Online Contention Resolution Schemes

Moran Feldman
Open University of Israel

Ola Svensson
EPFL

Rico Zenklusen
ETH Zurich

Online rounding of relaxations

Our focus in this talk: Online selection problems (and related problems).

An "online" relaxation-and-rounding approach

- Solve offline relaxation → fractional sol x^*.
- Use x^* to design online ALG.
- Compare expected value of ALG against value of x^*.

Can be interpreted as rounding x^* online.

Above approach was used in various online selection problems, e.g.:

- Stochastic matching ([Bansal, Gupta, Li, Mestre, Nagarajan, Rudra 2012]).
- Bayesian Mechanism Design ([Yan 2010], [Chawla, Malec, Hartline, Sivan 2010]).
- Stochastic Probing ([Gupta and Nagarajan 2013]).

Online contention resolution schemes (OCRSs)

Online rounding framework based on contention resolution schemes (CRSs).

Some remarks related to CRSs

- CRSs were introduced for constrained submodular function max. (SFM).
  [Chekuri, Vondrák, Z. 2010]
- Though SFM is not our focus here, we inherit many properties of CRSs (can combine constraints; online submodular maximization).

In this talk

Online contention resolution schemes (OCRSs)

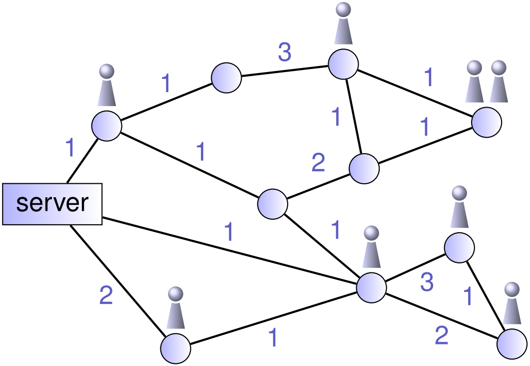
Online rounding framework based on contention resolution schemes (CRSs).

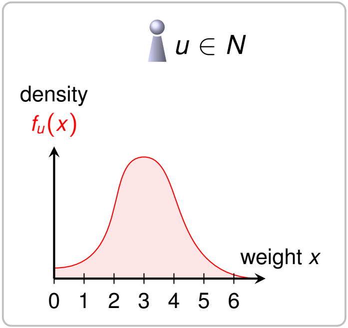
Some remarks related to CRSs

- CRSs were introduced for constrained submodular function max. (SFM).
  [Chekuri, Vondrák, Z. 2010]
- Though SFM is not our focus here, we inherit many properties of CRSs (can combine constraints; online submodular maximization).

To introduce OCRSs, let's start with an example problem.

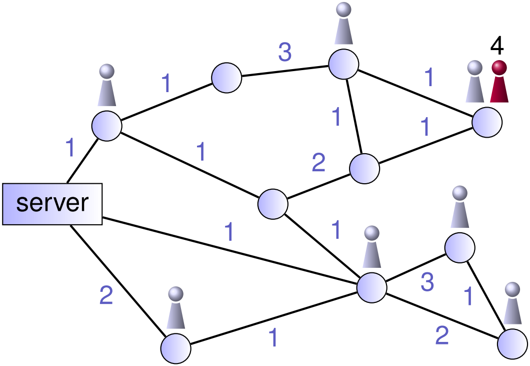
Example Problem: Bayesian Online Selection

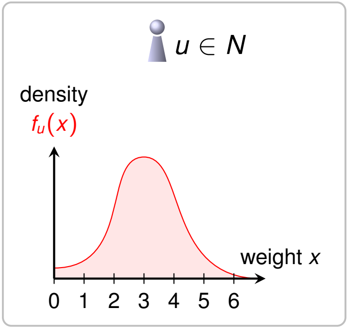




- reveal weights/bids in unknown order and want connection to server.
- Each offer must be accepted/rejected immediately.
- Known info: (indep.) weight distribution for each .

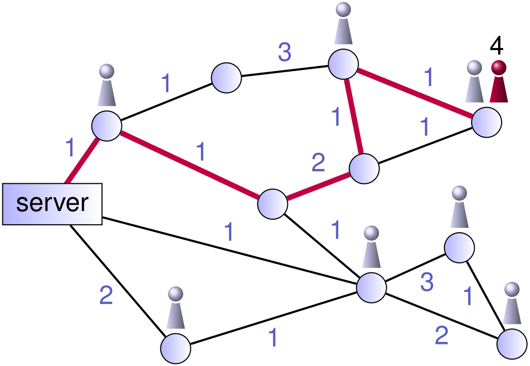
Goal: Maximize expected selected weight.

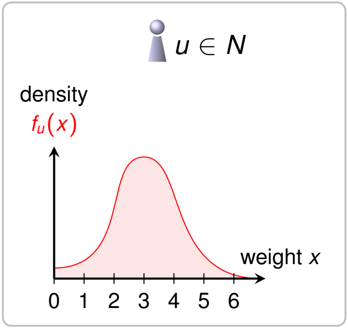




- reveal weights/bids in unknown order and want connection to server.
- Each offer must be accepted/rejected immediately.
- Known info: (indep.) weight distribution for each .

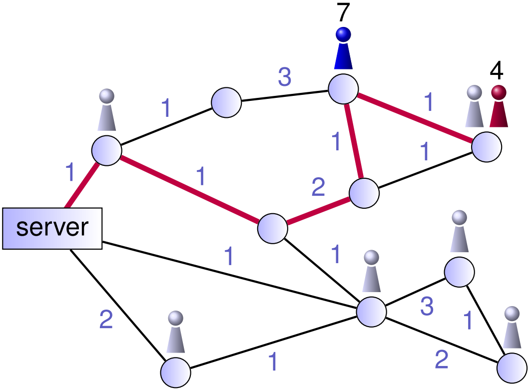
Goal: Maximize expected selected weight.

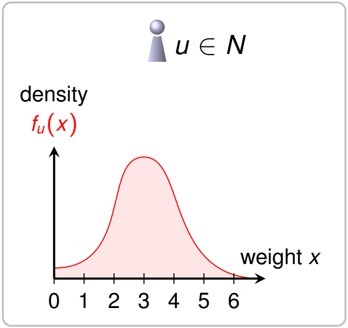




- reveal weights/bids in unknown order and want connection to server.
- Each offer must be accepted/rejected immediately.
- Known info: (indep.) weight distribution for each .

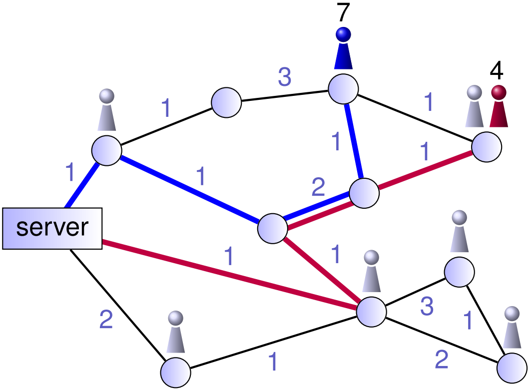
Goal: Maximize expected selected weight.

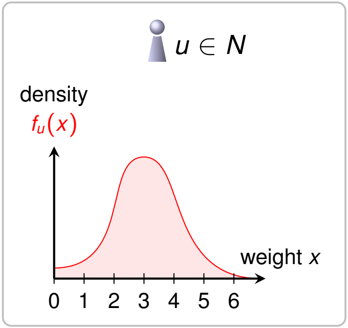




- reveal weights/bids in unknown order and want connection to server.
- Each offer must be accepted/rejected immediately.
- Known info: (indep.) weight distribution for each .

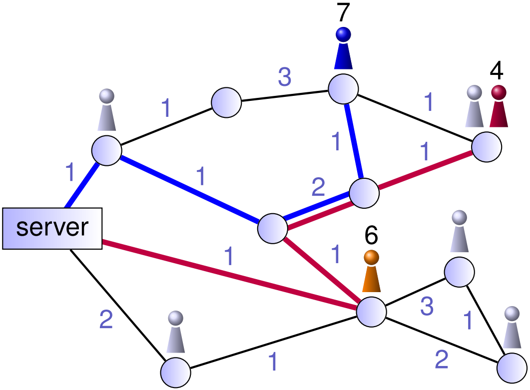
Goal: Maximize expected selected weight.

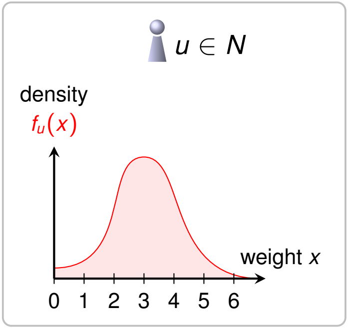




- reveal weights/bids in unknown order and want connection to server.
- Each offer must be accepted/rejected immediately.
- Known info: (indep.) weight distribution for each .

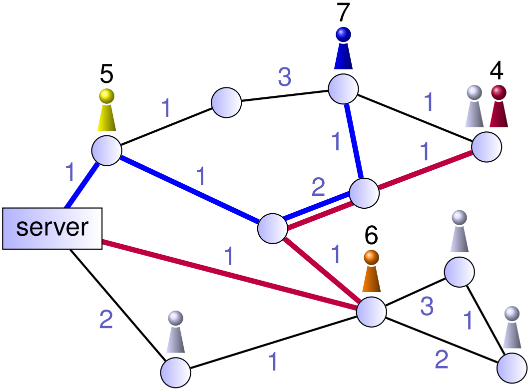
Goal: Maximize expected selected weight.

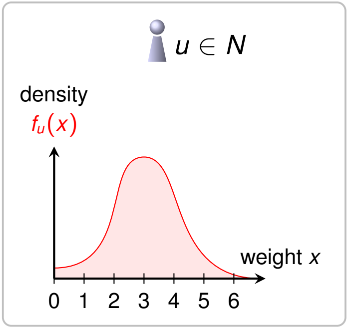




- reveal weights/bids in unknown order and want connection to server.
- Each offer must be accepted/rejected immediately.
- Known info: (indep.) weight distribution for each .

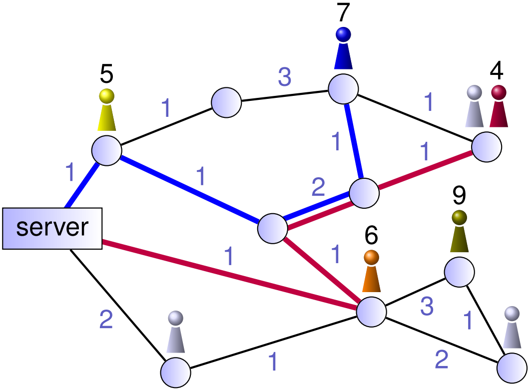
Goal: Maximize expected selected weight.

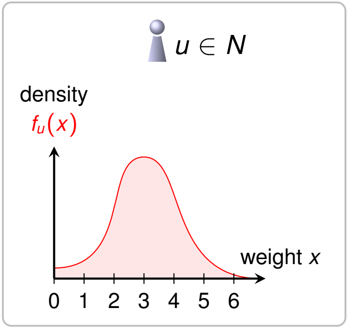




- reveal weights/bids in unknown order and want connection to server.
- Each offer must be accepted/rejected immediately.
- Known info: (indep.) weight distribution for each .

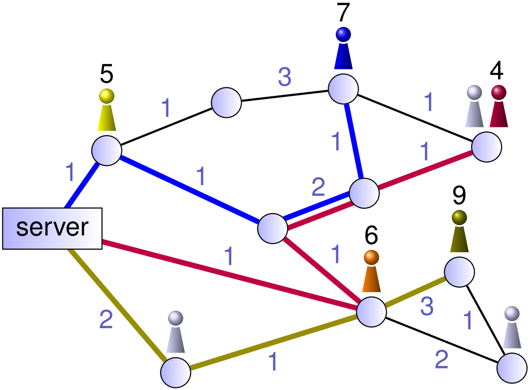
Goal: Maximize expected selected weight.

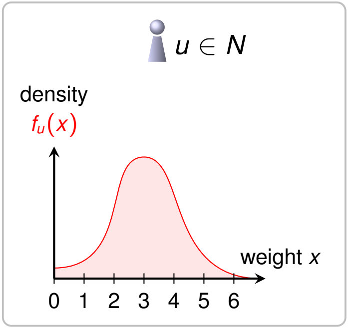




- reveal weights/bids in unknown order and want connection to server.
- Each offer must be accepted/rejected immediately.
- Known info: (indep.) weight distribution for each .

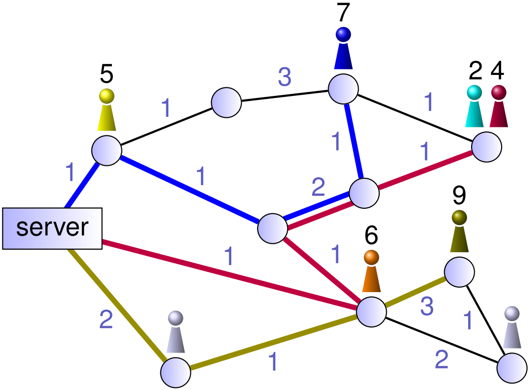
Goal: Maximize expected selected weight.

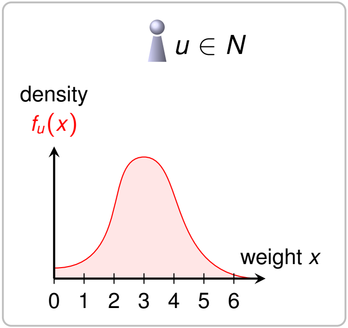




- reveal weights/bids in unknown order and want connection to server.
- Each offer must be accepted/rejected immediately.
- Known info: (indep.) weight distribution for each .

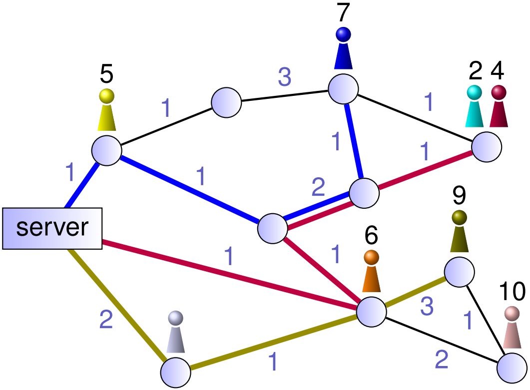
Goal: Maximize expected selected weight.

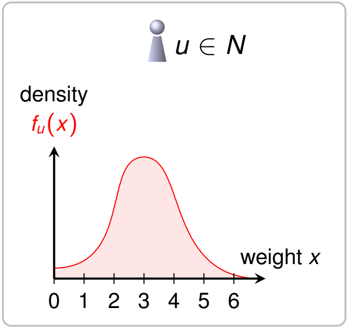




- reveal weights/bids in unknown order and want connection to server.
- Each offer must be accepted/rejected immediately.
- Known info: (indep.) weight distribution for each .

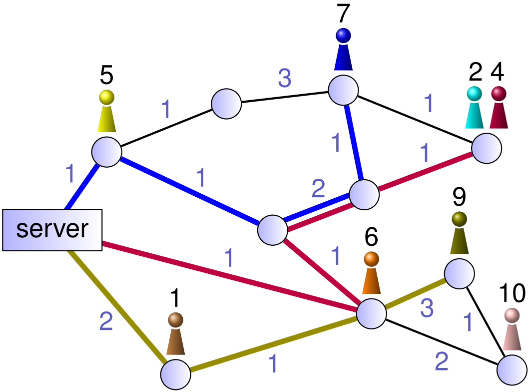
Goal: Maximize expected selected weight.

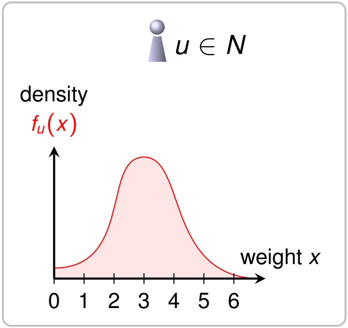




- ▶ [pawn icon] reveal weights/bids in unknown order and want connection to server.
- ▶ Each offer must be accepted/rejected immediately.
- ▶ Known info: (indep.) weight distribution for each [pawn icon].

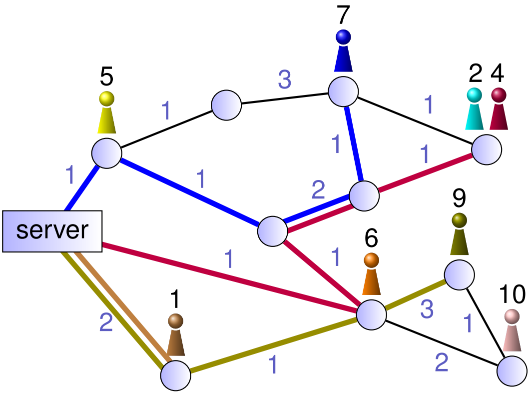
Goal: Maximize expected selected weight.

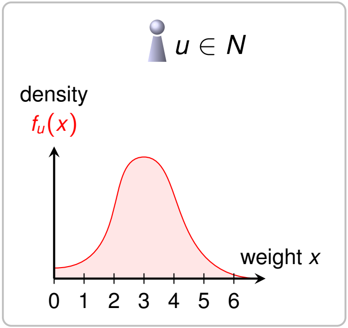




- reveal weights/bids in unknown order and want connection to server.
- Each offer must be accepted/rejected immediately.
- Known info: (indep.) weight distribution for each .

Goal: Maximize expected selected weight.

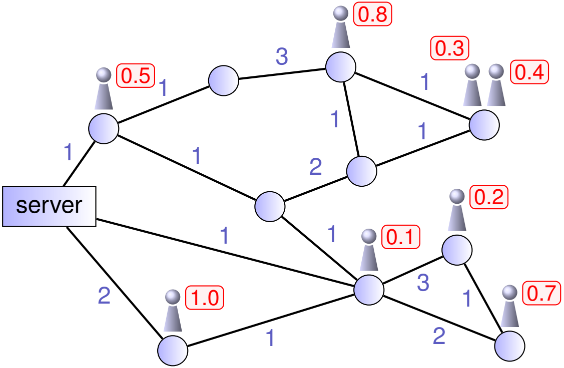


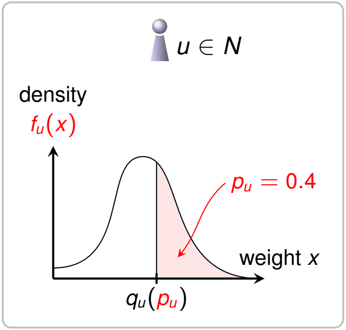


- reveal weights/bids in unknown order and want connection to server.
- Each offer must be accepted/rejected immediately.
- Known info: (indep.) weight distribution for each .

Goal: Maximize expected selected weight.

Toward an offline relaxation

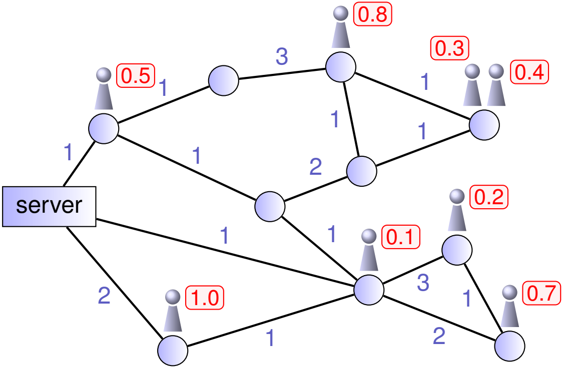


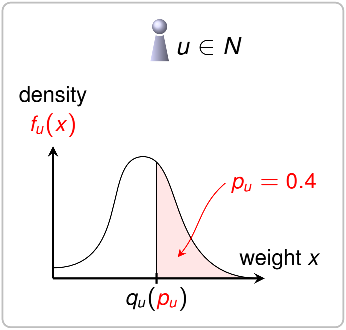


- $p(u) \in [0, 1]$: probability with which (opt.) algo selects $u \in N$.
- Any algo with acceptance prob. p selects expected weight of at most:

$$\phi(p) := \sum_{u \in N} \int_{q_u(p_u)}^{\infty} x \cdot f_u(x) dx.$$

Toward an offline relaxation



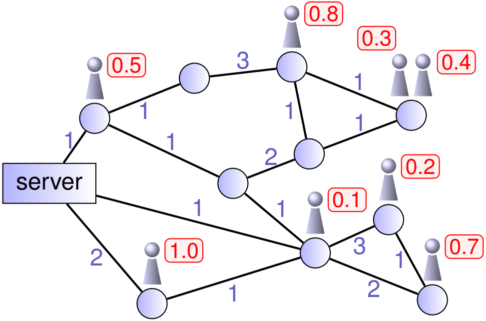


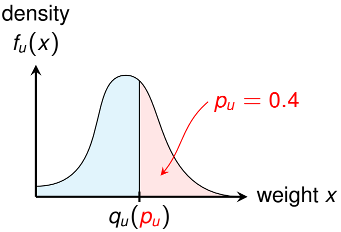
- $p(u) \in [0, 1]$: probability with which (opt.) algo selects $u \in N$.
- Any algo with acceptance prob. p selects expected weight of at most:

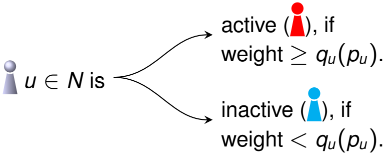
$$\phi(p) := \sum_{u \in N} \int_{q_u(p_u)}^{\infty} x \cdot f_u(x) dx.$$

concave in p

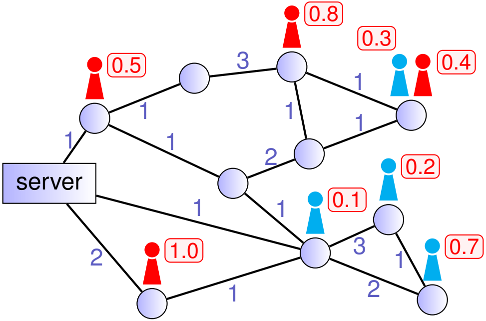
Interpretation of ϕ

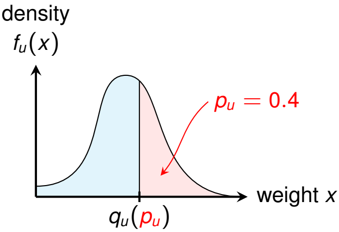






- $\phi(p)$: expected weight, when accepting all active (ignoring constraints).





$u \in N$ is
- active, if weight $\geq q_u(p_u)$.
- inactive, if weight $< q_u(p_u)$.

- $\phi(p)$: expected weight, when accepting all active (ignoring constraints).

- Let $\mathcal{F} \subseteq 2^N$ be feasible sets: $S \subseteq N$ s.t. S is connectable to server.

  ($\mathcal{F}$ are indep. sets of a matroid.)

- $Q = \text{conv}(\{\chi^S \mid S \in \mathcal{F}\}) \subseteq [0,1]^N$. ($Q$ is a matroid polytope.)

Convex relaxation

$$\max\{\phi(p) \mid p \in Q\}$$

Relaxation

- Let $\mathcal{F} \subseteq 2^N$ be feasible sets: $S \subseteq N$ s.t. S is connectable to server.
  ($\mathcal{F}$ are indep. sets of a matroid.)
- $Q = \text{conv}(\{\chi^S \mid S \in \mathcal{F}\}) \subseteq [0,1]^N$. ($Q$ is a matroid polytope.)

Convex relaxation

$$\max\{\phi(p) \mid p \in Q\}$$

- Let p^* be optimal sol to relaxation.
- Using p^*, we define active ( ) and inactive ( ):
  Active elems $A \subseteq N$ are distributed like $R(p^*)$: $\Pr[u \in R(p^*)] = p^*_u$ indep. for $u \in N$.

Relaxation

- Let $\mathcal{F} \subseteq 2^N$ be feasible sets: $S \subseteq N$ s.t. S is connectable to server.
  ($\mathcal{F}$ are indep. sets of a matroid.)
- $Q = \text{conv}(\{\chi^S \mid S \in \mathcal{F}\}) \subseteq [0,1]^N$. ($Q$ is a matroid polytope.)

Convex relaxation

$$\max\{\phi(p) \mid p \in Q\}$$

- Let p^* be optimal sol to relaxation.
- Using p^*, we define active ( ) and inactive ( ):
  Active elems $A \subseteq N$ are distributed like $R(p^*)$: $\Pr[u \in R(p^*)] = p^*_u$ indep. for $u \in N$.

How to decide online which elems of A to select, to be able to compare against $\phi(p^*)$?

(Greedy) Online Contention Resolution Schemes

Our goal

No matter in what order elems appear, select $I \subseteq A$, $I \in \mathcal{F}$ satisfying

$$\Pr[u \in I \mid u \in A] \geq c \quad \forall u \in N,$$

for some constant $c \in (0, 1]$.

Immediately implies c-competitive algo for discussed selection problem.

(Greedy) Online Contention Resolution Schemes

Our goal

No matter in what order elems appear, select $I \subseteq A$, $I \in \mathcal{F}$ satisfying

$$\Pr[u \in I \mid u \in A] \geq c \quad \forall u \in N,$$

for some constant $c \in (0, 1]$.

Immediately implies c-competitive algo for discussed selection problem.

Greedy OCRS

A *greedy OCRS* for a relaxation Q of $\mathcal{F}$, is an online selection algo defined as follows: For each $p \in Q$, there is $\mathcal{F}_p \subseteq 2^N$ s.t.

(i) $\mathcal{F}_p \subseteq \mathcal{F}$, and

(ii) each $u \in A$ is accepted if selected elems remain $\in \mathcal{F}_p$.

A greedy OCRS is *randomized* if choice of $\mathcal{F}_p$ is randomized.

Toward our goal via c-selectability

c-selectability of greedy OCRSs

A greedy OCRS is c-selectable if $\forall u \in N$, with prob. $\geq c$:

u can be selected if active even

- if u appears as last element, and
- no matter what active elems have been accepted previously.

Formally, a greedy OCRS is c-selectable if for any $p \in Q$:

$$\Pr[I \cup \{u\} \in \mathcal{F}_p \quad \forall I \subseteq R(p), I \in \mathcal{F}_p] \geq c \quad \forall u \in N.$$

(Some) results on greedy OCRSs

$\Omega(1)$-selectable (rand.) greedy OCRSs exist for interesting constraints

- Matroids: 0.25-selectable.
- Matching: $\frac{1}{2}e^{-1} \approx 0.18$-selectable.
- Knapsack: $1 - \frac{1}{\sqrt{2}} \approx 0.3$-selectable.

Some additional properties

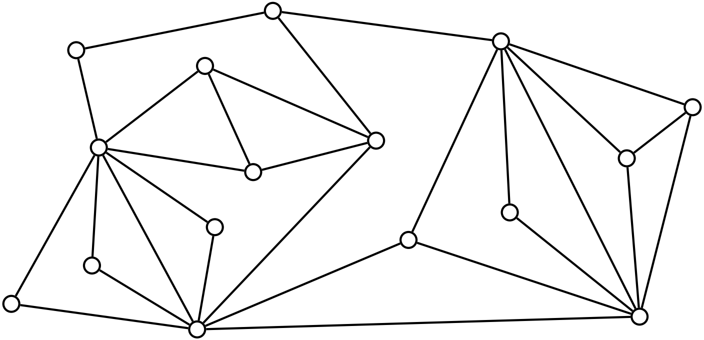
- Combining constraints:
  $O(1)$-selectable OCRSs for $Q^1, Q^2 \longrightarrow O(1)$-sel. OCRS for $Q^1 \cap Q^2$.
- OCRSs can be used for online submodular function maximization (using multilinear extension).

Example:

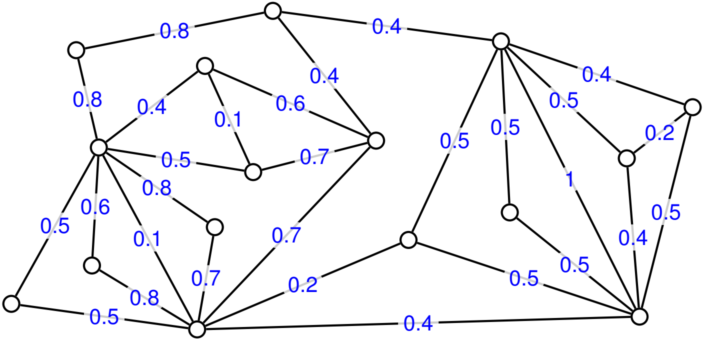
Radomized 0.25-selectable OCRS

for forest (matroid) polytope

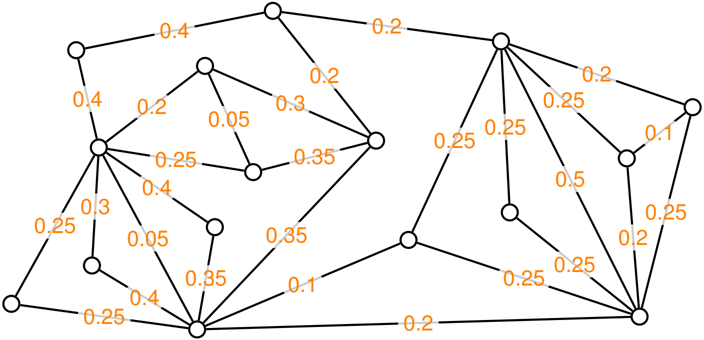
- $\mathcal{F} \subseteq 2^E$: all forests in a graph $G = (V, E)$.
- Let x be in forest polytope → active elems: $R(x)$.
- Discard each edge with prob 0.5 → active non-discarded sets: $R(y)$, with $y = \frac{1}{2}x$.
  (This “scaling-down” can be done with randomized greedy OCRS.)



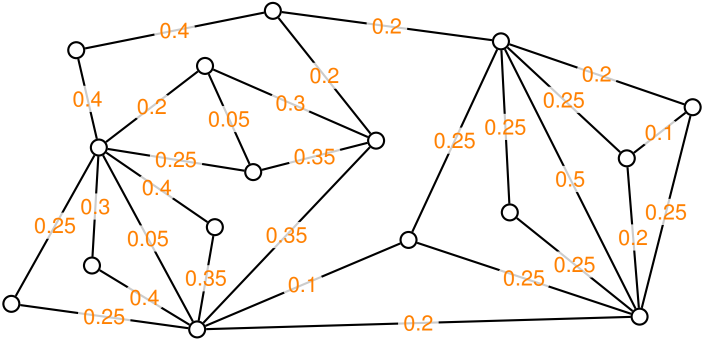
- $\mathcal{F} \subseteq 2^E$: all forests in a graph $G = (V, E)$.
- Let x be in forest polytope ⟶ active elems: $R(x)$.
- Discard each edge with prob 0.5 ⟶ active non-discarded sets: $R(y)$, with $y = \frac{1}{2}x$.
  (This "scaling-down" can be done with randomized greedy OCRS.)



- $\mathcal{F} \subseteq 2^E$: all forests in a graph $G = (V, E)$.
- Let x be in forest polytope ⟶ active elems: $R(x)$.
- Discard each edge with prob 0.5 ⟶ active non-discarded sets: $R(y)$, with $y = \frac{1}{2}x$.
  (This “scaling-down” can be done with randomized greedy OCRS.)

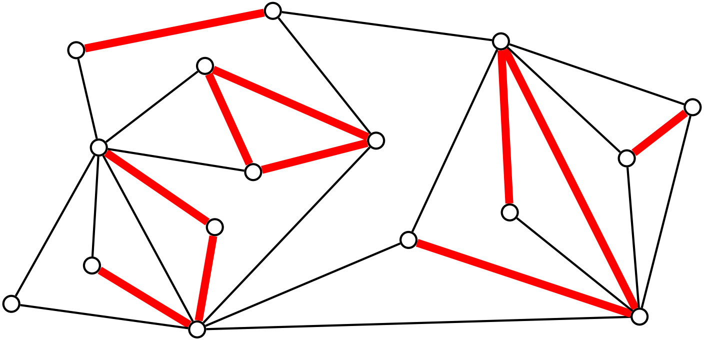


- $\mathcal{F} \subseteq 2^E$: all forests in a graph $G = (V, E)$.
- Let x be in forest polytope $\longrightarrow$ active elems: $R(x)$.
- Discard each edge with prob 0.5 $\longrightarrow$ active non-discarded sets: $R(y)$, with $y = \frac{1}{2}x$.
  (This "scaling-down" can be done with randomized greedy OCRS.)



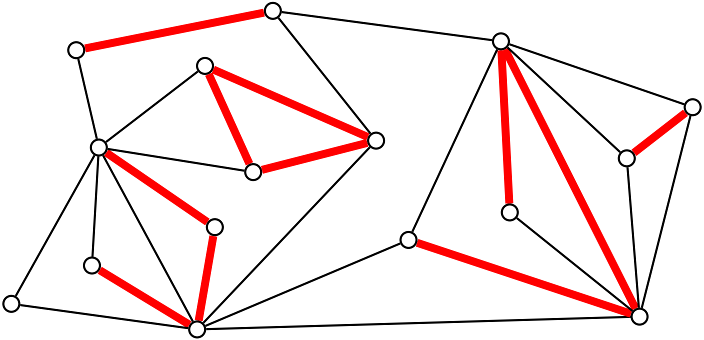
- We are interested in prob. of edges $e \in E$ being spanned by $R(y) \setminus \{e\}$.

- $\mathcal{F} \subseteq 2^E$: all forests in a graph $G = (V, E)$.
- Let x be in forest polytope $\longrightarrow$ active elems: $R(x)$.
- Discard each edge with prob 0.5 $\longrightarrow$ active non-discarded sets: $R(y)$, with $y = \frac{1}{2}x$.
  (This "scaling-down" can be done with randomized greedy OCRS.)



- We are interested in prob. of edges $e \in E$ being spanned by $R(y) \setminus \{e\}$.

- $\mathcal{F} \subseteq 2^E$: all forests in a graph $G = (V, E)$.
- Let x be in forest polytope $\longrightarrow$ active elems: $R(x)$.
- Discard each edge with prob 0.5 $\longrightarrow$ active non-discarded sets: $R(y)$, with $y = \frac{1}{2}x$.
  (This "scaling-down" can be done with randomized greedy OCRS.)



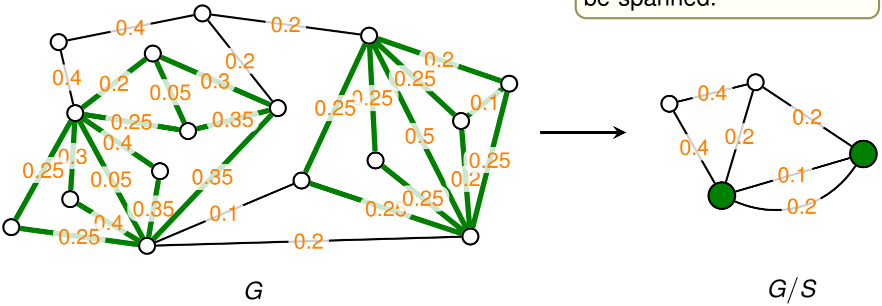
- We are interested in prob. of edges $e \in E$ being spanned by $R(y) \setminus \{e\}$.
- Edges likely to be spanned have to be "protected".
  $\longrightarrow$ They may appear last and should still be selectable with constant prob.

Theorem

There exists a set $S \subsetneq E$ such that:

$$\Pr[e \in \text{span}((R(y) \cup S) \setminus \{e\})] \leq 0.5 \qquad \forall\, e \in E \setminus S.$$

In words: After contracting S, no element is likely to be spanned.

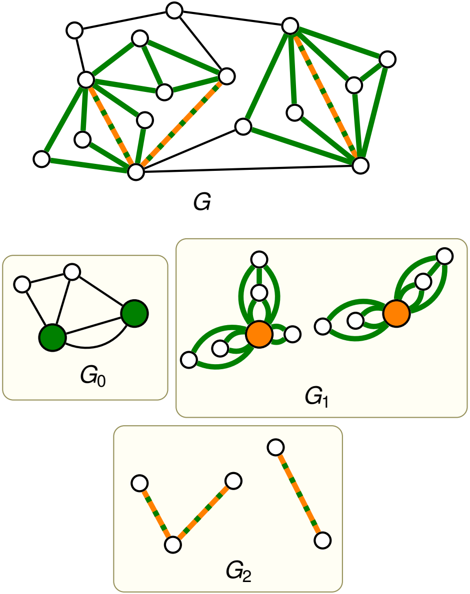


We apply above theorem recursively (to S) to obtain chain.

Greedy OCRS for forest (matroid) polytope

- Use theorem recursively to get: $E = S_0 \supsetneq S_1 \supsetneq S_2 \supsetneq S_3 = \emptyset$.
- $G_i = (V, S_i)/S_{i+1}$.
- $\mathcal{F}_y$: All $U \subseteq E$ s.t. $U \cap (S_i \setminus S_{i+1})$ is forest in G_i.
- $\Rightarrow \Pr[e \text{ selectable}] \geq 0.5 \; \forall e \in E$
- Another factor 2 lost from x to $y = \frac{1}{2}x$.

$\Rightarrow$ 0.25-selectable OCRS.



OCRSs

- New rounding strategy for online selection problems.
- Works for many interesting constraint types (e.g., matroids, matchings, knapsacks).

Some features (largely inherited from CRSs)

- Constraints can be combined.
- Can be used for submodular maximization.

Solves open problems in literature:

- Existence of $O(1)$-factor oblivious posted price mechanism for matroids.
  [Chawla, Hartline, Malec, Sivan 2010]
- Existence of $O(1)$-algo for weighted stochastic probing with deadlines.
  [Gupta and Nagarajan 2013]

References I

Bansal, N., Gupta, A., Li, J., Mestre, J., Nagarajan, V., and Rudra, A. (2012). When LP is the cure for your matching woes: Improved bounds for stochastic matchings. *Algorithmica*, 63(4):733–762.

Chawla, S., Hartline, J., Malec, D., and Sivan, B. (2010a). Multi-parameter mechanism design and sequential posted pricing. http://arxiv.org/abs/0907.2435v2.

Chawla, S., Hartline, J. D., Malec, D. L., and Sivan, B. (2010b). Multi-parameter mechanism design and sequential posted pricing. In *STOC '10: Proceedings of the 42nd ACM symposium on Theory of computing*, pages 311–320, New York, NY, USA. ACM.

Gupta, A. and Nagarajan, V. (2013). A stochastic probing problem with applications. In *Integer Programming and Combinatorial Optimization (IPCO)*, pages 205–216.

Yan, Q. (2011). Mechanism design via correlation gap. In *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 710–719. SIAM.