26 Hamiltonian Cycle, Intractability, Approximation Algorithms

- NP-completeness of Hamiltonian Cycle
- Coping with intractability
- Approximation algorithms: load balancing

26.1 Hamiltonian Cycle

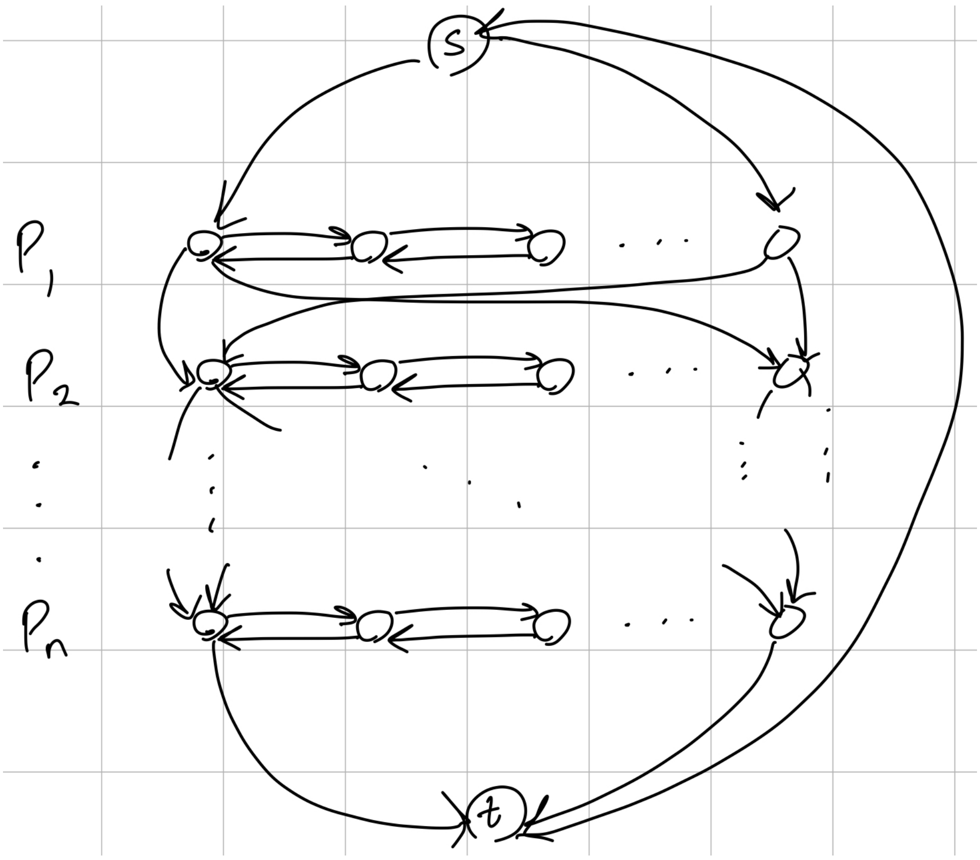
Theorem 26.1
Hamiltonian Cycle is NP-complete.

Proof. HC is in NP.
The proof is a list of all vertices.
The verifier checks that every adjacent pair of vertices is joined by an edge (including the last vertex to the first vertex) and that every vertex appears in the list exactly once.

These checks take polynomial time and accept iff the given list specifies a Hamiltonian Cycle.

It remains to show that HC is NP-hard.
We'll show this by reducing from 3-SAT.

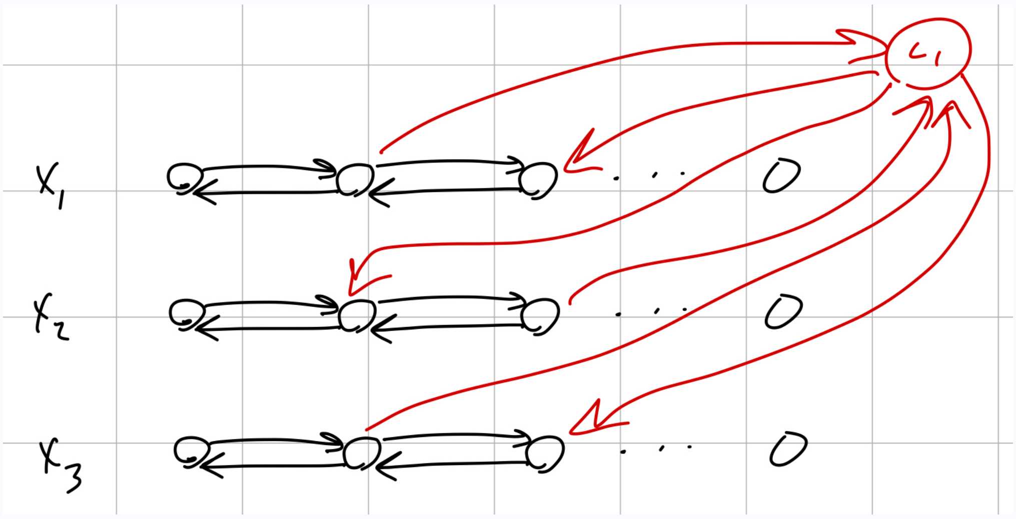
Given a 3-SAT formula with variables $x_1, \cdots, x_n$, construct a graph with long paths $P_1, \cdots, P_n$ and vertices s, t connected as follows:



Here, there are 2^n different Hamiltonian cycles: on each layer, we can choose to start from either the left or the right of the path. But this does not encode any information about the clauses in the 3-SAT problem.

For a clause C_j, add a vertex connected to 3 of the paths such that the vertex can be visited if one of the paths is traversed in the correct direction (left to right is true, right to left is false)

Example 26.2
For a clause $C_1 = x_1 \vee \overline{x}_2 \vee x_3$:



We do this for every clause, connecting the gadget to vertices $3j$ and $3j+1$. $\square$

26.2 Coping with Intractability

- Hope that typical instances are not hard
- Use an exponential time algorithms
- Give an efficient algorithm for special uses
- Find a good enough approximate solution

26.3 Approximation Algorithms

Consider an optimization problem.
Quantify quality of an approximation solution by the **approximation ratio** γ.

- For a maximization problem, find a solution with value x so that $\frac{x}{\text{opt}} \geq \gamma$ $(\gamma \leq 1)$.
- For a minimization problem, find a solution with value x so that $\frac{x}{\text{opt}} \leq \gamma$ $(\gamma \geq 1)$.

Let's consider the **Load Balancing Problem**:
Given jobs $1, \cdots, n$ where job i takes time t_i, assign each job to one of m machines.
A_j = set of all jobs assigned to machine j.
The **load** on machine j is $\sum_{i \in A_j} t_i$.
Our goal is to minimize the maximum load, $\max_j \sum_{i \in A_i} t_i$.

This problem is NP-hard.

Simple algorithm: go through all jobs and assign each to the machine with the lowest load so far.

This does not necessarily find an optimal solution.

Theorem 26.3
This algorithm achieves an approximation ratio of 2.