Online incremental network coding for multiple unicasts

Tracey Ho, Ralf Koetter
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign

Online incremental network coding for multiple unicasts

- We have an approach adding a unicast connection over existing connections in a network
- This is useful as part of an online algorithm that considers connections one at a time, and decides whether and how to add each
- We can extend the approach of Awerbuch et al. [AAP93] on throughput-competitive online routing to network coding

Problem statement

- sequentially consider connection requests $\beta_j, j = 1, 2, \ldots, k$, each specifying source s_j, destination d_j, duration, bandwidth and an associated reward
- goal is to maximize total reward (e.g. throughput)
- online algorithm either allocates sufficient capacity to accommodate request or rejects request; no rerouting of connections
- Evaluate online algorithm by *competitive ratio*: supremum, over all possible input sequences, of the ratio of the optimal offline reward to the online reward

Notation and assumptions

- m links in network
- reward ρ_j of j^{th} request bounded within range $[1, R]$
- consider single period and unit bandwidth requests for simplicity
- capacity u_e of each link e bounded within range $[\log \mu, P]$, where $P \geq 1$ and $\mu = 2mPR$

Online routing approach of Awerbuch et al.

- Online strategy:
  - assign link costs exponential in fractional usage
  - accept a connection only if "worthwhile" compared to costs
- Competitive ratio of $O(\log n)$ achieved, where n is the number of nodes

Incorporating network coding

- allow up to two connections to share capacity by being coded together
- associate with each edge e a cost, after considering $j-1$ requests, based on fraction $\lambda_{e,x}(j)$ of its capacity already assigned to $x = 1, 2$ or more flows:

$$c_e(j) = \frac{1}{m}\mu^{\lambda_{e,1}(j)+\alpha\lambda_{e,2}(j)\left(1-\frac{1}{u_e}-\lambda_{e,1}(j)\right)^+}$$

  where $\alpha < 1$

Online strategy

- algorithm accommodates request β_j iff it can find a solution such that

$$\begin{aligned}
\sum_e \frac{(a_{e,1} + \alpha a_{e,2})\, c_e(j)}{u_e} &\leq \rho_j \\
\frac{a_{e,2}}{u_e} &\leq \lambda_{e,1}(j) - \lambda_{e,2}(j) \;\; \forall \;\; e \;\; \& \\
\frac{a_{e,1} + \alpha a_{e,2}}{u_e} &\leq \frac{1}{\log \mu} \;\; \forall \;\; e
\end{aligned}$$

  where $a_{e,x}$ is the capacity of e used in adding β_j that becomes shared by $x = 1, 2$ connections

Finding a low-cost solution for a connection

- by linear programming based on the equations described earlier, or
- by combinatorial methods, e.g.
  - greedy approach similar to Dijkstra's algorithm, but involving two shortest path computations to determine cost of sharing a link
  - admits merging of remedy paths
  - larger solution space than the linear programming approach, but may not obtain optimal cost

Performance

- A $(2 \log \mu + 3)$ competitive ratio is obtained as long as each connection added uses a solution costing no more than the lowest cost routing-only solution
- Any online algorithm has a throughput competitive ratio of $\Omega(\log(mR))$
  - proof uses a line network with requests of exponentially decreasing resource requirements/exponentially increasing rewards [AAP93]

Enforcing capacity constraints

- When $\lambda_{e,1}(j) > 1 - \frac{1}{u_e}$, using the assumption that $\log \mu \leq u_e \leq P$, we have

$$\begin{aligned} \frac{c_e(j)}{u_e} &> \frac{\mu^{1-\frac{1}{\log \mu}}}{mu_e} \\ &\geq \frac{\mu}{2mP} \\ &= R, \end{aligned}$$

  so capacity will not be exceeded

Lower bounding online performance

- If request β_j is admitted,

$$\begin{aligned}
\sum_e \left(c_e(j+1) - c_e(j)\right) &\leq \sum_e c_e(j)\left(\mu^{\frac{a_{e,1}+\alpha a_{e,2}}{u_e}} - 1\right) \\
&\leq \sum_e c_e(j)\left(\frac{a_{e,1}+\alpha a_{e,2}}{u_e}\right)\log\mu \\
&\leq \rho_j \log\mu
\end{aligned}$$

- Summing over set $\mathcal{A}$ of all accepted requests,

$$\begin{aligned}
\sum_{j\in\mathcal{A}} \rho_j \log\mu &\geq \sum_e \left(c_e(k+1) - \frac{1}{m}\right) \\
&= \sum_e c_e(k+1) - 1,
\end{aligned}$$

where k is the total number of requests.

Upper bounding off-line performance

- If request j is admitted by the off-line algorithm but not the online algorithm,

$$\rho_j \quad < \quad \sum_{e \in P'_j} \frac{c_e(j)}{u_e} \leq \sum_{e \in P'_j} \frac{c_e(k+1)}{u_e}$$

  where P'_j is the path from s_j to t_j that forms part of the set of links used by the off-line algorithm to accommodate β_j

- Total reward from the set $\mathcal{Q}$ of these requests is at most

$$\begin{aligned} \sum_{j \in \mathcal{Q}} \sum_{e \in P'_j} \frac{c_e(k+1)}{u_e} &= \sum_{e:\, e \in P'_j,\, j \in \mathcal{Q}} c_e(k+1) \sum_{j \in \mathcal{Q}:\, e \in P'_j} \frac{1}{u_e} \\ &\leq 2 \sum_e c_e(k+1) \end{aligned}$$

Competitive ratio

- Off-line reward is at most

$$\begin{aligned}\sum_{j\in\mathcal{Q}}\rho_j+\sum_{j\in\mathcal{A}}\rho_j &\leq 2\sum_e c_e(k+1)+\sum_{j\in\mathcal{A}}\rho_j\\ &\leq 2+(2\log\mu+1)\sum_{j\in\mathcal{A}}\rho_j\end{aligned}$$

- Competitive ratio is $2\log\mu+3$

Conclusions and further work

- Online algorithm for sequentially phasing in multiple unicast connections
- Performance guaranteed to be within $2 \log \mu + 3$ factor of optimal off-line solution
- Further work:
  - comparison of online network coding algorithm with online routing on typical or random graphs
  - randomized online network coding algorithms
  - online algorithms for networks with concurrent multicast and unicast connections