

# DIRECTED HAMILTONIAN PATH

HAMPATH: Given a *directed* graph  $G$  and two nodes  $s, t \in V$ , is there a path in  $G$  from  $s$  to  $t$  that goes through every node exactly once.

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

**Proof:**

HAMPATH  $\in$  NP.

We reduce 3-SAT to HAMPATH.

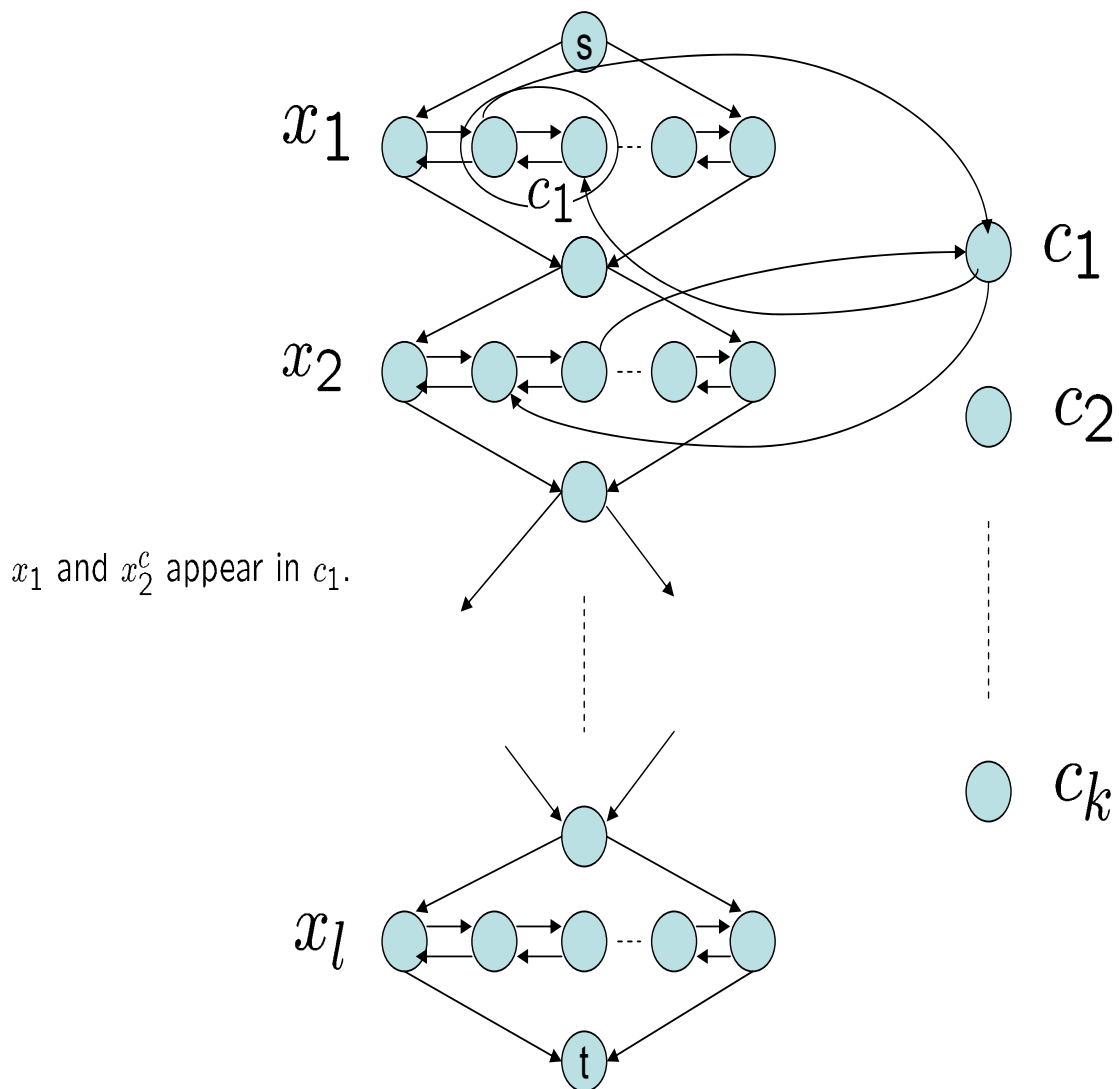
Let  $\phi$  be a 3-SAT instance containing  $k$  clauses:

$$\phi = (a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \wedge \dots \wedge (a_k \vee b_k \vee c_k)$$

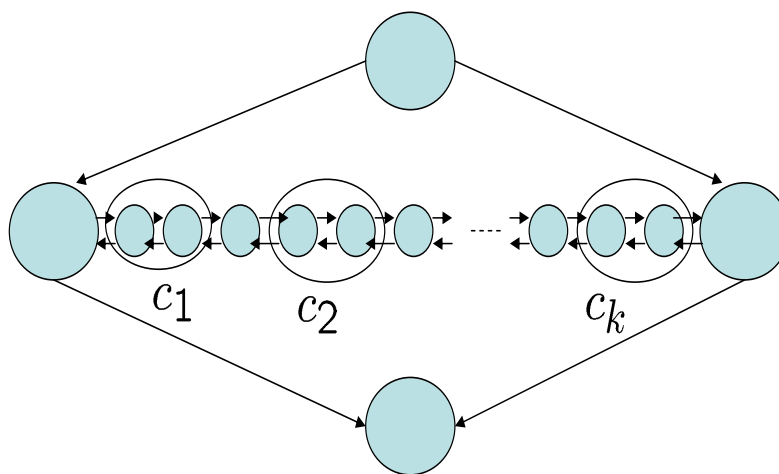
where each  $a, b, c$  is a literal  $x_i$  or  $x_i^c$ ,  $1 \leq i \leq l$ .

We convert  $\phi$  into a graph instance  $G$  as follows.

# The high-level structure of $G$



# The Diamond Gadget



We show that  $\phi$  is satisfiable iff there exists a Hamiltonian path from  $s$  to  $t$ .

Suppose that  $\phi$  is satisfiable.

The path begins at  $s$ , goes through each diamond (from top to bottom), in turn and ends up at  $t$ .

The horizontal nodes are included in the path as follows: if  $x_i$  is true then we zig-zag (left-in and right-out from top) and if it is false we zag-zig (right-in and left-out from top) through the diamond.

The clause nodes are included by adding detours at the horizontal nodes. We use one of the true literals of a clause to detour.

To show the other way, let  $G$  have a Hamiltonian path from  $s$  to  $t$ .

If the Hamiltonian path goes through the diamonds in order from top one to the bottom one (except for the detours to the clause nodes), the satisfying assignment is: zig-zag means true and zag-zig means false. By observing the diamond at which the detour is taken, we can determine which of the literals in the corresponding clause is true.

We now argue that any Hamiltonian path in  $G$  should be as above.

The only other way is for a path to enter a clause from one diamond and return via another. By our construction of  $G$  this is not possible.

# UNDIRECTED HAMILTONIAN PATH

UHAMPATH: Given a *undirected* graph  $G$  and two nodes  $s, t \in V$ , is there a path in  $G$  from  $s$  to  $t$  that goes through every node exactly once.

**Theorem 2.** *UHAMPATH is NP-complete.*

**Proof:** UHAMPATH  $\in$  NP.

We reduce from HAMPATH.

Let  $\langle G, s, t \rangle$  be an instance of HAMPATH. Construct an undirected graph  $G'$  as follows:

Each node of  $G$ , except for  $s$  and  $t$  is replaced by a triple of nodes  $u^{in}, u^{mid}, u^{out}$  in  $G'$ .  $s$  and  $t$  are replaced by  $s^{out}$  and  $t^{in}$  in  $G'$ .

Edges in  $G'$  connect  $u^{mid}$  with  $u^{in}$  and  $u^{out}$ . Also, there is an edge between  $u^{out}$  and  $v^{in}$  in  $G'$  if an edge goes from  $u$  to  $v$  in  $G$ .

$G$  has a Hamiltonian path from  $s$  to  $t$  iff  $G'$  has a Hamiltonian path from  $s^{out}$  to  $t^{in}$ .

# UNDIRECTED HAMILTONIAN CYCLE

UHAMCYCLE: Given a *undirected* graph  $G$ , is there a cycle in  $G$  that goes through every node exactly once.

**Theorem 3.** *UHAMCYCLE is NP-Complete.*

**Proof:**

Reduce from UHAMPATH.

# Traveling Salesman Problem

TSP: Given a complete graph  $G$  and an integer cost function  $c$  on the edges and an integer  $k$ , is there a tour (that goes through each vertex exactly once) with cost at most  $k$ .

**Theorem 4.** *TSP is NP-Complete.*

**Proof:**

Reduce from UHAMCYCLE.

Let  $G = (V, E)$  be an instance of UHAMCYCLE. We construct an instance  $G'$  of TSP as follows.

$G'$  has the same set of vertices as  $G$ . The cost function is defined as:

if  $(i, j) \in E$ , then  $c(i, j) = 1$  else  $c(i, j) = 2$ .

$G'$  has a tour of cost at most  $|V|$  iff  $G$  has an Hamiltonian cycle.

**Corollary 1.** *Metric-TSP (the costs satisfy triangle inequality) is NP-Complete.*

# SUBSET-SUM

SUBSET-SUM: Given a set  $S$  of natural numbers  $x_1, \dots, x_k$  and a target number  $t$ , is there a subset of  $S$  that adds up to  $t$ .

**Theorem 5.** *SUBSET-SUM is NP-Complete.*

**Proof:** SUBSET-SUM  $\in NP$ . We reduce 3-SAT to SUBSET-SUM.

Let  $\phi$  be a boolean formula with variables  $x_1, \dots, x_l$  and clauses  $c_1 \dots c_k$ .

We construct an instance  $\langle S, t \rangle$  of SUBSET-SUM such that it contains a subset summing to  $t$  iff  $\phi$  is satisfiable.

The numbers in  $S$  and  $t$  are rows in the following table:

$S$  contains a pair of numbers  $y_i$  and  $z_i$  for each variable  $x_i$  in  $\phi$ .

Each number in decimal notation consists of two parts: the left-hand part comprises a 1 followed by  $l - i$  0s; the right-hand part contains one digit for each clause, where the  $j$ th digit of  $y_i$  ( $z_i$ ) is 1 if  $c_j$  contains  $x_i$  ( $x_i^c$ ). All other digits are 0.

Also  $S$  contains one pair of numbers  $g_j$  and  $h_j$  for each  $c_j$ . The two numbers are equal and consist of a 1 followed by  $k - j$  0s.

The target  $t$  consists of  $l$  1s followed by  $k$  3s.

# Example

If  $\phi$  is  $(x_1 \vee x_2^c \vee x_3) \wedge (x_2 \vee x_3 \vee \dots) \wedge \dots \wedge \dots (x_3^c \vee \dots \vee \dots)$ :

	1	2	3	.....	$l$	$c_1$	$c_2$	.....	$c_k$
$y_1$	1	0	0	.....	0	1	0	.....	0
$z_1$	1	0	0		0	0	0		0
$y_2$		1	0		0	0	1		0
$z_2$		1	0		0	1	0		0
⋮				⋮	⋮	⋮		⋮	
$y_l$					1	0	0		0
$z_l$					1	0	0		0
$g_1$						1	0	.....	0
$h_1$						1	0		0
$g_2$							1		0
$h_2$							1		0
⋮									
$g_k$									1
$h_k$									1
$t$	1	1	1	.....	1	3	3	.....	3

We argue that  $\phi$  is satisfiable iff some subset of  $S$  sums to  $t$ .

Suppose  $\phi$  is satisfiable. We select a subset of  $S$  as follows.

We choose  $y_i$  if  $x_i$  is assigned true and  $z_i$  otherwise.

We choose enough of the  $g$  and  $h$  numbers so that the last  $k$  digits add up to 3.

Suppose that a subset of  $S$  sums to  $t$ . We construct a satisfying assignment as follows.

If the subset contains  $y_i$  we assign  $x_i$  true, otherwise we assign it false.

The sum of the final  $k$  columns is always 3.

In column  $c_j$ , at least 1 in this column must come from some  $y_i$  and  $z_i$ .

If it is  $y_i$ , then  $x_i$  appears in  $c_j$  and is assigned true, so  $c_j$  is satisfied.

If it is  $z_i$ , then  $x_i^c$  appears in  $c_j$  and  $x_i$  is assigned false, so  $c_j$  is satisfied.

The reduction can be done in polynomial time: the table has size  $O((l + k)^2)$  and each entry can be calculated in polynomial time.

# KNAPSACK

KNAPSACK: Given a set  $S$  of objects with an integer weight and an integer profit for each object, and two integer values  $W$  and  $K$ , is there a subset of  $S$  whose sum of weights is at most  $W$  and sum of profits is at least  $K$ .

**Theorem 6.** *KNAPSACK is NP-Complete.*

**Proof:** Reduce from SUBSET-SUM.

# Strongly NP-Complete Problems

If a problem remains NP-Complete even if any instance of length  $n$  is restricted to contain integers of size at most  $p(n)$ , a polynomial, then we say that the problem is strongly NP-Complete.

Examples: All problems we have seen except SUBSET-SUM and KNAPSACK.

Problems which are not strongly NP-complete usually admit pseudo-polynomial time algorithms.