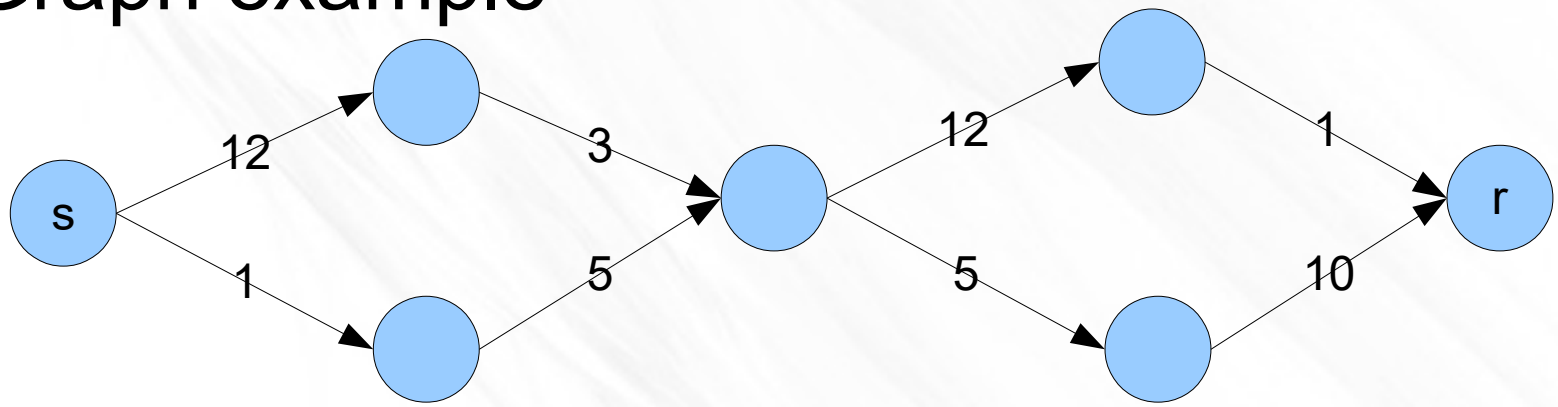
Online linear optimization and adaptive routing

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 a arm. Since we have n! routes, our regret will be

$$O\left(\sqrt{\left(K G_{max} \ln K\right)}\right) \rightarrow O\left(\sqrt{\left(n! \ln n!\right)}\right)$$

- We have seen online shortest paths with (full information)

$$E[cost] \leq (1+\epsilon) mincost_T + O(mn \log n / \epsilon)$$

Problem definition

- G=(V,E) – Directed graph
- For each j = 1, …, T the adaptive adversary select cost for each edge $c_j : E \rightarrow [0,1]$
- The algorithm select a path of length ≤ 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

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

Pre-processing

- We will transform the graph G to a leveled directed acyclic graph $\tilde{G}=(\tilde{V},\tilde{E})$
- Start by calculating G x {0, 1, …, H}
  - Vertex set V x {0, 1, …, 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

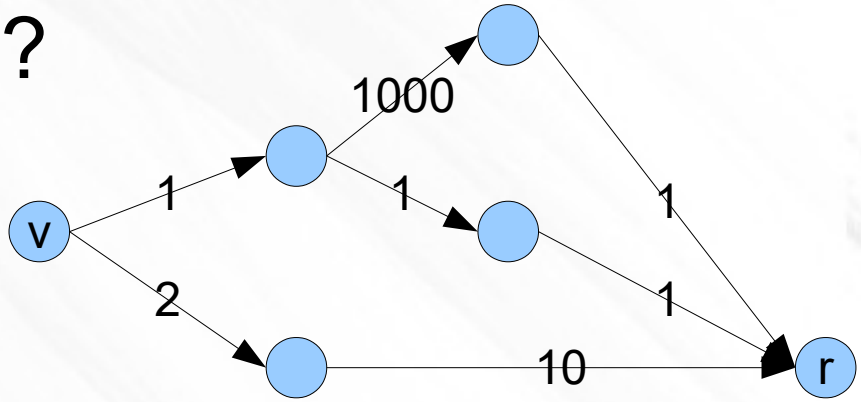
- We can sample the experts according to the distribution BEX returns (according to the previous phases costs)
- The problem – We might ignore some edges that might be better at next phases
- We will add some exploration steps at each phase

Exploration

- Will occur with probability δ
- Choose an edge $e=(u,v)$ uniformly at random
- Construct a path by joining prefix(u), e and 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

$$(1-\delta)\,Pr(v \in suffix(s))/P_{\phi}(v)$$

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

$$(\delta/\tilde{m})\,Pr(v \in suffix(u))/P_{\phi}(v)$$

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 Φ

Updating costs

- Phase length $\tau = \lceil 2\, mH \log(mH\, T)/\delta \rceil$
- 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

$$C^{-}(v)=\sum_{j=1}^{T} E[c_j(prefix(v))]$$

$$C^{+}(v)=\sum_{j=1}^{T} E[c_j(suffix(v))]$$

$$OPT(v)=min_{paths\,\pi:v\rightarrow r}\sum_{j=1}^{T} c_j(\pi)$$

Algorithm analysis

- We know that for BEX

$$\sum_{j=1}^{t}\sum_{i=1}^{K} p_j(i)c_j(i) \le \sum_{j=1}^{t} c_j(k) + O(\epsilon t + \log K/\epsilon)M$$

- Let p_ϕ be the probability distribution supplied by BEX(v) during phase ϕ

$$\sum_{\phi=1}^{t}\sum_{e\in\Delta(v)} p_\phi(e)\tilde{c}_\phi(e) \le \sum_{\phi=1}^{t} \tilde{c}_\phi(e_0) + O(\epsilon H t + H\log\Delta/\epsilon)$$

Algorithm analysis

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

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

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

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

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 $\tilde{c}_\phi$

$$(Eq.12)\ \sum_{\phi=1}^{t} \sum_{e\in\Delta(v)} \sum_{j\in\tau_\phi} p_\phi(e)\chi_j(e)c_j(\pi_j)/\mu_\phi(e)$$

$$\leq \sum_{\phi=1}^{t} \sum_{j\in\tau_\phi} \chi_j(e_0)c_j(\pi_j)/\mu_\phi(e_0) + O\left(\epsilon Ht + \frac{H}{\epsilon}\log\Delta\right)$$

Algorithm analysis

- Claim 3.2.

$$Pr(\pi \subset \pi_j \mid \chi_j(e)=1) = Pr(prefix(v)=\pi)$$

$$\pi : s \rightarrow v$$

Algorithm analysis

- Proof of claim 3.2

$$\chi_j(e)=1 \rightarrow e \in \pi_j^0 \vee e \in \pi_j^+$$

$$Pr(\pi \subseteq \pi_j | e \in \pi_j^0) = Pr(prefix(v) = \pi)$$

$$Pr(\pi \subseteq \pi_j | e \in \pi_j^+) = Pr(prefix(v) = \pi)$$

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

Algorithm analysis

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

$$Pr(\pi \subseteq \pi_j \mid e \in \pi_j^+) = Pr(\pi \in \pi_j \mid v \in \pi_j^+)$$

$$Pr(v \in \pi_j^+) Pr(\pi \subseteq \pi_j \mid v \in \pi_j^+) = Pr(\pi \subseteq \pi_j \cap v \in \pi_j^+)$$

$$= (1-\delta) Pr(v \in suffix(s)) Pr(\pi = suffix(s \mid v))$$

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

$$= Pr(v \in \pi_j^+) Pr(\pi = prefix(v))$$

Algorithm analysis

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

$$E[\chi_j(e)c_j(\pi_j)]=(\mu(e)/\tau)(A_j(v)+B_j(w)+c_j(e))$$

$$A_j(v)=E[c_j(prefix(v))]$$
$$B_j(w)=E[c_j(suffix(w))]$$

- 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

$$\sum_{t}\sum_{\phi=1}\sum_{e\in\Delta(v)}\sum_{j\in\tau_\phi} p_\phi(e)\left(A_j(v)+B_j(w)+c_j(e)\right)$$

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

- The right side will become

$$\frac{1}{\tau}\sum_{j=1}^{T}\left(A_j(v)+B_j(w_0)+c_j(e_0)\right)$$

Algorithm analysis

- After removing $A_j(v)$ from both sides and notice that

$$\sum_{e \in \Delta(v)} p_\phi(e)(B_j(w) + c_j(e)) = E[c_j(suffix(v))]$$

- So the left side will become

$$\frac{1}{\tau} \sum_{j=1}^{T} E[c_j(suffix(v))] = c^+(v)/\tau$$

Algorithm analysis

- The right side will become

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

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

$$c^+(v) \le c^+(w_0) + \sum_{j=1}^{T} c_j(e_0) + O\left(\epsilon HT + \frac{\tau}{\epsilon} H\log\Delta\right)$$

Global performance guarantee

- Claim 3.4

$$c^{+}(v) \leq OPT(v) + O\left(\epsilon HT + \frac{\tau}{\epsilon} H \log \Delta\right) h(v)$$

- To prove we can use the following observation

$$OPT(v) = min_{e_0=(v,w_0)} \left\{ \sum_{j=1}^{T} c_j(e_0) + OPT(w_0) \right\}$$

Global performance guarantee

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

$$F = O\left(\epsilon H t + \tau \frac{H}{\epsilon} \log \Delta\right)$$

- 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

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

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

It's true by the local performance guarantee

Global performance guarantee

- h(v)=k+1

$$c^{+}(v) \le c^{+}(v_k) + \sum_{j=1}^{T} c_j(e_{k+1}) + F$$

$$\le \sum_{j=1}^{T} c_j(e_{k+1}) + OPT(v_k) + kF + F$$

$$= OPT(v_{k+1}) + (k+1)F$$

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 δTH
- The exploitation contributes $c^+(s)-OPT(s)$
- Also $\tau=2mH\log\left(mHT\right)/\delta$
- Substituting in claim 3.4 we get total exploitation cost

$$c^+(s)-OPT(s)=O\left(\epsilon T+2\mathrm{mH}\log\Delta\frac{\log\left(mhT\right)}{\epsilon\delta}\right)H^2$$

Regret

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

- We can assign

$$\epsilon = \delta = \left(2\mathrm{mH}\log\Delta\log\left(mhT\right)\right)^{1/3} T^{-1/3}$$

And we will get the desired regret

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