CS 577- Intro to Algorithms

Greed (Part 2)

Dieter van Melkebeek

October 8, 2020

Outline

Discrete multivariate optimization

- System consisting of n components.
- Each component can be in any of a finite number of states.
- Want to set the states of the components so as to optimize an certain objective under certain constraints.

Outline

Discrete multivariate optimization

- System consisting of n components.
- Each component can be in any of a finite number of states.
- Want to set the states of the components so as to optimize an certain objective under certain constraints.

Paradigm

- Consider components in some order.
- Locally optimize setting of component based on prior components and settings only.

Outline

Discrete multivariate optimization

- System consisting of n components.
- Each component can be in any of a finite number of states.
- Want to set the states of the components so as to optimize an certain objective under certain constraints.

Paradigm

- Consider components in some order.
- Locally optimize setting of component based on prior components and settings only.

Correctness argument

- Greed stays ahead
- Exchanges

Outline

Discrete multivariate optimization

- System consisting of n components.
- Each component can be in any of a finite number of states.
- Want to set the states of the components so as to optimize an certain objective under certain constraints.

Paradigm

- Consider components in some order.
- Locally optimize setting of component based on prior components and settings only.

Correctness argument

- Greed stays ahead: interval scheduling
- Exchanges

Outline

Discrete multivariate optimization

- System consisting of n components.
- Each component can be in any of a finite number of states.
- Want to set the states of the components so as to optimize an certain objective under certain constraints.

Paradigm

- Consider components in some order.
- Locally optimize setting of component based on prior components and settings only.

Correctness argument

- Greed stays ahead: interval scheduling, shortest paths
- Exchanges: interval scheduling

Shortest Paths – nonnegative weights

Shortest Paths – nonnegative weights

Specification

Input: (di)graph $G = (V, E)$; lengths $\ell : E \to [0, \infty)$

Shortest Paths – nonnegative weights

Specification

Input: (di)graph $G = (V, E)$; lengths $\ell : E \to [0, \infty)$
$s, t \in V$

Shortest Paths – nonnegative weights

Specification

Input: (di)graph $G = (V, E)$; lengths $\ell : E \to [0, \infty)$
$s, t \in V$

Ouput: path P from s to t with minimum length
$\ell(P) \doteq \sum_{e \in P} \ell(e)$

Shortest Paths – nonnegative weights

Specification

Input: (di)graph $G = (V, E)$; lengths $\ell : E \to [0, \infty)$
$s, t \in V$

Ouput: path P from s to t with minimum length
$\ell(P) \doteq \sum_{e \in P} \ell(e)$

Variants

- Single pair
- Single source

Shortest Paths – nonnegative weights

Specification

Input: (di)graph $G = (V, E)$; lengths $\ell : E \to [0, \infty)$
$s, t \in V$

Ouput: path P from s to t with minimum length
$\ell(P) \doteq \sum_{e \in P} \ell(e)$

Variants

- Single pair
- Single source

Distance $d(s, t)$

$= \min\{\ell(P) \mid P \text{ path from } s \text{ to } t\}$
$= \infty$ if there is no path from s to t

Greedy Algorithm

Greedy Algorithm

Approach

Grow set S of $v \in V$ for which we know $d(s, v)$.

Greedy Algorithm

Approach

Grow set S of $v \in V$ for which we know $d(s, v)$.

Initialization

Greedy Algorithm

Approach

Grow set S of $v \in V$ for which we know $d(s, v)$.

Initialization

$S = \{s\}$ as $d(s, s) = 0$.

Greedy Algorithm

Approach

Grow set S of $v \in V$ for which we know $d(s, v)$.

Initialization

$S = \{s\}$ as $d(s, s) = 0$.

Claim

Every path P from s to some vertex in $\overline{S}$ satisfies

$$\ell(P) \geq \min_{(u,v) \in E \cap S \times \overline{S}} (d(s, u) + \ell(u, v))$$

Greedy Algorithm

Approach

Grow set S of $v \in V$ for which we know $d(s, v)$.

Initialization

$S = \{s\}$ as $d(s, s) = 0$.

Claim

Every path P from s to some vertex in $\overline{S}$ satisfies

$$\ell(P) \geq \min_{(u,v) \in E \cap S \times \overline{S}} (d(s, u) + \ell(u, v))$$

Proof

- P has to include an edge $(u, v) \in E \cap S \times \overline{S}$.
- $\ell(P) = \ell(P|_{s \rightsquigarrow u}) + \ell(u, v) + \ell(P|_{v \rightsquigarrow}) \geq d(s, u) + \ell(u, v) + 0$

Greed Stays Ahead

Claim

Every path P from s to some vertex in $\overline{S}$ satisfies

$$\ell(P) \geq \min_{(u,v) \in E \cap S \times \overline{S}} (d(s,u) + \ell(u,v))$$

Greed Stays Ahead

Claim

Every path P from s to some vertex in $\overline{S}$ satisfies

$$\ell(P) \geq \min_{(u,v) \in E \cap S \times \overline{S}} (d(s,u) + \ell(u,v))$$

Extending S

- Let $(u^*, v^*) = \arg\min_{(u,v) \in E \cap S \times \overline{S}} (d(s,u) + \ell(u,v))$

Greed Stays Ahead

Claim

Every path P from s to some vertex in $\overline{S}$ satisfies

$$\ell(P) \geq \min_{(u,v) \in E \cap S \times \overline{S}} (d(s,u) + \ell(u,v))$$

Extending S

- Let $(u^*, v^*) = \arg\min_{(u,v) \in E \cap S \times \overline{S}} (d(s,u) + \ell(u,v))$
- $d(s, v^*) = d(s, u^*) + \ell(u^*, v^*)$

Greed Stays Ahead

Claim

Every path P from s to some vertex in $\overline{S}$ satisfies

$$\ell(P) \geq \min_{(u,v) \in E \cap S \times \overline{S}} (d(s,u) + \ell(u,v))$$

Extending S

- Let $(u^*, v^*) = \arg\min_{(u,v) \in E \cap S \times \overline{S}} (d(s,u) + \ell(u,v))$
- $d(s, v^*) = d(s, u^*) + \ell(u^*, v^*)$
- Shortest path $s \rightsquigarrow u^*$ followed by (u^*, v^*) is shortest path $s \rightsquigarrow v^*$.

Greed Stays Ahead

Claim

Every path P from s to some vertex in $\overline{S}$ satisfies

$$\ell(P) \geq \min_{(u,v) \in E \cap S \times \overline{S}} (d(s,u) + \ell(u,v))$$

Extending S

- Let $(u^*, v^*) = \arg\min_{(u,v) \in E \cap S \times \overline{S}} (d(s,u) + \ell(u,v))$
- $d(s, v^*) = d(s, u^*) + \ell(u^*, v^*)$
- Shortest path $s \rightsquigarrow u^*$ followed by (u^*, v^*) is shortest path $s \rightsquigarrow v^*$.
- $S \leftarrow S \cup \{v^*\}$

Implementation

Implementation

Priority queue

Key for $v \in \overline{S}$: $\lambda(v) \doteq \min_{u \in S : (u,v) \in E}(d(s,u) + \ell(u,v))$

Implementation

Priority queue

Key for $v \in \overline{S}$: $\lambda(v) \doteq \min_{u \in S:(u,v) \in E}(d(s,u) + \ell(u,v))$

Running time with binary heap

Implementation

Priority queue

Key for $v \in \overline{S}$: $\lambda(v) \doteq \min_{u \in S:(u,v) \in E}(d(s,u) + \ell(u,v))$

Running time with binary heap

- Initialization: $O(n)$

Implementation

Priority queue

Key for $v \in \overline{S}$: $\lambda(v) \doteq \min_{u \in S:(u,v) \in E}(d(s,u) + \ell(u,v))$

Running time with binary heap

- Initialization: $O(n)$
- n min extractions: $O(n \log n)$

Implementation

Priority queue

Key for $v \in \overline{S}$: $\lambda(v) \doteq \min_{u \in S:(u,v) \in E}(d(s,u) + \ell(u,v))$

Running time with binary heap

- Initialization: $O(n)$
- n min extractions: $O(n \log n)$
- $m = \sum_{v \in V} \text{outdeg}(v)$ key updates: $O(m \log n)$

Implementation

Priority queue

Key for $v \in \overline{S}$: $\lambda(v) \doteq \min_{u \in S:(u,v) \in E}(d(s,u) + \ell(u,v))$

Running time with binary heap

- Initialization: $O(n)$
- n min extractions: $O(n \log n)$
- $m = \sum_{v \in V} \text{outdeg}(v)$ key updates: $O(m \log n)$
- Total: $O((n+m) \log n)$

Implementation

Priority queue

Key for $v \in \overline{S}$: $\lambda(v) \doteq \min_{u \in S : (u,v) \in E}(d(s,u) + \ell(u,v))$

Running time with binary heap

- Initialization: $O(n)$
- n min extractions: $O(n \log n)$
- $m = \sum_{v \in V} \text{outdeg}(v)$ key updates: $O(m \log n)$
- Total: $O((n + m) \log n)$

Better algorithms

Implementation

Priority queue

Key for $v \in \overline{S}$: $\lambda(v) \doteq \min_{u \in S : (u,v) \in E} (d(s,u) + \ell(u,v))$

Running time with binary heap

- Initialization: $O(n)$
- n min extractions: $O(n \log n)$
- $m = \sum_{v \in V} \text{outdeg}(v)$ key updates: $O(m \log n)$
- Total: $O((n+m) \log n)$

Better algorithms

- Improved data structures (Fibonacci heaps): $O(m + n \log n)$

Implementation

Priority queue

Key for $v \in \overline{S}$: $\lambda(v) \doteq \min_{u \in S:(u,v) \in E}(d(s,u) + \ell(u,v))$

Running time with binary heap

- Initialization: $O(n)$
- n min extractions: $O(n \log n)$
- $m = \sum_{v \in V} \mathrm{outdeg}(v)$ key updates: $O(m \log n)$
- Total: $O((n+m)\log n)$

Better algorithms

- Improved data structures (Fibonacci heaps): $O(m + n \log n)$
- Other approaches:
  $O(n+m)$ undirected, $O(m + n \log \log n)$ directed

From DP to Greed

From DP to Greed

- $\mathrm{OPT}(k, v) =$ length shortest path $s \rightsquigarrow v$ using $\leq k$ edges
- $\mathrm{OPT}(v) = d(s, v) = \lim_{k\to\infty} \mathrm{OPT}(k, v)$
- $\mathrm{OPT}(s) = 0$
- $\mathrm{OPT}(v) = \min_{(u,v)\in E}(\mathrm{OPT}(u) + \ell(u, v))$ for $v \neq s$
- Let S be set of $u \in V$ for which we know $\mathrm{OPT}(u)$.
- Expanding n levels until we hit S expresses $\mathrm{OPT}(v)$ for $v \notin S$ as min of:
  (a) $\mathrm{OPT}(u) + \ell(u, v)$ for each $u \in S$ with $(u, v) \in E$
  (b) $\mathrm{OPT}(u) + \ell(u, v') + \ell(P)$ for some $(u, v') \in E \cap S \times \overline{S}$ and P
  (c) $\mathrm{OPT}(u) + \ell(P)$ for some $u \notin S$ and P containing n edges
- Minimum of terms over all $v \in \overline{S}$ achieved by term of type (a).
- Finding $(u^*, v^*) = \arg\min_{(u,v)\in E\cap S\times\overline{S}}(\mathrm{OPT}(u) + \ell(u, v))$ allows extending S.
- Time complexity: $O((n + m)n) \to O((n + m)\log n)$.

Greed Stays Ahead vs Exchange Argument

Greed Stays Ahead vs Exchange Argument

Greed stays ahead

Design a quality measure for partial solutions such that:

- For every valid solution S and every point in time k, the quality measure of the greedy solution G up to k is at least as good as S up to k.
- For a full solution, optimal quality measure implies optimal objective value.

Greed Stays Ahead vs Exchange Argument

Greed stays ahead

Design a quality measure for partial solutions such that:

- For every valid solution S and every point in time k, the quality measure of the greedy solution G up to k is at least as good as S up to k.
- For a full solution, optimal quality measure implies optimal objective value.

Exchange argument

Greed Stays Ahead vs Exchange Argument

Greed stays ahead

Design a quality measure for partial solutions such that:

- For every valid solution S and every point in time k, the quality measure of the greedy solution G up to k is at least as good as S up to k.
- For a full solution, optimal quality measure implies optimal objective value.

Exchange argument

Consider an optimal solution S. Establish a sequence of local transformations (exchanges) such that:

Greed Stays Ahead vs Exchange Argument

Greed stays ahead

Design a quality measure for partial solutions such that:

- For every valid solution S and every point in time k, the quality measure of the greedy solution G up to k is at least as good as S up to k.
- For a full solution, optimal quality measure implies optimal objective value.

Exchange argument

Consider an optimal solution S. Establish a sequence of local transformations (exchanges) such that:

- Each transformation maintains validity and does not deteriorate the objective value.

Greed Stays Ahead vs Exchange Argument

Greed stays ahead

Design a quality measure for partial solutions such that:

- For every valid solution S and every point in time k, the quality measure of the greedy solution G up to k is at least as good as S up to k.
- For a full solution, optimal quality measure implies optimal objective value.

Exchange argument

Consider an optimal solution S. Establish a sequence of local transformations (exchanges) such that:

- Each transformation maintains validity and does not deteriorate the objective value.
- The sequence ends in the greedy solution G.

Interval Scheduling

Interval Scheduling

Problem

Input: meetings $i \in [n]$ specified by start time $s_i \in \mathbb{R}$ and end time $e_i \in \mathbb{R}$.

Ouput: $S \subseteq [n]$ such that no distinct intervals $[s_i, e_i)$ for $i \in S$ overlap and $|S|$ is maximized.

Interval Scheduling

Problem

Input: meetings $i \in [n]$ specified by start time $s_i \in \mathbb{R}$ and end time $e_i \in \mathbb{R}$.

Ouput: $S \subseteq [n]$ such that no distinct intervals $[s_i, e_i)$ for $i \in S$ overlap and $|S|$ is maximized.

Greedy algorithm

- Order: earliest end time first
- Local criterion

Exchange Argument

Exchange Argument

- Consider an optimal solution S that differs from G.
- There exists a first meeting i in the greedy order on which S differs from G.
- It has to be the case that $i \in G$ and $i \notin S$.
- There exists a meeting $j > i$ such that $j \in S$.
- $S' \doteq S \setminus \{j\} \cup \{i\}$ is an optimal solution.
- The first meeting i' on which S' differs from G (if any) satisfies $i' > i$.
- As there are only a finite number of meetings, the process has to end in G.