02-613 Week 14

Algorithms and Advanced Data Structures

Aidan Jan

December 5, 2025

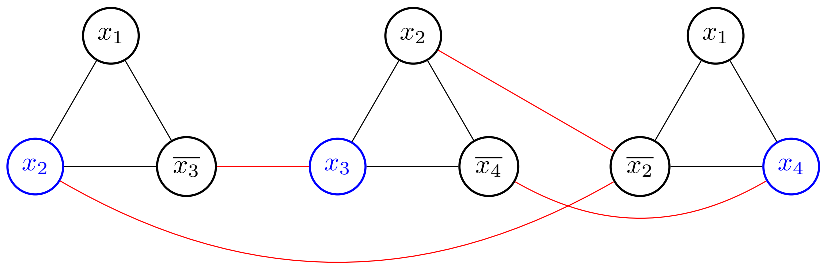
Using 3SAT to Prove NP-Hard

3SAT has been proven to be NP-Hard (in fact, it was the first) in 1971 in the Cook-Levin paper. Now we can use it in other proofs.

Proving Independent Set is NP-Hard

We can reduce 3SAT to Independent Set to show that Independent Set is NP-Hard. To do this, draw a graph. Each clause of three terms OR'ed together is converted to 3 nodes on the graph connected with edges. Furthermore, each node that reference the same term (e.g., if t_1 is in two different clauses), the two nodes are also connected with an edge. Now, this is an independent set problem, where whichever nodes are set to true are possible values that would satisfy 3SAT.

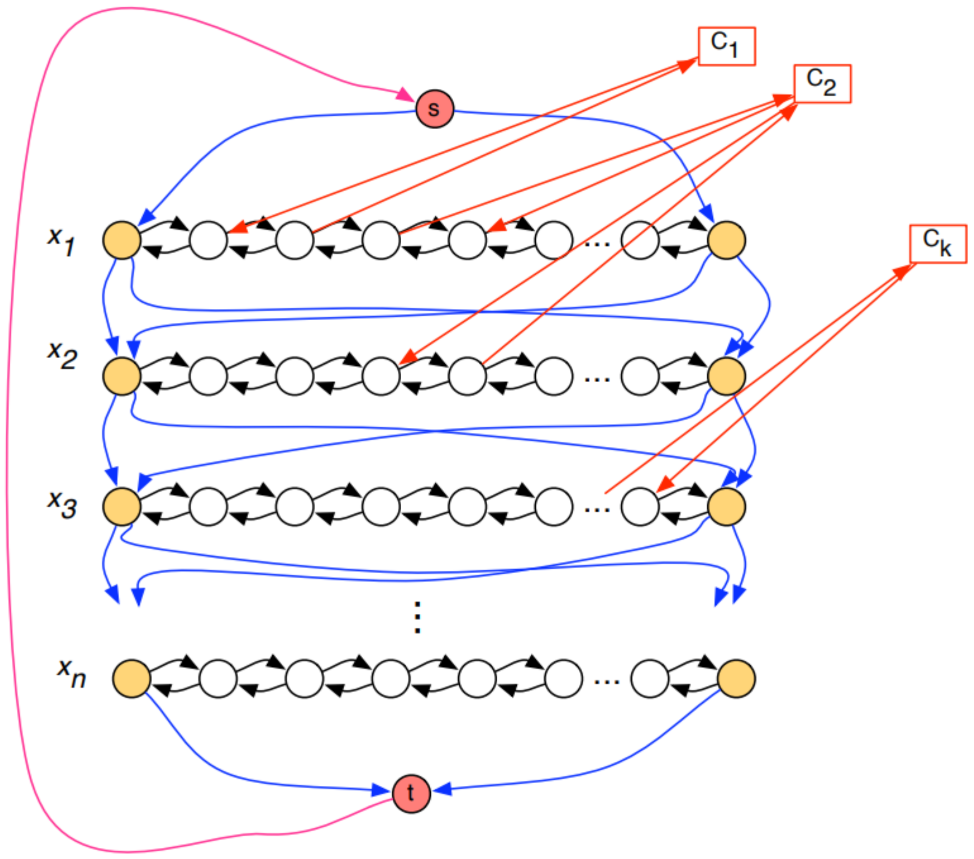
For example, consider $(x_1 \vee x_2 \vee \overline{x_3}) \wedge (x_2 \vee x_3 \vee \overline{x_4}) \wedge (x_1 \vee \overline{x_2} \vee x_4)$



- Red edges are edges connecting the same variables.
- Blue nodes are those we can select as part of our independent set.
- The set of nodes we select as our independent set are the variables we set to true in 3SAT in order to satisfy it.
- The 3SAT problem is satisfied if and only if our independent set is equal to the number of clauses. (In this case, 3 clauses means we select three nodes.) Our selected nodes represent the variable in each clause that is true.

Proving Hamiltonian Cycle is NP-Hard

Given a directed graph G, is there a $\underline{\text{cycle}}$ that visits each vertex exactly once? The theorem says Hamiltonian Cycle is NP-complete, but we can prove that. To do this, we build the following construction.



Each variable in 3-SAT gets a row of nodes, doubly linked. The length of the row of nodes is $2 \times n + 2$, where n is the number of clauses in 3-SAT. Each start point and end point of each row is connected to both the start point and end point of the next row. At this point, the Hamiltonian cycle must traverse across each row, in either the forwards or backwards direction.

Now, we link each clause into the graph. For each clause, c_i, if it uses a given variable x_j, then add an edge going from $x_{j,2i}$ to c_i, and an edge from c_i to $x_{j,2i+1}$. If the clause uses the complement of a variable, (x_j is negated), then draw the edges in the reverse direction, from $x_{j,2i+1}$ to c_i and from c_i to $x_{j,i}$. Each clause gets two columns on the graph to link to its variables.

A Hamiltonian Cycle passing through this graph would be required to pass through each clause at least once. If a Hamiltonian Cycle exists in this graph, it gives the variable assignments necessary to satisfy every clause. Moving from left to right in a variable's row corresponds to it being assigned to True, and moving backwards corresponds to it being assigned to False.

Proving Hamiltonian Path is NP-Hard

We can prove Hamiltonian Path from Hamiltonian Cycle very easily. Given a graph G for Hamiltonian Cycle, if there exists a cycle, then every node must have one incoming edge and one outgoing edge. Based on this fact, we simply pick any node, i, and separate it into two nodes. One node, i_i will contain all the incoming edges of i, and the other, i_o will contain all the outgoing edges of i. When given to the Hamiltonian Path solver, it will be forced to start a path at i_o and end at i_i. If the path does not exist, then the cycle must not exist either. To retrieve the solution for the Cycle, connect the start and end points of the graph.

Traveling Salesman Problem

Given n cities, distances $d(i, j)$ between them, does there exist a path of length less than k that visits every city exactly once, and returns home? Importantly, the distance function need not satisfy the triangle inequality, and $d(i, j)$ need not equal $d(j, i)$.

To prove that, we can do a reduction from Hamiltonian Cycle. However, we need a lemma first. Let $G = (V, E)$ and $D = (V, d(i, j))$, where all distances are either 1 or 2. 1 if an edge exists between the two cities, 2 otherwise.

Lemma: G has a Hamiltonian Cycle if and only if D has a tour of length $\leq$ n.

Proof. If G has a Hamiltonian Cycle, that is an ordering of the cities giving a tour in D of length n. Therefore, the forward implication is proven.
Suppose D has a tour of length $\leq n$. Then, the tour cannot use a distance-2 connection because with n steps, that would be too long. Therefore, it visits cities only on distance-1 edges which in G are nodes with edges.

Thus, Hamiltonian cycle reduces to TSP, and TSP is NP-hard.

"Solving" NP-Hard Problems

Once a problem is determined to be NP-Hard, chances are that we will not be finding a solution to it that runs in polynomial time. In this case, we have two options:

- **Heuristics:** (with something like A*). Not guaranteed to be optimal, but usually polynomial.
- **Approximation:** See below.

Approximation Algorithms

Let $A_p(I)$ be the approximation solution vector.

- A = algorithm
- p = problem
- I = instance

Let $\text{OPT}_p(I)$ be the theoretical optimal solution vector (which may normally take exponential time to find).

Now, an approximation algorithm satisfies the following equation:

$$A_p(I) \leq \alpha(|I|)\ \text{OPT}_p(I)$$

α is a bound, typically around 2 or $\log(n)$. This basically means, our approximate solution will be at worst some factor times the optimal solution. For example, for TSP, generating an optimal solution is NP-Hard, but we can get a solution that is at worst twice as bad as the optimal. We can also provide a lower bound function, $B_p(I) <\ \text{OPT}_p(I)$, if $A(I) \leq \alpha B(I) \Rightarrow A(I) \leq \alpha\ \text{OPT}(I)$

Example: Approximation of TSP

Given n cities in $\mathbb{R}^2$ and Euclidian metric, find the shortest path to visit all the cities once.

One valid approximation algorithm would be

- Construct MST

- Use preorder traversal, and leave out repeated visits

How bad is this approximation? Let A be the edges visited on the tour by our algorithm, counting multiplicity. Now, let A^* be the edges visited in an optimal tour. Let the function $\text{cost}(S)$ be the cost of a tour.

Theorem: $\text{cost}(A) \leq 2\ \text{cost}(A^*)$ Let $\text{cost}(T) \leq \text{cost}(A^*)$. A walk w that traces the MST has length $2\text{cost}(T)$. This is because every edge in the MST is visited twice. Therefore,

$$\Rightarrow \text{cost}(w) = 2\ \text{cost}(T) \leq 2\ \text{cost}(A^*)$$

w is not a tour since it repeats cities. However, by cutting out repetitions, we are guaranteed to reduce the walk distance because of the triangle inequality. We then get

$$\Rightarrow \text{cost}(A) \leq 2\ \text{cost}(T) \leq 2\ \text{cost}(A^*)$$

Therefore, this MST approximation of the Traveling Salesman Problem is at worst twice as bad as the optimal. Notice that we never actually found the cost of the optimal algorithm