Learning to Prune:
Speeding up Repeated Computations

Ellen Vitercik with

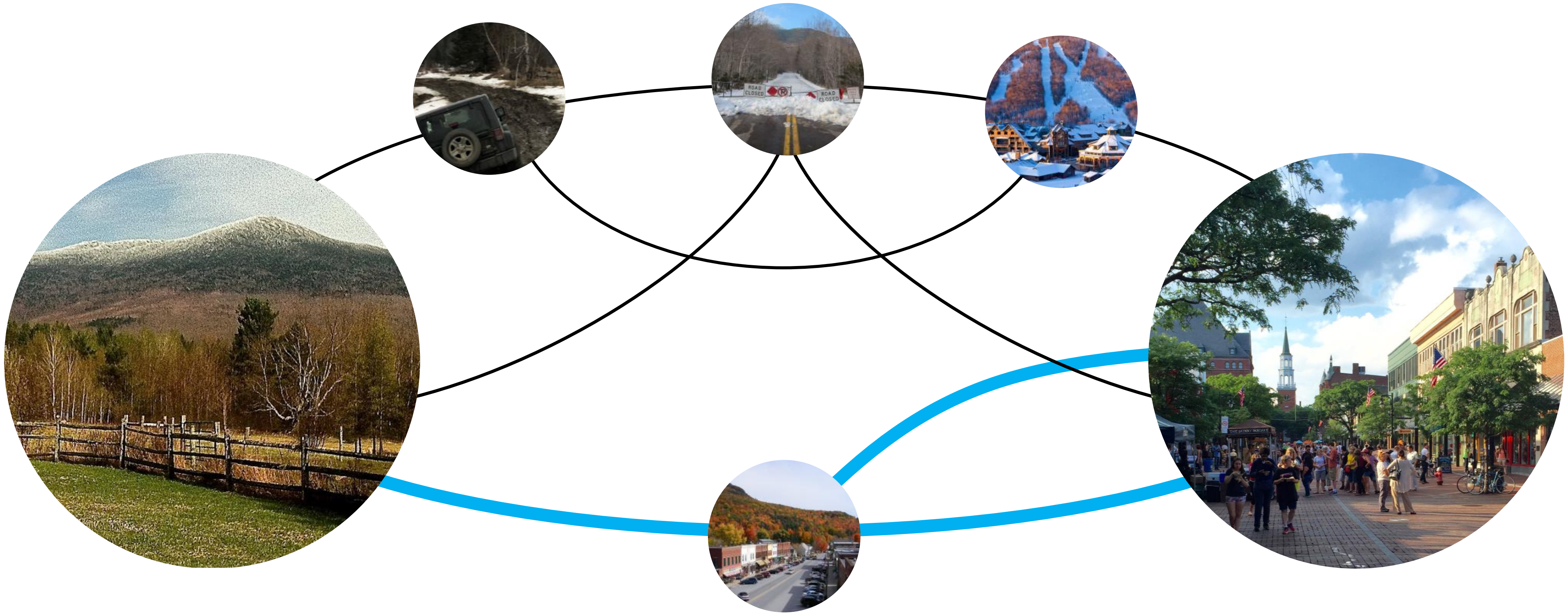
Daniel Alabi, Adam Kalai, Katrina Ligett, Cam Musco, Christos Tzamos





LINCOLN

BURLINGTON



Dijkstra’s algorithm wastes time on those muddy mountain roads

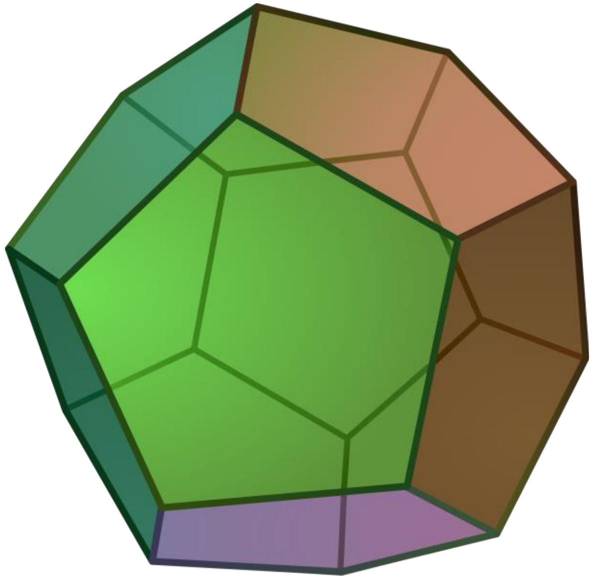
GOAL

Solve sequence of similar problems, exploiting common structure

Speeding up repeated computations

Typically, large swaths of search space **never** contain solutions…

Learn to ignore them!



Only handful of LP constraints ever bind

Large portions of DNA strings never contain patterns of interest

Model

Function $f: X \to Y$ maps problem instances x to solutions y

Learning algorithm receives sequence $x_1, \ldots, x_T \in X$

E.g., each $x_i \in \mathbb{R}^{|E|}$ *equals edge weights for fixed road network*

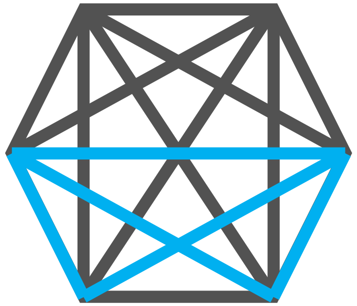
Model

Goal: Correctly compute f on *most* rounds, minimize runtime

Worst-case algorithm would compute and return $f(x_i)$ *for each* x_i

Assume access to other functions mapping $X \to Y$

- Faster to compute
- Defined by subsets (*prunings*) S of universe $\mathcal{U}$
  - Universe $\mathcal{U}$ represents entire search space
  - Denote corresponding function $f_S: X \to Y$
  - $f_{\mathcal{U}} = f$



Example:

$\mathcal{U}$ = all edges in fixed graph

S = subset of edges

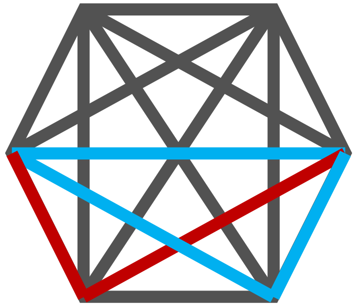
Model

Goal: Correctly compute f on *most* rounds, minimize runtime

Worst-case algorithm would compute and return $f(x_i)$ for each x_i

Assume access to other functions mapping $X \to Y$

- Faster to compute
- Defined by subsets (*prunings*) S of universe $\mathcal{U}$
  - Universe $\mathcal{U}$ represents entire search space
  - Denote corresponding function $f_S: X \to Y$
  - $f_{\mathcal{U}} = f$



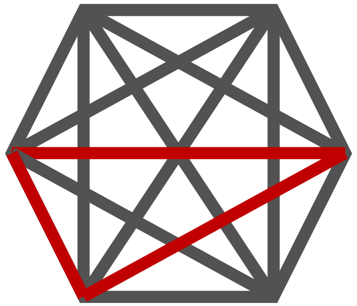
Assume exists set $S^*(x) \subseteq \mathcal{U}$ where $f_S(x) = f(x)$ iff $S^*(x) \subseteq S$

- "Minimally pruned set"
- E.g., the shortest path

Algorithm

1. Initialize pruned set $\bar{S}_1 \leftarrow \emptyset$
2. For each round $j \in \{1, \dots, T\}$:
   a. Receive problem instance x_j
   b. With probability $1/\sqrt{j}$, **explore**:
      i. Output $f(x_j)$
      ii. Compute minimally pruned set $S^*(x_j)$
      iii. Update pruned set: $\bar{S}_{j+1} \leftarrow \bar{S}_j \cup S^*(x_j)$
   c. Otherwise (with probability $1 - 1/\sqrt{j}$), **exploit**:
      i. Output $f_{\bar{S}_j}(x_j)$
      ii. Don't update pruned set: $\bar{S}_{j+1} \leftarrow \bar{S}_j$

Guarantees



Recap: At round j, algorithm outputs $f_{S_j}(x_j)$.

S_j depends on $x_{1:j}$.

Goal 1: Minimize $|S_j|$

In our applications, time it takes to compute $f_{S_j}(x_j)$ grows with $|S_j|$

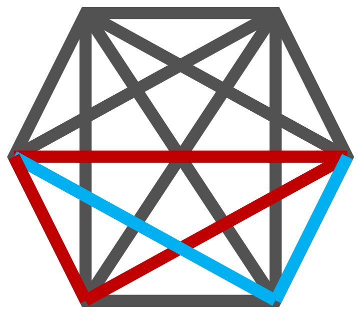
Theorem: $\mathbb{E}\left[\frac{1}{T}\sum_{j=1}^{T}|S_j|\right] \leq |S^*| + \frac{|\mathcal{U}|-|S^*|}{\sqrt{T}}$, where $S^* = \bigcup_{j=1}^{T} S^*(x_j)$

Proof: $\mathbb{E}[|S_j|] = \frac{1}{\sqrt{j}}|\mathcal{U}| + \left(1-\frac{1}{\sqrt{j}}\right)\mathbb{E}[|\bar{S}_j|] \leq \frac{1}{\sqrt{j}}|\mathcal{U}| + \left(1-\frac{1}{\sqrt{j}}\right)|S^*|$

Guarantees

Recap: At round j, algorithm outputs $f_{S_j}(x_j)$.

S_j depends on $x_{1:j}$.



Goal 2: Minimize # of mistakes

Rounds where $f_{S_j}(x_j) \neq f(x_j)$

Theorem: $\mathbb{E}[\text{\# of mistakes}] \leq \frac{|S^*|}{\sqrt{T}}$, where $S^* = \cup_{j=1}^{T} S^*(x_j)$

See poster for proof sketch.

Goal: Reach bottom star from top star

Grey nodes: Nodes Dijkstra's algorithm explores over 30 rounds

Black nodes: Nodes in the pruned subgraph

Fraction of mistakes: 0.068 over 5000 runs of the algorithm, $T = 30$ rounds each

Conclusion

Algorithm for quickly solving series of related problems

It learns to prune irrelevant regions of solution space

With high probability, algorithm makes few mistakes

It may prune large swaths of the search space

Learning to Prune:
Speeding up Repeated Computations

Ellen Vitercik with

Daniel Alabi, Adam Kalai, Katrina Ligett, Cam Musco, Christos Tzamos