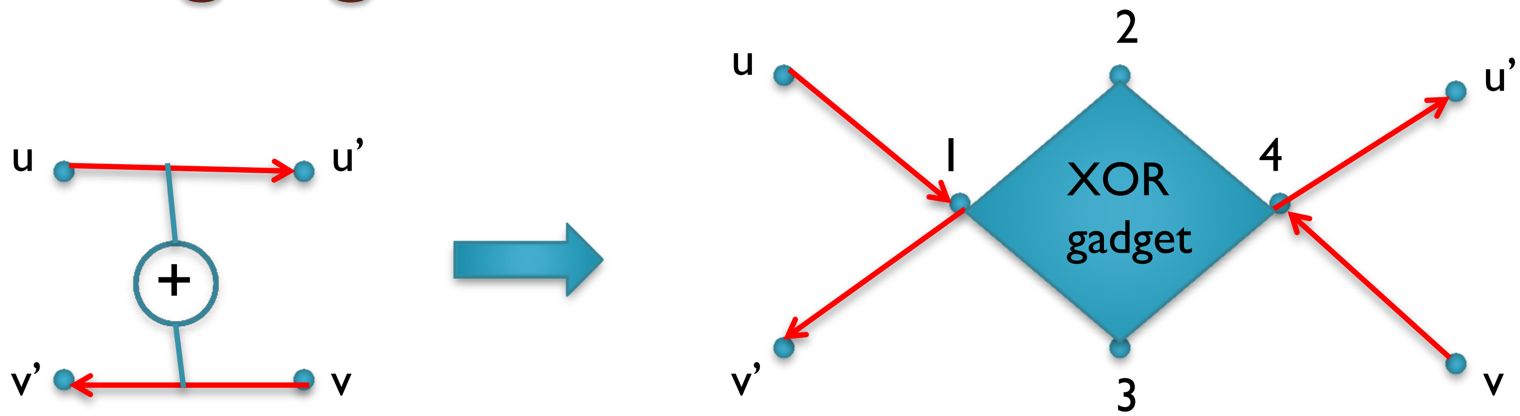
XOR gadget



- Let $X = (x_{i,j})_{4x4}$ be the adj. matrix of the XOR gadget.
- We need to pick $x_{i,j}$ in a way such that Feature 1, 2, 3 and 4 are satisfied.

- Condition 4. Feature 4 implies $\mathrm{Perm}(Z) = 4$, where Z is the adjacency matrix of the above 5-vertex graph.

XOR gadget



- Set X as follows to satisfy Condition 1, 2, 3 and 4.

- X =

0	1	-1	-1
1	-1	1	1
0	1	1	2
0	1	3	0

0/1-Permanent is #P-complete

- Theorem. *(Valiant 1979)* 0/1-Perm is #P-complete.
- Proof. Let ϕ be a 3CNF that has n variables and m clauses. Assume that every clause has <u>exactly</u> 3 literals.

- Step 1: From ϕ we'll form a graph $H = H_\phi$ that has edge weights in $\{-1, 0, 1, 2, 3\}$ such that

$$\mathrm{Perm}(A_H) = \sum_{\substack{C:\ C \text{ is cycle} \\ \text{cover of } H}} \mathrm{wt}(C) = 4^{3m} . \#\phi .$$

- We have completed Step 1.

0/1-Permanent is #P-complete

- Theorem. *(Valiant 1979)* 0/1-Perm is #P-complete.
- Proof. Let ϕ be a 3CNF that has n variables and m clauses. Assume that every clause has <u>exactly</u> 3 literals.

- Step 2: We'll process H further to get a new graph $G = G_\phi$ with edge weights in $\{0,1\}$ such that $\#\phi$ can be efficiently computed from $\mathrm{Perm}(A_G)$.

- Let us now focus on Step 2.

Step 2

- Covert H to H' that has edge weights from {-1, 0, 1} by first introducing parallel edges, and then, introducing extra vertices to get rid of the parallel edges. Let p = poly(n,m) be the number of vertices of H'.

Step 2

- Covert H to H' that has edge weights from $\{-1, 0, 1\}$ by first introducing parallel edges, and then, introducing extra vertices to get rid of the parallel edges. Let $p = poly(n,m)$ be the number of vertices of H'.

- Observe that $Perm(A_H) = Perm(A_{H'}) \in [0, p!]$. Set $r = p^2$ and note that $2^r + 1 > p!$.

Step 2

- Covert H to H' that has edge weights from $\{-1, 0, 1\}$ by first introducing parallel edges, and then, introducing extra vertices to get rid of the parallel edges. Let $p = \text{poly}(n,m)$ be the number of vertices of H'.

- Observe that $\text{Perm}(A_H) = \text{Perm}(A_{H'}) \in [0, p!]$. Set $r = p^2$ and note that $2^r + 1 > p!$.
- Hence, $\text{Perm}(A_{H'})$ is the same as $\text{Perm}(A_{H'}) \bmod (2^r+1)$.

Step 2

- Covert H to H' that has edge weights from $\{-1, 0, 1\}$ by first introducing parallel edges, and then, introducing extra vertices to get rid of the parallel edges. Let $p = \mathrm{poly}(n,m)$ be the number of vertices of H'.

- Observe that $\mathrm{Perm}(A_H) = \mathrm{Perm}(A_{H'}) \in [0, p!]$. Set $r = p^2$ and note that $2^r + 1 > p!$.
- Hence, $\mathrm{Perm}(A_{H'})$ is the same as $\mathrm{Perm}(A_{H'}) \bmod (2^r+1)$.
- As $-1 = 2^r \bmod (2^r + 1)$, we can replace the weights of the edges in H' that are labelled by -1 with 2^r to form a graph G' and compute $\mathrm{Perm}(A_{G'}) \bmod (2^r+1)$.

Step 2

- Covert H to H' that has edge weights from $\{-1, 0, 1\}$ by first introducing parallel edges, and then, introducing extra vertices to get rid of the parallel edges. Let $p = poly(n,m)$ be the number of vertices of H'.

- Finally, transform G' to G with $0/1$ edge weights by
  - replacing every edge with weight 2^r by a sequence of r edges each having weight 2, and then
  - replacing every edge with weight 2 by a pair of parallel weight 1 edges, and then
  - removing parallel edges like before.

Step 2

- Covert H to H' that has edge weights from $\{-1, 0, 1\}$ by first introducing parallel edges, and then, introducing extra vertices to get rid of the parallel edges. Let $p = poly(n,m)$ be the number of vertices of H'.

- In the end, we get $Perm(A_G) = 4^m. \#\phi \mod (2^r + 1)$, where G is a graph with 0/1 edge weights.

- It is because of the modulus "mod $(2^r + 1)$" that an FPRAS for 0/1-Perm doesn't imply an FPRAS for #3SAT.