8.4 Sequencing Problems

Basic genres:
- Packing problems: SET-PACKING, INDEPENDENT SET.
- Covering problems: SET-COVER, VERTEX-COVER.
- Constraint satisfaction problems: SAT, 3-SAT.
- Sequencing problems: HAMILTONIAN-CYCLE, TSP.
- Partitioning problems: 3-COLOR, 3D-MATCHING.
- Numerical problems: SUBSET-SUM, KNAPSACK.

**Hamiltonian Cycle**

**HAM-CYCLE:** given an undirected graph \( G = (V, E) \), does there exist a simple cycle \( I \) that contains every node in \( V \).

**YES:** vertices and faces of a dodecahedron.

**NO:** bipartite graph with odd number of nodes.

**Claim.** \( \text{DIR-HAM-CYCLE} \preceq_p \text{HAM-CYCLE} \).

**Pf.** Given a directed graph \( G = (V, E) \), construct an undirected graph \( G' \) with \( 3n \) nodes.

**Directed Hamiltonian Cycle**

**DIR-HAM-CYCLE:** given a digraph \( G = (V, E) \), does there exists a simple directed cycle \( I \) that contains every node in \( V \)?
Claim. $G$ has a Hamiltonian cycle iff $G'$ does.

Pf. $\Rightarrow$
- Suppose $G$ has a directed Hamiltonian cycle $\Gamma$.
- Then $G'$ has an undirected Hamiltonian cycle (same order).

Pf. $\Leftarrow$
- Suppose $G'$ has an undirected Hamiltonian cycle $\Gamma'$.
- $\Gamma'$ must visit nodes in $G'$ using one of the following two orders:
  - $\ldots, B, G, R, B, G, R, B, \ldots$
  - $\ldots, B, R, G, B, R, G, B, \ldots$
- Blue nodes in $\Gamma'$ make up directed Hamiltonian cycle $\Gamma$ in $G$, or reverse of one.

3-SAT Reduces to Directed Hamiltonian Cycle

Construction. Given 3-SAT instance $\Phi$ with $n$ variables $x_i$ and $k$ clauses.
- Construct $G$ to have $2^n$ Hamiltonian cycles.
- Intuition: traverse path $i$ from left to right $\Rightarrow$ set variable $x_i = 1$. 

3-SAT Reduces to Directed Hamiltonian Cycle

Construction. Given 3-SAT instance $\Phi$ with $n$ variables $x_i$ and $k$ clauses.
- For each clause: add a node and 6 edges.
3-SAT Reduces to Directed Hamiltonian Cycle

Claim. \( \Phi \) is satisfiable iff \( G \) has a Hamiltonian cycle.

Pf. \( \Rightarrow \)
- Suppose 3-SAT instance has satisfying assignment \( x^* \).
- Then, define Hamiltonian cycle in \( G \) as follows:
  - if \( x^*_{i,j} = 1 \), traverse row \( i \) from left to right
  - if \( x^*_{i,j} = 0 \), traverse row \( i \) from right to left
  - for each clause \( C_j \), there will be at least one row \( i \) in which we are going in "correct" direction to splice node \( C_j \) into tour

\( \Leftarrow \)
- Suppose \( G \) has a Hamiltonian cycle \( \Gamma \).
  - If \( \Gamma \) enters clause node \( C_j \), it must depart on mate edge.
    - Thus, nodes immediately before and after \( C_j \) are connected by an edge \( e \) in \( G \)
    - Removing \( C_j \) from cycle, and replacing it with edge \( e \) yields Hamiltonian cycle on \( G - \{ C_j \} \)
  - Continuing in this way, we are left with Hamiltonian cycle \( \Gamma' \) in \( G - \{ C_1, C_2, \ldots, C_k \} \).
  - Set \( x^*_{i,j} = 1 \) iff \( \Gamma' \) traverses row \( i \) left to right.
  - Since \( \Gamma' \) visits each clause node \( C_j \), at least one of the paths is traversed in "correct" direction, and each clause is satisfied.

The Longest Path

\[ \text{SHORTEST-PATH.} \text{ Given a digraph } G = (V, E), \text{ does there exists a simple path of length at most } k \text{ edges?} \]

\[ \text{LONGEST-PATH.} \text{ Given a digraph } G = (V, E), \text{ does there exists a simple path of length at least } k \text{ edges?} \]

Claim. 3-SAT \( \leq_{P} \) LONGEST-PATH.

Pf. 1. Redo proof for DIR-HAM-CYCLE, ignoring back-edge from \( t \) to \( s \).

Pf. 2. Show HAM-CYCLE \( \leq_{P} \) LONGEST-PATH.

Music. Sung to the tune of The Longest Time by Billy Joel.

Copyright © 1988 by Daniel J. Barrett.

Copyright © 1988 by Daniel J. Barrett.

The Longest Path

Lyrics. Copyright © 1988 by Daniel J. Barrett.

Music. Sung to the tune of The Longest Time by Billy Joel.

Woh-oh-oh-oh, find the longest path!
Woh-oh-oh-oh, find the longest path!
If you said P is NP tonight,
There would still be papers left to write,
I have a weakness,
I’m addicted to completeness,
And I keep searching for the longest path.

The algorithm I would like to see
Is of polynomial degree,
But it’s elusive:
Nobody has found conclusive
Evidence that we can find a longest path.

I have been hard working for so long.
I swear it’s right, and he marks it wrong.
Some how I’ll feel sorry when it’s done.
GPA 2.1
Is more than I hope for.

Gary Johnson, Karp and other men (and women)
Tried to make it order N log N.
Am I a mad fool
If I spend my life in grad school,
Forever following the longest path?

Woh-oh-oh-oh, find the longest path!
Woh-oh-oh-oh, find the longest path!
Woh-oh-oh-oh, find the longest path.

† Recorded by Dan Barrett while a grad student at Johns Hopkins during a difficult algorithms final.
Traveling Salesperson Problem

TSP. Given a set of n cities and a pairwise distance function $d(u, v)$, is there a tour of length $= D$?

13,509 cities in US with a population of at least 500
Reference: http://www.tsp.gatech.edu

Optimal TSP tour
Reference: http://www.tsp.gatech.edu

11,849 holes to drill in a programmed logic array
Reference: http://www.tsp.gatech.edu
Traveling Salesperson Problem

**TSP.** Given a set of n cities and a pairwise distance function \( d(u, v) \), is there a tour of length \( \leq D \)?

**HAM-CYCLE:** Given a graph \( G = (V, E) \), does there exist a simple cycle that contains every node in \( V \)?

**Claim.** HAM-CYCLE \( \leq \_ \) TSP.

**Pf.**
- Given instance \( G = (V, E) \) of HAM-CYCLE, create \( n \) cities with distance function 
  
  \[
  d(u, v) = \begin{cases} 
  1 & \text{if } (u, v) \in E \\
  2 & \text{if } (u, v) \notin E 
  \end{cases}
  \]

- TSP instance has tour of length \( \leq n \) iff \( G \) is Hamiltonian.

**Remark.** TSP instance in reduction satisfies \( \Delta \)-inequality.

8.5 3-Dimensional Matching

**3-Dimensional Matching**

**3D-MATCHING.** Given \( n \) instructors, \( n \) courses, and \( n \) times, and a list of the possible courses and times each instructor is willing to teach, is it possible to make an assignment so that all courses are taught at different times?

<table>
<thead>
<tr>
<th>Instructor</th>
<th>Course</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wayne</td>
<td>COS 423</td>
<td>MW 11-12:20</td>
</tr>
<tr>
<td>Wayne</td>
<td>COS 423</td>
<td>TTh 11-12:20</td>
</tr>
<tr>
<td>Wayne</td>
<td>COS 226</td>
<td>TTh 11-12:20</td>
</tr>
<tr>
<td>Wayne</td>
<td>COS 126</td>
<td>TTh 11-12:20</td>
</tr>
<tr>
<td>Tarjan</td>
<td>COS 523</td>
<td>TTh 3-4:20</td>
</tr>
<tr>
<td>Tarjan</td>
<td>COS 423</td>
<td>TTh 11-12:20</td>
</tr>
<tr>
<td>Tarjan</td>
<td>COS 423</td>
<td>TTh 3-4:20</td>
</tr>
<tr>
<td>Sedgewick</td>
<td>COS 226</td>
<td>TTh 3-4:20</td>
</tr>
<tr>
<td>Sedgewick</td>
<td>COS 226</td>
<td>MW 11-12:20</td>
</tr>
<tr>
<td>Sedgewick</td>
<td>COS 423</td>
<td>MW 11-12:20</td>
</tr>
</tbody>
</table>

**3-Dimensional Matching**

**Claim.** 3-SAT \( \leq \_ \) INDEPENDENT-COVER.

**Pf.** Given an instance \( \Phi \) of 3-SAT, we construct an instance of 3D-matching that has a perfect matching iff \( \Phi \) is satisfiable.
3-Dimensional Matching

**Construction. (part 1)**
- Create gadget for each variable \( x_i \) with 2k core and tip elements.
- No other triples will use core elements.
- In gadget \( i \), 3D-matching must use either both grey triples or both blue ones.

For each variable \( x_i \), create two elements and three triples. Exactly one of these triples will be used in any 3D-matching. Ensures any 3D-matching uses either (i) grey core of \( x_i \) or (ii) blue core of \( x_j \) or (iii) grey core of \( x_k \).

\[ C_j = x_1 \lor x_2 \lor x_3 \]

**Construction. (part 2)**
- For each variable \( x_i \), create two elements and three triples.
- In gadget \( i \), 3D-matching must use either both grey triples or both blue ones.
- Set \( x_i = \text{true} \) or \( x_i = \text{false} \).

**Construction. (part 3)**
- For each tip, add a cleanup gadget.

Claim. Instance has a 3D-matching iff \( \Phi \) is satisfiable.

Detail. What are \( X, Y, \) and \( Z \)? Does each triple contain one element from each of \( X, Y, Z \)?
3-Dimensional Matching

Claim. Instance has a 3D-matching iff \( \Phi \) is satisfiable.

Detail. What are X, Y, and Z? Does each triple contain one element from each of X, Y, Z?

8.6 Graph Coloring

Basic genres.
- Packing problems: SET-PACKING, INDEPENDENT SET.
- Covering problems: SET-COVER, VERTEX-COVER.
- Constraint satisfaction problems: SAT, 3-SAT.
- Sequencing problems: HAMILTONIAN-CYCLE, TSP.
- Partitioning problems: 3-COLOR, 3D-MATCHING.
- Numerical problems: SUBSET-SUM, KNAPSACK.

Register Allocation

Register allocation. Assign program variables to machine register so that no more than k registers are used and no two program variables that are needed at the same time are assigned to the same register.

Interference graph. Nodes are program variables names, edge between u and v if there exists an operation where both u and v are "live" at the same time.

Observation. [Chaitin, 1982] Can solve register allocation problem iff interference graph is k-colorable.

Fact. 3-COLOR \( \leq_p \) k-REGISTER-ALLOCATION for any constant \( k \geq 3 \).
3-Colorability

Claim. 3-SAT ≤p 3-COLOR.

Pf. Given 3-SAT instance Φ, we construct an instance of 3-COLOR that is 3-colorable iff Φ is satisfiable.

Construction.
i. For each literal, create a node.
ii. Create 3 new nodes T, F, B; connect them in a triangle, and connect each literal to B.
iii. Connect each literal to its negation.
iv. For each clause, add gadget of 6 nodes and 13 edges to be described next.

3-Colorability

Claim. Graph is 3-colorable iff Φ is satisfiable.

Pf. Suppose graph is 3-colorable.
• Consider assignment that sets all T literals to true.
• (ii) ensures each literal is T or F.
• (iii) ensures a literal and its negation are opposites.
• (iv) ensures at least one literal in each clause is T.

3-Colorability

Claim. Graph is 3-colorable iff Φ is satisfiable.

Pf. Suppose graph is 3-colorable.
• Consider assignment that sets all T literals to true.
• (ii) ensures each literal is T or F.
• (iii) ensures a literal and its negation are opposites.
• (iv) ensures at least one literal in each clause is T.
3-Colorability

Claim. Graph is 3-colorable iff $\Phi$ is satisfiable.

Pf. Suppose 3-SAT formula $\Phi$ is satisfiable.
   - Color all true literals $T$.
   - Color node below green node $F$, and node below that $B$.
   - Color remaining middle row nodes $B$.
   - Color remaining bottom nodes $T$ or $F$ as forced.

Planar 3-Colorability

PLANAR-3-COLOR. Given a planar map, can it be colored using 3 colors so that no adjacent regions have the same color?

YES instance.

NO instance.

Planarity

Def. A graph is planar if it can be embedded in the plane in such a way that no two edges cross.

Applications: VLSI circuit design, computer graphics.

Kuratowski's Theorem. An undirected graph $G$ is non-planar iff it contains a subgraph homeomorphic to $K_5$ or $K_{3,3}$.
Planarity Testing

**Kuratowski’s Theorem.** An undirected graph G is non-planar iff it contains a subgraph homeomorphic to $K_5$ or $K_{3,3}$.

**Brute force.** $O(n^6)$.
- Step 1. Contract all nodes of degree 2.
- Step 2. Check all subsets of 5 nodes to see if they form a $K_5$.
- Step 3. Check all subsets of 6 nodes to see if they form a $K_{3,3}$.

**Cleverness.** [Hopcroft-Tarjan 1974] $O(n)$.

A simple planar graph can have at most $3n$ edges.

**Remark.** Many intractable graph problems can be solved in poly-time if the graph is planar; many tractable graph problems can be solved faster if the graph is planar.

Polynomial-Time Detour

**Graph minor theorem.** [Robertson-Seymour 1980s]

**Corollary.** There exist an $O(n^3)$ algorithm to determine if a graph is embeddable in the torus.

**Pf of theorem.** Tour de force.

Planar 3-Colorability

**Claim.** $3$-COLOR $\leq_p$ PLANAR-$3$-COLOR.

**Proof sketch:** Given instance of $3$-COLOR, draw graph in plane, letting edges cross if necessary.
- Replace each edge crossing with the following planar gadget $W$.
  - in any 3-coloring of $W$, opposite corners have the same color
  - any assignment of colors to the corners in which opposite corners have the same color extends to a 3-coloring of $W$
Planar k-Colorability

**PLANAR-2-COLOR.** Solvable in linear time.

**PLANAR-3-COLOR.** NP-complete.

**PLANAR-4-COLOR.** Solvable in O(1) time.

**Theorem.** [Appel-Haken, 1976] Every planar map is 4-colorable.
- Resolved century-old open problem.
- Used 50 days of computer time to deal with many special cases.
- First major theorem to be proved using computer.

**False intuition.** If PLANAR-3-COLOR is hard, then so is PLANAR-4-COLOR and PLANAR-5-COLOR.

---

8.7 Numerical Problems

**Basic genres.**
- Packing problems: SET-PACKING, INDEPENDENT SET.
- Covering problems: SET-COVER, VERTEX-COVER.
- Constraint satisfaction problems: SAT, 3-SAT.
- Sequencing problems: HAMILTONIAN-CYCLE, TSP.
- Partitioning problems: 3-COLOR, 3D-MATCHING.
- **Numerical problems:** SUBSET-SUM, KNAPSACK.

---

**Subset Sum**

**SUBSET-SUM.** Given natural numbers \(w_1, \ldots, w_n\) and an integer \(W\), is there a subset that adds up to exactly \(W\)?

**Ex:** \(\{1, 4, 16, 64, 256, 1040, 1041, 1093, 1284, 1344\}\), \(W = 3754\).
**Yes.** \(1 + 16 + 64 + 256 + 1040 + 1093 + 1284 = 3754\).

**Remark.** With arithmetic problems, input integers are encoded in binary. Polynomial reduction must be polynomial in binary encoding.

**Claim.** 3-SAT \(\leq_p\) SUBSET-SUM.

**Pf.** Given an instance \(\Phi\) of 3-SAT, we construct an instance of SUBSET-SUM that has solution iff \(\Phi\) is satisfiable.

**Construction.** Given 3-SAT instance \(\Phi\) with \(n\) variables and \(k\) clauses, form \(2n + 2k\) decimal integers, each of \(n+k\) digits, as illustrated below.

**Claim.** \(\Phi\) is satisfiable iff there exists a subset that sums to \(W\).

**Pf.** No carries possible.
### Scheduling With Release Times

**SCHEDULE-RELEASE-TIMES.** Given a set of n jobs with processing time \( t_i \), release time \( r_i \), and deadline \( d_i \), is it possible to schedule all jobs on a single machine such that job \( i \) is processed with a contiguous slot of \( t_i \) time units in the interval \([r_i, d_i]\)?

**Claim.** \( \text{SUBSET-SUM} \leq_p \text{SCHEDULE-RELEASE-TIMES} \).

**Pf.** Given an instance of \( \text{SUBSET-SUM} \) \( w_1, \ldots, w_n \) and target \( W \),
- Create \( n \) jobs with processing time \( t_i = w_i \), release time \( r_i = 0 \), and no deadline \( (d_i = 1 + \sum_j w_j) \).
- Create job 0 with \( t_0 = 1 \), release time \( r_0 = W \), and deadline \( d_0 = W+1 \).

Can schedule jobs 1 to \( n \) anywhere but \([W, W+1]\)

```
0          W          W+1        S+1
```

### Polynomial-Time Reductions

- 3-SAT
- INDEPENDENT SET
- DIR-HAM-CYCLE
- GRAPH 3-COLOR
- SUBSET-SUM
- VERTEX COVER
- HAM-CYCLE
- PLANAR 3-COLOR
- SCHEDULING
- SET COVER
- TSP

packing and covering    sequencing    partitioning    numerical