To prove MAX-CUT is NP-complete, we need to prove it’s both NP and NP-hard. It’s NP because it’s easy to verify in polynomial time whether a potential solution is correct because we just need to see whether the cut goes through edges that have one endpoint in and the other in .

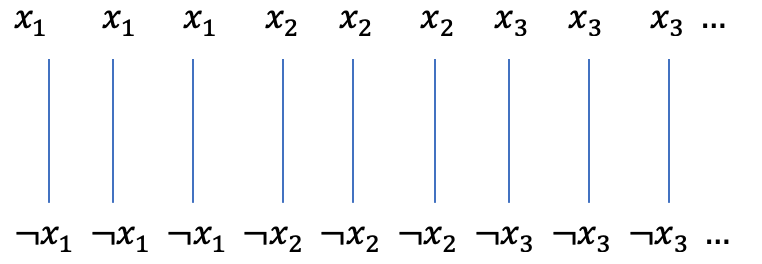
To prove it’s NP-hard, we will reduce from a -SAT problem as per the hint. This reduction is polynomial time because we’re constructing nodes and edges.

Claim: There is a cut of size on the transformed graph if and only if is -satisfiable.

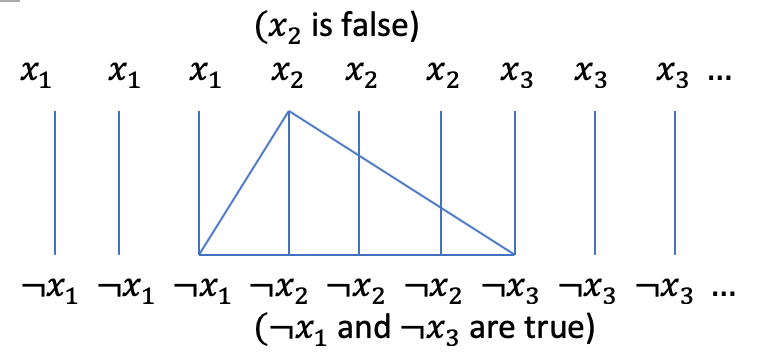
Proof:

Claim: If ­ is -satisfiable, then there exists a cut of size on the transformed graph.

Proof: Visualize all the nodes to be lined up horizontally on the upper side of the page and all the nodes to be lined up horizontally on the lower side of the page as such:



If a literal is true, put it in ; otherwise put it in . There are thus edges between all the ’s and ’s. The satisfying -assignment makes it so that one or more but less than three literals in each clause is true. The resulting triangle thus has nodes on both sides, as such:



A cut for a satisfying -assignment thus has the size . The is because of the edges between ’s and ’s, and the is because each satisfied clause’s triangle would get cut twice.

Claim: If there exists a cut of size , then is -satisfiable.

Proof: Consider all the triangles getting cut. Try two things. The first thing to try is

Claim: If is -unsatisfiable, then there does not exist a cut of size or greater.

Proof: There would be at least one clause triangle with all three vertices entirely in or entirely in . (because they all have the same truth value). The biggest cut for such a graph is at most .