CSCI3390-Lecture 17: A sampler of **NP**-complete problems

1 List of Problems

We now know that if L is any problem in **NP**, that

$$L \leq_P \text{SAT},$$

and thus SAT is **NP**-hard. Since SAT is also in **NP** we find that SAT is **NP**-complete. Further,

$$\text{SAT} \leq_P \text{3-SAT},$$

so 3-SAT is also **NP**-complete.

It turns out that most (though not all) of the problems in **NP** that we saw, unless they are obviously in **P**, are **NP**-complete. These notes list a few of these results, along with similar results for problems we haven't discussed. We will then give a detailed proof that the Hamiltonian path problem is **NP**-complete.

1.1 Cliques in Graphs

A k-clique in an undirected graph is a set of k vertices such that each is adjacent to the other $k-1$.

The following problem is $\mathbf{NP}$-complete:

> Input: An undirected graph G and an integer $k \leq n$, where n is the number of vertices of G.
>
> Output: Yes if G contains a clique of size k, no otherwise.

An *independent set* in a graph is a set of vertices such that no vertex in the set is adjacent to any other vertex in the set. (Think of trying to find a large group of people so that no one in the group knows anyone else in the group.)

The following problem is $\mathbf{NP}$-complete:

Input: An undirected graph G and an integer $k \leq n$, where n is the number of vertices of G.

Output: Yes if G contains an independent set of size k, no otherwise.

Comment: These are the *same* problem: A clique in a graph G is an independent set in the dual graph G' and conversely. The dual graph is obtained by adjoining two vertices if and only if they are not adjoined in the original graph. Observe that we can construct the dual graph of G in time polynomial in the size of G, so each of these problems is polynomial-time reducible to the other; thus if one of them is **NP**-complete, the other is as well.

There is still a third problem in this vein, the *Vertex Cover* problem.

Input: An undirected graph G and an integer $k \leq n$, where n is the number of vertices of G.

Output: Yes if there is a set S of vertices such that every *edge* of G has at least one of its two vertices in S

(Think of trying to place mailboxes on corners of city streets so that everyone is no more than one block away from a mailbox. The vertices of the graph here are the intersection, and the edges the streets joining the intersections. So this is a problem of finding a vertex cover.)

Comment: This too is equivalent to the preceding problems, since S is a vertex cover of G if and only if $V(G) - S$ is an independent set.

1.2 Graph coloring

A k-coloring of a graph G is a function $f : V \to \{1, \ldots, k\}$, where V is the set of vertices of G, such that if $v, v' \in V$ are adjacent, then $f(v) \neq f(v')$. (Put otherwise, adjacent vertices must be colored different colors.)

The following problem is **NP**-complete.

Input: An undirected graph G.

Output: Yes if G has a 3-coloring, no otherwise.

Comment. Based on this, it is easy to prove(homework problem) that 4-colorability (and 5-colorability, etc) are all **NP**-complete.

1.3 Subset sum

The following problem is **NP**-complete:

> Input: A set $S = \{k_1, \ldots, k_m\}$ of positive integers, and a target value M.
>
> Output: Yes if S has a subset T such that $\sum_{s \in T} s = M$.

For example, if $S = \{14, 32, 53, 75, 96\}$ and $M = 181$, then the output is yes, because $181 = 32 + 53 + 96$.
Comment: The integers are encoded in binary or decimal (or some other radix). There is an 'efficient' dynamic programming algorithm for solving this problem that works as follows. We construct a rectangular table $T(i, j)$, for $1 \leq i \leq m$ and $0 \leq j \leq M$, such that $T(i, j) = 1$ if and only if there is a subset of $k_1, \ldots, k_i$ that sums to M. There is a simple recurrence that allows us to fill in the table row by row:

$$T(0, j) = 1.$$

$$T(i, j) = 1 \text{ if } j = k_i.$$

$$T(i + 1, j) = 1 \text{ if} T(i, j) = 1.$$

$$T(i, j + k_i) = 1 \text{ if} T(i, j) = 1.$$

In all other cases $T(i, j) = 0$. The output is yes if $T(m, M) = 1$. There is no contradiction here: note that the size mM of this table, and hence the number of steps of the dynamic programming algorithm, is exponential in the size $(m + 1) \cdot \log M$ of the input. It makes a difference here that we code the numbers in binary or decimal rather than in unary.

1.4 Hamiltonian Path

The following problem is **NP**-complete.

> Input: An undirected graph G and vertices s, t of G.
>
> Output: Yes if G contains an Hamiltonian path from s to t, no otherwise.

Comment. Still **NP**-complete if the graph is directed, or if we ask for a Hamiltonian circuit rather than a path from s to t.

Compare this to the famous optimization problem *the Traveling Salesman Problem*:

Input: An undirected graph with positive integer weights on the edges.

Output: A circuit in the graph that visits every vertex and has minimum total weight for all such paths.

Suppose the weights are distances, or train ticket prices. The salesman wants to visit every city on his route and return home, and do so covering the smallest distance possible, or paying the least amount of money.

Unless **P=NP**, TSP has no polynomial-time algorithm. If it did, we could apply it to a graph in which all edge weights are 1. The graph has a Hamiltonian circuit if and only if the optimal path returned by TSP has total weight equal to the number of vertices. Thus an efficient solution to TSP would imply a polynomial-time algorithm for the Hamiltonian circuit problem, and thus that **P=NP**.

2 Some Proofs

It is easy to prove that these problems are in **NP**, the art is in proving they are **NP**-hard

2.1 Directed Hamiltonian Path

We begin with the problem of determining if there is a Hamiltonian path from a given source vertex s to a destination vertex t in a *directed* graph. We will do this by showing a polynomial-time reduction

$$SAT \leq_{\mathbf{P}} \text{Directed Hamiltonian Path.}$$

So, given a formula ϕ in CNF, we will show how to construct a directed graph G with two distinguished vertices s, t such that ϕ is satisfiable if and only if G contains a Hamiltonian path from s to t. In other words, we have to translate a logic problem into a graph problem. The reduction proceeds by the introduction of a number of 'gadgets' (that is the term that is actually used).

Figure 1 shows variable gadgets. There are three of these, the diamond-shaped figures stacked one on top of the other. The horizontal paths within each variable gadget are divided into three stages. In general, the number of variable gadgets is equal to the number of distinct variables in the formula, and the number of stages in the horizontal path paths within each gadget is equal to the number of clauses in the formula.

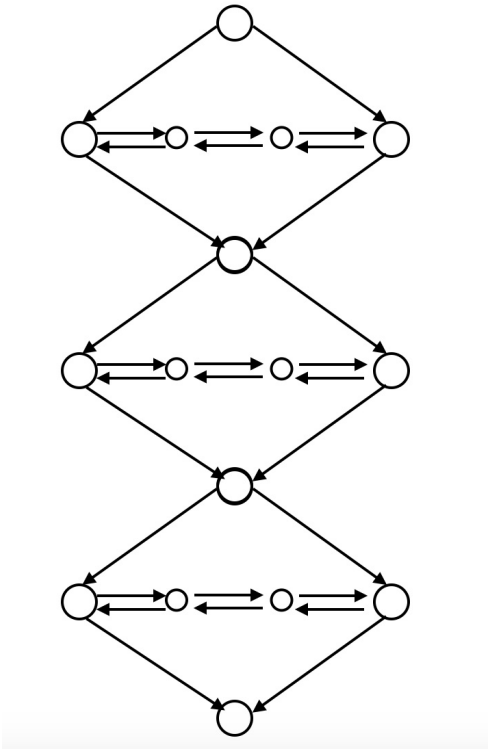

Figure 1: An assemblage of three variable gadgets. There are 8 different Hamiltonian paths from top to bottom, corresponding to the 8 different choices of a sequence of starting directions from the top of each gadget.

Pause a moment and consider the problem of finding a directed Hamiltonian path from the top vertex to the bottom vertex of Figure 1. There are clearly many such paths. Within a single gadget I can begin by choosing to take the left-branching or right-branching edge. Whichever edge I choose, I am obliged to take the horizontal path across the gadget to the opposite side, and then continue down to the top of the next variable gadget. In general, if there are k variables, then there are 2^k different Hamiltonian paths from the top to the bottom, one for each sequence of choices for the starting direction in the k gadgets.

Now let's add a new vertex (Figure 2) and connect it to the variable gadgets. This is a clause gadget. The clause gadget represents a kind of detour away from the rightmost stage of the horizontal paths in each gadget. A Hamiltonian path in the enlarged graph must include this new vertex exactly once, so we have to take exactly one of the available detours. Observe that if we take the top detour, we must be traversing the horizontal path from right to left, so we must have started in the right-branching direction in the top gadget. If we decide instead to take either the middle or the bottom detour, then we must start in the left-branching direction in the corresponding variable gadget.

Figure 3 shows the effect of adding a new clause gadget, associated with the middle stage of the horizontal paths. We have provided three stages in our diagram, so we could add a third clause gadget, but the diagram is getting rather cluttered, so we will leave it as it is. Figure 4 shows a Hamiltonian path in the complete diagram. Observe that we have chosen to visit the blue clause gadget from the second variable gadget, and the red clause gadget from the third variable gadget, but there are a lot of different choices that work here. (It is just a coincidence that the number of variable gadgets and clause gadgets are equal in this example.)

Figure 5 shows a CNF formula, in this case consisting of a pair of clauses, associated with this diagram. (In carrying out the reduction, of course we start from the formula and construct the gadgets–the construction was presented this way to help you understand how the diagrams worked). We arbitrarily associate the directions left and right with true and false, so that choice of a starting direction in each variable gadget corresponds to an assignment of a truth value to the corresponding variable. We connect the clause gadgets to the variable gadgets according to the following rule: if the variable appears negated in the clause, we introduce a right-to-left detour from the corresponding variable gadget, otherwise we introduce a left-to-right detour. Thus the only way to take the detour is to have assigned the right truth value to the variable.

In this manner, every assignment that satisfies all the clauses gives a Hamilto-

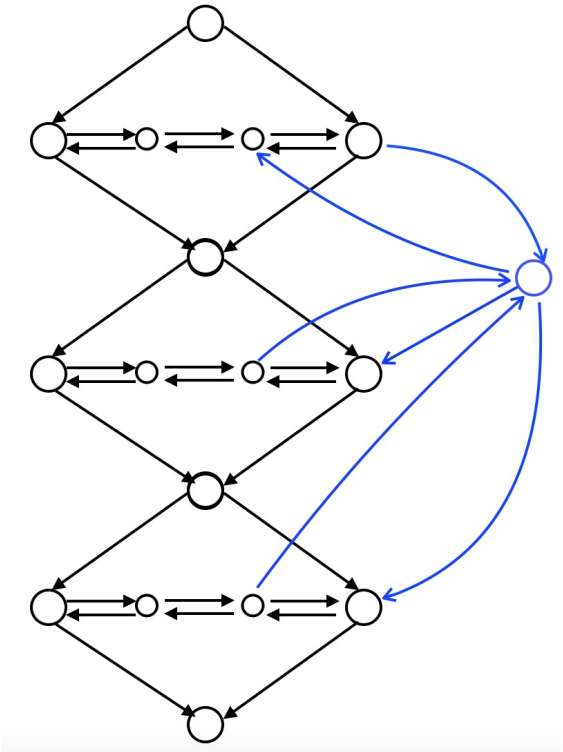

Figure 2: Addition of a clause gadget. A Hamiltonian path must take a detour through the new vertex, and we provide such a detour from each variable gadget. But whether you can take that detour depends on the starting direction in that variable gadget.

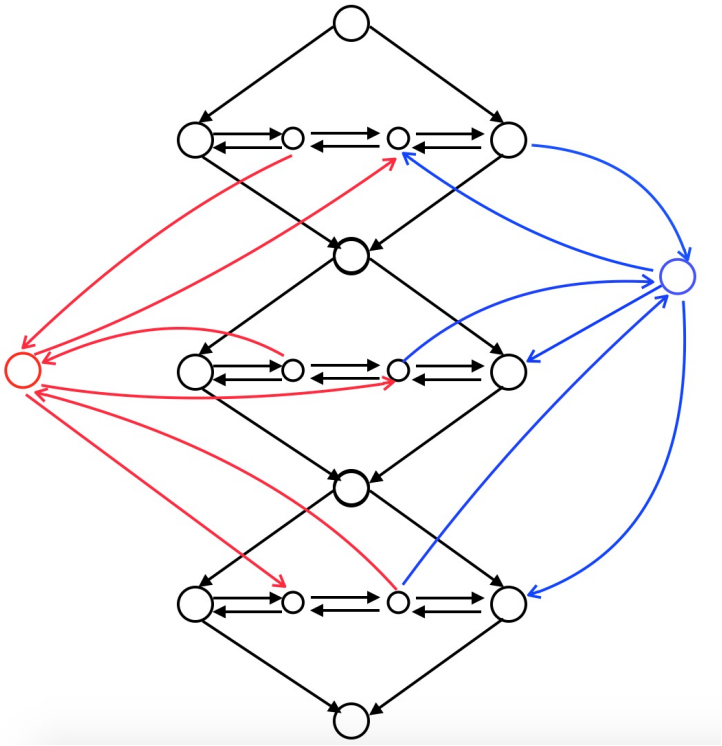

Figure 3: Addition of a second clause gadget, associated with the middle stage of each variable gadget. There is room in the diagram for still another clause gadget, but we don't want to clutter things up too much.

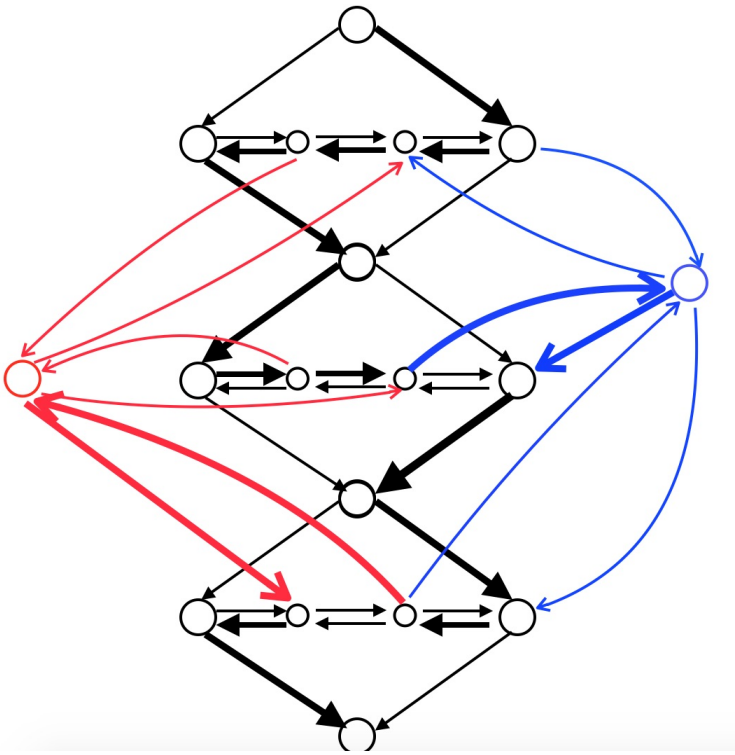

Figure 4: A Hamiltonian path—can you find all the others?

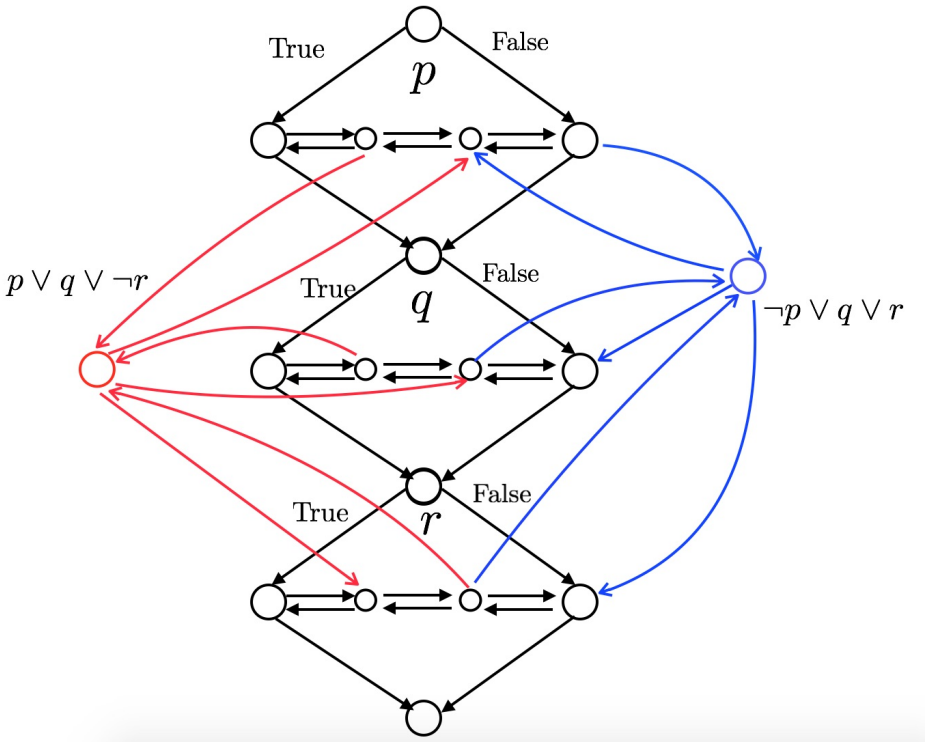

Figure 5: The corresponding CNF formula. We show only two clauses, but the diagram could accommodate a third. Each variable gadget is associated to a variable, fixed directions are chosen corresponding to True and False, and the clause gadgets are connected to the variable gadgets according to whether the variable in question appears positively or negatively in the clause. Satisfying assignments for the formula thus correspond to Hamiltonian paths.

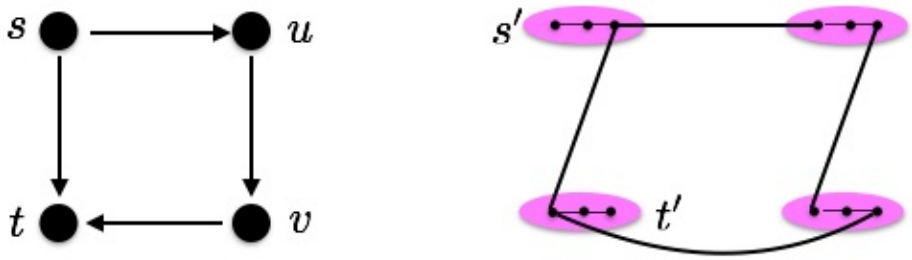

Figure 6: Reduction of the Hamiltonian path problem for directed graphs to the version for undirected graphs. Each vertex is replaced by a cluster of three connected vertices.

nian path from the top to the bottom, and conversely, so a Hamiltonian path exists if and only if the original formula is satisfiable.

Observe that the total number of vertices in the graph is $O(km)$, where k is the number of variables and m the number of clauses. The total number of edges is likewise $O(km)$. So the size of the graph is bounded above by a polynomial in the size of the formula. Again, the algorithm for constructing the graph consists basically of the instructions for writing down each vertex and edge (There is no involved computation to do beyond counting the number of variables and clauses and translating each literal into an edge.) So this is a polynomial-time reduction.

2.2 Undirected Hamiltonian Path

We show that the problem of determining the existence of a Hamiltonian Path in an undirected graph is also **NP**-complete, by a reduction from the directed case. Given a directed graph G and vertices s, t, we will show how to construct in polynomial time an undirected Graph G' with vertices s' and t', such that G has a directed Hamiltonian path from s to t if and only if G' has an undirected Hamiltonian path from s' to t'.

Figure 6 illustrates the construction: Each vertex of the directed graph is mapped to a cluster of three vertices in the undirected graph, and a directed edge from v to w is mapped to an edge from the right-hand vertex of the cluster corresponding to v to the left-hand vertex of the cluster corresponding to w.

Easily, a directed Hamiltonian path from s to t in a digraph G gives rise to a Hamiltonian path in G' from the left-hand vertex s' of the cluster corresponding

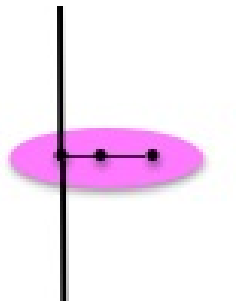

Figure 7: If s' is the left-hand vertex of a cluster and t' the right-hand vertex of another cluster, then the two vertical edges shown in the diagram above cannot be part of a Hamiltonian path from s' to t', because we would never be able to visit the middle vertex of the cluster without retracing our steps. Thus a Hamiltonian path from s' to t' can never follow edges in the 'wrong' direction.

to s to the right-hand vertex t' of the cluster corresponding to t. It is a little more delicate to argue that any Hamiltonian path in G' from s' to t' must give rise to a directed Hamiltonian path in G from s to t. We must argue that the Hamiltonian path in G' can never follow an edge between clusters in the 'wrong' direction, from the left-hand vertex of cluster C_1 to the right-hand vertex of a cluster C_2. If we did so, our path could never reach the middle vertex of C_1: It would have to get there from the right vertex of C_1, and then have no exit without revisiting a vertex already visited.