Online linear optimization and adaptive routing

Baruch Awerbuch, Robert Kleinberg
Motivation

- Overlay network routing – Send a packet from source to target using the route with minimum delay
- The total route delay is revealed
- Graph example
Using previous algorithms

- We can use EXP3. Each route is an arm. Since we have $n!$ routes, our regret will be
  
  $$O\left(\sqrt{\left(K G_{\text{max}} \ln K\right)}\right) \rightarrow O\left(\sqrt{\left(n! \ln n!\right)}\right)$$

- We have seen online shortest paths with (full information)
  
  $$E\left[\text{cost}\right] \leq (1 + \varepsilon) \mincost_T + O\left(mn \log n / \varepsilon\right)$$
Problem definition

- $G=(V,E)$ – Directed graph
- For each $j = 1, \ldots, T$ the adaptive adversary select cost for each edge $c_j : E \rightarrow [0,1]$
- The algorithm select a path of length $\leq H$
- Receive cost of the entire path
- Goal to minimize the difference between the algorithm's expected total cost and the cost of the best single path from source to target
Regret

\[ O\left(H^2 (mH \log \Delta \log mHT)^{1/3} T^{2/3}\right) \]
Pre-processing

- We will transform the graph $G$ to a leveled directed acyclic graph $\tilde{G}=(\tilde{V}, \tilde{E})$

- Start by calculating $G \times \{0, 1, \ldots, H\}$
  - Vertex set $V \times \{0, 1, \ldots, H\}$
  - $e_i$ from $(u, i - 1)$ to $(v, i)$ for every $e=(u, v)$ in $E$

- The graph $\tilde{G}$ is obtained by:
  - Deleting paths that doesn't reach to $r$
Main idea

- We can traverse the graph by querying BEX for probabilities on the outgoing edges until we reach r.
- To do so we need to feed BEX with information on all experts.
- We will run in phases, at each phase we will estimate the cost for all experts. At the end of each phase we will update BEX.
- We will feed BEX with the total path cost.
Sampling experts

- We can sample the experts according to the distribution BEX returns (according to the previous phases costs)
- The problem – We might ignore some edges that might be better at next phases
- We will add some exploration steps at each phase
Exploration

- Will occur with probability $\delta$
- Choose an edge $e=(u,v)$ uniformly at random
- Construct a path by joining $\text{prefix}(u)$, $e$ and $\text{suffix}(v)$
**Suffix**

- Suffix(v) will return the distribution on s – v paths
- Implementation – Choose edge by BEX probabilities, traverse the edge, repeat until r is reached
- Why can't it be random?
Prefix

- Prefix(v) – Will return the distribution on s - v paths
- Let suffix(u | v) be the distribution on u – v paths
- Obtained by sampling from suffix(u) conditional to the event that the path passes through v.
Prefix

• Sample from suffix(s | v) with probability

\[
(1 - \delta) \frac{Pr(v \in \text{suffix}(s))}{P_\Phi(v)}
\]

• For all e = (q,u) from $\tilde{E}$, with probability

\[
(\delta/\tilde{m}) \frac{Pr(v \in \text{suffix}(u))}{P_\Phi(v)}
\]
sample from suffix(u | v) prepend e and then prepend a sample from prefix(q)

• Where $P_\Phi(v)$ is the probability v is contained in the suffix of a path in phase $\Phi$
Updating costs

- Phase length \( \tau = \left\lfloor \frac{2 mH \log(mH T)}{\delta} \right\rfloor \)
- At each phase we will sum the costs for each edge only if the edge wasn't part of the path chosen by prefix
- The reason for that is that we cannot control the probability those edges came from
Updating costs

At the end of each phase

\( \forall e \in \tilde{E} \),

\[
\mu_\phi(e) \leftarrow E[\sum_{j \in \tau_\phi} \chi_j(e)]
\]

\[
\tilde{c}_\phi(e) \leftarrow (\sum_{j \in \tau_\phi} \chi_j(e) c_j(\pi_j))/\mu_\phi(e)
\]

Where

\( \phi = 1, \ldots, \lceil T/\tau \rceil \)

\( j = \tau(\phi-1) + 1, \tau(\phi-1) + 2, \ldots, \tau \phi \)
Algorithm analysis

- Let

\[
C^-(v) = \sum_{j=1}^{T} E[c_j(prefix(v))] \\
C^+(v) = \sum_{j=1}^{T} E[c_j(suffix(v))] \\
OPT(v) = \min_{\text{paths } \pi: v \to r} \sum_{j=1}^{T} c_j(\pi)
\]
Algorithm analysis

- We know that for BEX
  \[ \sum_{j=1}^{t} \sum_{i=1}^{K} p_j(i) c_j(i) \leq \sum_{j=1}^{t} c_j(k) + O(\epsilon t + \log K / \epsilon) M \]

- Let \( p_\phi \) be the probability distribution supplied by BEX(v) during phase \( \phi \)
  \[ \sum_{\phi=1}^{t} \sum_{e \in \Delta(v)} p_\phi(e) \tilde{c}_\phi(e) \leq \sum_{\phi=1}^{t} \tilde{c}_\phi(e_0) + O(\epsilon H t + H \log \Delta / \epsilon) \]
Algorithm analysis

- We used the fact that cost of a phase $M$ is smaller than $3H$ with high probability. By Chernoff bound

$$\tau = \frac{2mH \log(mHT)}{\delta}$$

$$\mu_\phi > \frac{\delta \tau}{mH} = 2 \log(mHT)$$

$$\Pr\left( \sum_{j \in \tau_\phi} \chi_j \geq 3 \times 2 \log(mHT) \right) \leq e^{-\frac{2}{3} \frac{2 \log(mHT)}{mHT}} \leq \frac{1}{mHT}$$
Algorithm analysis

• Now by applying union bound over all phases we get that this low probability event contributes at most $HT / (mHT) < 1$. So we will ignore this event.
Algorithm analysis

- Expanding $\tilde{c}_\phi$

\[
\sum_{\phi=1}^{t} \sum_{e \in \Delta(v)} \sum_{j \in \tau_{\phi}} p_\phi(e) \chi_j(e) c_j(\pi_j) / \mu_\phi(e) \leq \sum_{\phi=1}^{t} \sum_{j \in \tau_{\phi}} \chi_j(e_0) c_j(\pi_j) / \mu_\phi(e_0) + O(\epsilon Ht + \frac{H}{\epsilon} \log \Delta)
\]
Algorithm analysis

- Claim 3.2.

\[ Pr(\pi \subset_{\pi_j} | \chi_j(e) = 1) = Pr(prefix(v) = \pi) \]

\(\pi : s \rightarrow v\)
Algorithm analysis

- Proof of claim 3.2

\[ \chi_j(e) = 1 \rightarrow e \in \pi_j^0 \lor e \in \pi_j^+ \]

\[ Pr(\pi \subseteq \pi_j | e \in \pi_j^0) = Pr(\text{prefix}(v) = \pi) \]

\[ Pr(\pi \subseteq \pi_j | e \in \pi_j^+) = Pr(\text{prefix}(v) = \pi) \]

- The first claim is by definition, let's prove the second claim
Algorithm analysis

- e is sampled independently from the path preceding v, so

\[ Pr(\pi \subseteq \pi_j | e \in \pi^+_j) = Pr(\pi \in \pi_j | v \in \pi^+_j) \]

\[ Pr(v \in \pi^+_j) Pr(\pi \subseteq \pi_j | v \in \pi^+_j) = Pr(\pi \subseteq \pi_j \cap v \in \pi^+_j) \]

\[ = (1 - \delta) Pr(v \in \text{suffix}(s)) Pr(\pi = \text{suffix}(s | v)) \]

\[ + \sum_{e = (q, u) \in \tilde{E}} \delta \frac{\delta}{\tilde{m}} Pr(v \in \text{suffix}(u)) Pr(\pi = \text{prefix}(q) \cup \{e\} \cup \text{suffix}(u | v)) \]

\[ = Pr(v \in \pi^+_j) Pr(\pi = \text{prefix}(v)) \]
Algorithm analysis

- Claim 3.3. If $e = (v, w)$ then

$$E[\chi_j(e)c_j(\pi_j)] = \left(\frac{u(e)}{\tau}\right)(A_j(v) + B_j(w) + c_j(e))$$

$$A_j(v) = E[c_j(prefix(v))]$$

$$B_j(w) = E[c_j(suffix(w))]$$

- Follows from claim 3.2 that the portion of the path preceding $e$ is distributed by prefix($v$)
Algorithm analysis

- Taking the expectation of eq.12

The left side will become

\[
\sum_{\phi=1}^{T} \sum_{e \in \Delta(v)} \sum_{j \in \tau_{\phi}} p_{\phi}(e)(A_j(v) + B_j(w) + c_j(e))
\]

\[
= \frac{1}{\tau} \sum_{j=1}^{T} \sum_{e \in \Delta(v)} p_{\phi}(e)(A_j(v) + B_j(w) + c_j(e))
\]

- The right side will become

\[
\frac{1}{\tau} \sum_{j=1}^{T} (A_j(v) + B_j(w_0) + c_j(e_0))
\]
Algorithm analysis

- After removing \( A_j(v) \) from both sides and notice that
  \[
  \sum_{e \in \Delta(v)} p_{\phi}(e) (B_j(w) + c_j(e)) = E[c_j(\text{suffix}(v))]
  \]

- So the left side will become
  \[
  \frac{1}{\tau} \sum_{j=1}^{T} E[c_j(\text{suffix}(v))] = c^+(v)/\tau
  \]
Algorithm analysis

- The right side will become

\[
\frac{1}{\tau} \sum_{j=1}^{T} E[c_j(suffix(v))] + c^+(w_0)/\tau + O(\epsilon Ht + \frac{H}{\epsilon} \log \Delta)
\]

- Thus we have derived the local performance guarantee (Eq.13)

\[
c^+(v) \leq c^+(w_0) + \sum_{j=1}^{T} c_j(e_0) + O(\epsilon HT + \frac{\tau}{\epsilon} H \log \Delta)
\]
Global performance guarantee

• Claim 3.4

\[ c^+ (v) \leq \text{OPT} (v) + O (\epsilon HT + \frac{T}{\epsilon} H \log \Delta) h(v) \]

• To prove we can use the following observation

\[ \text{OPT} (v) = \min_{e_0 = (v, w_0)} \left\{ \sum_{j=1}^{T} c_j (e_0) + \text{OPT} (w_0) \right\} \]
Global performance guarantee

- Proof – By induction on \( h(v) \) and by using the local performance guarantee
- Lets mark

\[
F = O(\epsilon Ht + \tau \frac{H}{\epsilon} \log \Delta)
\]

- Now rewrite the claim and eq.13

\[
c^+(v) \leq OPT(v) + F h(v)
\]

\[
c^+(v) \leq c^+(w_0) + \sum_{j=1}^{T} c_j(e_0) + F
\]
Global performance guarantee

- \( h(v) = 1 \)

\[
c^+ (v) \leq \text{OPT} (v) + F = \sum_{j=1}^{T} c_j(e_0) + \text{OPT} (r) + F : \forall e_0 = (v, r)
\]

\[
c^+ (v) \leq \sum_{j=1}^{T} c_j(e_0) + F : \forall e_0 = (v, r)
\]

It's true by the local performance guarantee
Global performance guarantee

- \( h(v) = k + 1 \)

\[
\begin{align*}
  c^+ (v) &\leq c^+ (v_k) + \sum_{j=1}^{T} c_j (e_{k+1}) + F \\
  &\leq \sum_{j=1}^{T} c_j (e_{k+1}) + OPT (v_k) + kF + F \\
  &= OPT (v_{k+1}) + (k + 1) F
\end{align*}
\]
Regret

- Theorem 3.5. The algorithm suffers regret
  \[ O \left( H^2 \left( mH \log \Delta \log mHT \right)^{1/3} T^{2/3} \right) \]
- The exploration step contributes \( \delta \cdot TH \)
- The exploitation contributes \( c^+ (s) - OPT (s) \)
- Also \( \tau = 2 mH \log (mH T) / \delta \)
- Substituting in claim 3.4 we get total exploitation cost
  \[ c^+ (s) - OPT (s) = O \left( \epsilon T + 2mH \log \Delta \frac{\log (mhT)}{\epsilon \delta} \right) H^2 \]
Regret

\[ \text{Regret} \leq O \left( \delta T + \epsilon T + \frac{2mH \log \Delta \log (mhT)}{\epsilon \delta} \right) H^2 \]

- We can assign
  \[ \epsilon = \delta = \left(2mH \log \Delta \log (mhT)\right)^{1/3} T^{-1/3} \]

And we will get the desired regret

\[ O \left( H^2 \left( mH \log \Delta \log mHT \right)^{1/3} T^{2/3} \right) \]