Greedy Algorithms

IE 496 Lecture 15

Reading for This Lecture

- Primary
  - Horowitz and Sahni, Chapter 4

Greedy Algorithms

Combinatorial Optimization

- A combinatorial optimization problem consists of
  - A ground set E
  - An associated set F of subsets of E called the *feasible subsets*.
  - A cost vector c.
- The cost of each subset in F is the sum of the costs of the individual members.
- The goal is to find a subset of minimum cost.

Greedy Algorithms

- *Greedy algorithms* are algorithms in which the solution is constructed by adding one item at a time.
- The items are added according to a myopic *selection criteria*.
- In some cases, this leads to a globally optimal solution.
- In other cases, the procedure may be used as a heuristic for constructing solutions to a difficult problem.

Basic Algorithm

```
A is an array of the inputs
S = Ø;
for (i = 0; i < n; i++){
  x = SELECT(A);
  if (FEASIBLE(UNION(S, x))){
     S = UNION(S, x);
   }
}
```

Basic Data Structures

- SELECT

- UNION

Fractional Knapsack Problem

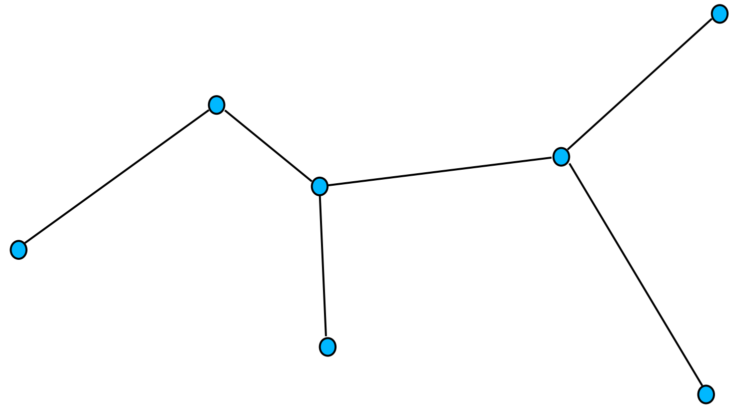
- We are given n objects.
- Each object has a weight w_i and a profit p_i.
- We also have a knapsack with capacity M.
- Objective: Fill the knapsack as profitably as possible.
- We allow fractional objects.
- Algorithm
- Analysis

Job Sequencing with Deadlines

- We are given a set of n jobs.
- Each job takes one unit of time.
- Each job has a deadline d_i and a profit p_i.
- Objective: A feasible schedule that maximizes profit.
- Algorithm
- Analysis

Spanning Trees

- We are given a graph $G = (V, E)$.
- A spanning tree of E is a *maximal acyclic subgraph* (V, T) of G.
- A spanning tree always has $|V|-1$ edges (why?).



Minimum Spanning Tree

- We associate a weight w_e with each edge e.
- Objective: Find a spanning tree of minimum weight.
- Applications

Optimality Conditions

- A spanning tree T^* is a minimum spanning tree if and only if for every tree arc (i,j) in T^*, $c_{ij} \leq c_{kl}$ for every arc (k,l) contained in the cut formed by deleting arc (i,j) from T^*.
- A spanning tree T^* is a MST if and only if for every non-tree arc (k,l) of G, $c_{ij} \leq c_{kl}$ for every arc (i,j) contained in the path in T^* connecting nodes k and l.

Kruskal's Algorithm

T is the set of edges in the tree

```
T = Ø
Sort the edges by weight
for (i = 0; i < m; i++){
  SELECT the next edge e in the list
  if (FEASIBLE(UNION(T, e)){
    UNION(T, e);
}
```

Analysis of Kruskal's Algorithm

- Correctness
- Optimality
- Implementation
- Complexity

Prim's Algorithm

S is the set of nodes in the graph

```
S = {0}
for (i = 0; i < n; i++){
  SELECT i ∉ S nearest to S;
  S = UNION(S, i);
}
```

Analysis of Prim's Algorithm

- Correctness
- Optimality
- Implementation
- Complexity

Single-source Shortest Paths

- Given an undirected graph $G = (V, E)$, a length l_e for each edge e, and a source vertex v_0.
- We are looking for the *shortest path* from v_0 to all other vertices in the graph.
- The algorithm is almost identical to Prim's MST algorithm.

Dijkstra's Algorithm

S is the set of nodes that have been examined

```
S = {0}
d[v] = c(0,v) ∀v∈V\S
for (i = 1; i < n; i++){
  SELECT w ∉ S with minimum d[w];
  S = UNION(S, w);
  set d[v] = min(d[v], d[w]+c(w,v));
}
```

Analysis of Dijkstra's Algorithm

- Correctness
- Optimality
- Implementation
- Complexity