CSE 521: Design and Analysis of Algorithms I

Representative Problems

Paul Beame
5 Representative Problems

- **Interval Scheduling**
  - Single resource
  - Reservation requests
    - Of form “Can I reserve it from start time $s$ to finish time $f$?”
    - $s < f$
  - **Find:** maximum number of requests that can be scheduled so that no two reservations have the resource at the same time
Interval Scheduling

- Input. Set of jobs with start times and finish times.
- Goal. Find maximum cardinality subset of mutually compatible jobs.

jobs don't overlap
Interval scheduling

Formally

- Requests 1, 2, ..., n
  - request i has start time $s_i$ and finish time $f_i > s_i$
- Requests i and j are compatible iff either
  - request i is for a time entirely before request j
    - $f_i \leq s_j$
  - or, request j is for a time entirely before request i
    - $f_j \leq s_i$
- Set $A$ of requests is compatible iff every pair of requests $i, j \in A, i \neq j$ is compatible
- Goal: Find maximum size subset $A$ of compatible requests
Interval Scheduling

- We shall see that an optimal solution can be found using a “greedy algorithm”
  - Myopic kind of algorithm that seems to have no look-ahead

- These algorithms only work when the problem has a special kind of structure

- When they do work they are typically very efficient
Weighted Interval Scheduling

- Same problem as interval scheduling except that each request \( i \) also has an associated value or weight \( w_i \)
  - \( w_i \) might be
    - amount of money we get from renting out the resource for that time period
    - amount of time the resource is being used

- **Goal**: Find compatible subset \( A \) of requests with maximum total weight
Weighted Interval Scheduling

- **Input.** Set of jobs with start times, finish times, and weights.
- **Goal.** Find maximum weight subset of mutually compatible jobs.
Weighted Interval Scheduling

- Ordinary interval scheduling is a special case of this problem
  - Take all \( w_i = 1 \)
- Problem is quite different though
  - E.g. one weight might dwarf all others
  - “Greedy algorithms” don’t work

- **Solution:** “Dynamic Programming”
  - builds up optimal solutions from smaller problems using a compact table to store them
Bipartite Matching

- A graph $G=(V,E)$ is bipartite iff
  - $V$ consists of two disjoint pieces $X$ and $Y$ such that every edge $e$ in $E$ is of the form $(x,y)$ where $x \in X$ and $y \in Y$
  - Similar to stable matching situation but in that case all possible edges were present

- $M \subseteq E$ is a matching in $G$ iff no two edges in $M$ share a vertex
- **Goal:** Find a matching $M$ in $G$ of maximum possible size
Bipartite Matching

- Input. Bipartite graph.
- Goal. Find maximum cardinality matching.
Bipartite Matching

- Models assignment problems
  - $X$ represents jobs, $Y$ represents machines
  - $X$ represents professors, $Y$ represents courses
- If $|X| = |Y| = n$
  - $G$ has perfect matching iff maximum matching has size $n$
- **Solution**: polynomial-time algorithm using “augmentation” technique
  - also used for solving more general class of network flow problems
Independent Set

- Given a graph $G=(V,E)$
  - A set $I \subseteq V$ is independent iff no two nodes in $I$ are joined by an edge
- **Goal**: Find an independent subset $I$ in $G$ of maximum possible size
- Models conflicts and mutual exclusion
Independent Set

- Input. Graph.
- Goal. Find maximum cardinality independent set.
Independent Set

- Generalizes
  - Interval Scheduling
    - Vertices in the graph are the requests
    - Vertices are joined by an edge if they are not compatible
  - Bipartite Matching
    - Given bipartite graph $G = (V, E)$ create new graph $G' = (V', E')$ where
      - $V' = E$
      - Two elements of $V'$ (which are edges in $G$) are joined if they share an endpoint in $G$
Bipartite Matching vs Independent Set

\[ G = (U \cup V, E) \]

\[ G' = (V', E') \]
Independent Set

- No polynomial-time algorithm is known
  - But to convince someone that there was a large independent set all you’d need to do is show it to them
    - they can easily convince themselves that the set is large enough and independent
  - Convincing someone that there isn’t one seems much harder
- We will show that Independent Set is NP-complete
  - Class of all the hardest problems that have the property above
Competitive Facility Location

- Two players competing for market share in a geographic area
  - e.g. McDonald’s, Burger King
- Rules:
  - Region is divided into $n$ zones, $1,\ldots,n$
  - Each zone $i$ has a value $b_i$
    - Revenue derived from opening franchise in that zone
  - No adjacent zones may contain a franchise
    - i.e., zoning regulations limit density
  - Players alternate opening franchises
- Find: Given a target total value $B$ is there a strategy for the second player that always achieves $\geq B$?
Competitive Facility Location

- Model geography by
  - A graph $G=(V,E)$ where
    - $V$ is the set $\{1,\ldots,n\}$ of zones
    - $E$ is the set of pairs $(i,j)$ such that $i$ and $j$ are adjacent zones

- Observe:
  - The set of zones with franchises will form an independent set in $G$
Competitive Facility Location

Target $B = 20$ achievable?

What about $B = 25$?
Checking that a strategy is good seems hard
- You’d have to worry about all possible responses at each round!
  - a giant search tree of possibilities
- Problem is PSPACE-complete
  - Likely strictly harder than \( \text{NP-complete} \) problems
  - PSPACE-complete problems include
    - Game-playing problems such as \( n \times n \) chess and checkers
    - Logic problems such as whether quantified boolean expressions are always true
    - Verification problems for finite automata
Five Representative Problems

- Variations on a theme: independent set.
- Interval scheduling: $n \log n$ greedy algorithm.
- Weighted interval scheduling: $n \log n$ dynamic programming algorithm.
- Bipartite matching: $n^k$ max-flow based algorithm.
- Independent set: NP-complete.
- Competitive facility location: PSPACE-complete.