Hamiltonian Cycle and 3-Dimensional Matching

1 Hamiltonian Cycle

1.1 Hamiltonian Cycle

HAMILTONIAN CYCLE (HC): Given a graph $\mathcal{G}$, is there a tour that visits each vertex exactly once? (Not to be confused with Eulerian cycles, which contain each edge exactly once).

Theorem 1. *HC is **NP**-complete.*

Proof. A two-part proof: (1) HC $\in$ **NP**, (2) HC is **NP**-hard.

(1). HC $\in$ **NP**

Certificate: A proposed tour.

Certifier: A polytime algorithm which checks existence of all edges and that the tour contains every vertex exactly once.

(2). HC is **NP**-hard

We reduce from a known **NP**-hard problem (3SAT) to HC.

Theorem 2. *3SAT $\leq p$ HC*

Proof. The input to the reduction is a formula F; the output is a digraph $\mathcal{G}$. We must encode the truth assignment of variables into the choice of the order for visiting each node in $\mathcal{G}$.

For each x_i, we build a "variable gadget", a long bidirected path. We add a start node s, and connect it to the left and right endpoints of the x_1 gadget. Then, we connect the left and right endpoints of each x_i gadget to the left and right endpoints of the x_{i+1} gadget, and the left and right endpoints of the x_m gadget back up to s. Thus, a true or false decision is encoded with a left or right traversal respectively.

For each clause $C_j \in F$, we have a new node (a "clause gadget") u_j. If $C_j = l_{j,1} \vee l_{j,2} \vee l_{j,3}$, then we have one incoming and one outgoing edge per $l_{j,i}$. If $l_{j,i} = x_k$, then the edges are $(v_{k,2j}, u_j), (u_j, v_{k,2j+1})$. Otherwise if $l_{j,i} = \bar{x_k}$, the edges are $(u_j, v_{k,2j}), (v_{k,2j+1}, u_j)$. Since there are m clauses, we can tell that each variable gadget must have length $2m + 2$ (2 nodes for each clause gadget, and a terminal node on either end).

This algorithm runs in polytime $\Theta(mn)$ and satisfies:

(1). If F is satisfiable, then $\mathcal{G}$ has an HC.

For each i, if x_i is true, we traverse the corresponding variable gadget left to right; otherwise, we traverse it right to left. For each C_j, let $l_{j,i}$ be a true literal. Then the HC will take the corresponding detour to the clause gadget, thus visiting every vertex exactly once.

(2). If $\mathcal{G}$ has an HC, then F is satisfiable.

Observation: If the HC follows an edge from $(v_{i,2j}, u_j)$, it must immediately continue with $(u_j, v_{i,2j+1})$ instead of moving to a different variable gadget; otherwise, $v_{i,2j+1}$ cannot ever be visited again (and thus the HC does not exist). So, the HC must traverse each variable gadget completely, taking detours to each u_j. Then, we define $x_i :=$ true if its corresponding variable gadget was traversed left to right, and false otherwise. Because we were able to take detours to all u_j, all C_j are satisfied, so F must be satisfied. □

Since we proved that HC is both in **NP** and **NP**-hard, HC must be **NP**-complete. □

1.2 Hamiltonian Path

HAMILTONIAN PATH (HP): Given a graph $\mathcal{G}$, is there a path that visits each vertex exactly once? (Doesn't have to return back to the start node).

Theorem 3. *HP is **NP**-complete.*

Proof. A two-part proof: (1) HP $\in$ **NP**, (2) HP is **NP**-hard.

(1). HP $\in$ **NP**

Certificate: A proposed path.

Certifier: A polytime algorithm which checks existence of all edges and that the path contains every vertex exactly once.

(2). HP is **NP**-hard

We reduce from a known **NP**-hard problem (HC) to HP.

Theorem 4. *HC $\leq p$ HP*

Proof. The input to the reduction is a digraph $\mathcal{G}$; the output is a digraph $\mathcal{G}'$. We select an arbitary v from $\mathcal{G}$ and break it into v^- and v^+, where v^- has all of v's incoming edges and v^+ has all of v's outgoing edges. Then, $\mathcal{G}$ has an HC if and only if $\mathcal{G}'$ has an HP from $v^+ \to v^-$. □

Since we proved that HP is both in **NP** and **NP**-hard, HP must be **NP**-complete. □

2 Traveling Salesman

TRAVELING SALESMAN PROBLEM (TSP): Given a graph $\mathcal{G}$ with edge costs and an integer k, is there a tour of total cost $\leq k$?

Theorem 5. *TSP is **NP**-complete.*

Proof. A two-part proof: (1) TSP $\in$ **NP**, (2) TSP is **NP**-hard.

(1). TSP $\in$ **NP**

Showing this is simple except in the case of Euclidean TSP (ETSP, where we are given (x, y) coordinates instead of edge weights). This is because those edge weights may be irrational, and most algorithms with irrational numbers are in **PSPACE**. It is an open question whether ETSP $\in$ **NP**.

(2). TSP is **NP**-hard

We reduce from a known **NP**-hard problem (HC) to TSP.

Theorem 6. *HC $\leq p$ TSP*

Proof. The input to the reduction is a digraph $\mathcal{G}$ without costs, and the output is a digraph $\mathcal{G}'$ with edge costs and an integer k. We set $k = n$ (to guarantee a tour) and set distances $d_{i,j}$ such that:

$$d_{i,j} = \begin{cases} 1 & (i,j) \in \mathcal{G} \\ \infty & (i,j) \notin \mathcal{G} \end{cases}$$

Then, $\mathcal{G}$ has an HC if and only if it has a tour with cost $\leq k = n$. □

Since we proved that TSP is both in **NP** and **NP**-hard, TSP must be **NP**-complete. □

3 3-Dimensional Matching

3.1 3-Dimensional Matching

3-DIMENSIONAL MATCHING (3DM): Given sets X, Y, Z such that $|X| = |Y| = |Z| = n$, and triples $T_1, T_2, \ldots, T_m \subseteq X \times Y \times Z$, is there a subset $S \subseteq \{1, 2, \ldots, m\}$ such that each X_i, Y_j, Z_k is in exactly one T_l for all $l \in S$? This problem is important because it is simultaneously a cover and packing problem (the triples have to cover the sets, and each set element can only be in a triple once); thus, it is very useful to reduce from.

Theorem 7. *3DM is **NP**-complete.*

Proof. A two-part proof: (1) 3DM $\in$ **NP**, (2) 3DM is **NP**-hard.

(1). 3DM $\in$ **NP**

Certificate: A set S that supposedly matches the triples with the sets.

Certifier: A polytime algorithm which ensures coverage and absence of overlap.

(2). 3DM is **NP**-hard

We reduce from a known **NP**-hard problem (3SAT) to 3DM.

Theorem 8. *3SAT $\leq p$ 3DM*

Proof. We will encode $x_i =$ true or false in making lots of $z_{i,j}$ set elements for x_i available; our goal is to ensure that either elements for x_i or for $\bar{x_i}$ are available, but not both. The variable gadget on the next page ensures that either all even $z_{i,j}$ are free, or all odd $z_{i,j}$ are free. For the remaining $x_{i,j}$ and $y_{i,j}$, it is only possible to cover them by selecting all even or odd $z_{i,j}$.

We then design a clause gadget for C_j consisting of two new nodes a_j and b_j. We know that the variable gadgets have $2m$ $z_{i,j}$ nodes each (corresponding to one x_i and one $\bar{x_i}$ per clause). Thus, we triple $(a_j, b_j, z_{k,2j-1})$ if x_k, and $(a_j, b_j, z_{k,2j})$ if $\bar{x_k}$.

Finally, we produce $2m(n-1)$ cleanup elements ($m(n-1)$ each in X and Y). Each pair of cleanup elements can be formed into a triple with each leftover $z_{i,j}$.

This algorithm runs in polytime $\Theta(mn)$ and satisfies:

(1). If F is satisfiable, then a 3DM exists.

We choose triples to make $z_{i,j}$ available for even j if $\bar{x_i}$, and odd j if x_i. For each clause, we activate the corresponding triple for the chosen literal, then triple the remaining $z_{i,j}$ with the cleanup elements.

(2). If a 3DM exists, then F is satisfiable.

For each i, either all odd $z_{i,j}$ or all even $z_{i,j}$ are matched with $x_{i,j}$ and $y_{i,j}$. If j even, then set $\bar{x_k}$; if j odd, then set x_k. For each clause C_j, a_j and b_j are mathced with some $z_{k,2j}$ or $z_{k,2j-1}$. Because that $z_{k,i}$ was free, the corresponding literal is true, so C_j is satisfied. Since all m clauses are satisfied, F must be satisfied. □

Since we proved that 3DM is both in **NP** and **NP**-hard, 3DM must be **NP**-complete. □

3.2 Exact Cover by 3 Sets

EXACT COVER BY 3 SETS (X3C): Given a set X with $|X| = 3n$ and triples $T_1, T_2, \ldots, T_m \subseteq X$, is there a set $S \subseteq \{1, 2, \ldots, m\}$ with each $x \in X$ contained in exactly one T_j for $j \in S$? This problem is 3DM generalized, with one large set instead of three equally-sized smaller sets.

Theorem 9. *X3C is **NP**-complete.*

Proof. A two-part proof: (1) X3C $\in$ **NP**, (2) X3C is **NP**-hard.

(1). X3C $\in$ **NP**

Certificate: A set S that supposedly matches the triples with the sets.

Certifier: A polytime algorithm which ensures coverage and absence of overlap.

(2). X3C is **NP**-hard

We reduce from a known **NP**-hard problem (3DM) to X3C.

Theorem 10. *3DM $\leq p$ X3C*

Proof. We simply define X in X3C as $X \cup Y \cup Z$ from 3DM, and the rest of the proof follows trivially. $\square$

Since we proved that X3C is both in **NP** and **NP**-hard, X3C must be **NP**-complete. $\square$