Online Algos: Worst case and Beyond

Introduction and Set Cover

Anupam Gupta (NYU)

NYU | COURANT

Online Algorithms

Requests arrive over time, must be served immediately/irrevocably

Goal: (say) minimize cost of the decisions taken

Competitive ratio of algorithm A:

worst case! →

$$\max_{\text{instances } I} \frac{\text{cost of algorithm } A \text{ on instance } I}{\text{optimal cost to serve } I}$$

Want to minimize the competitive ratio.

[Graham 66, Sleator Tarjan 85]

Online Set Cover

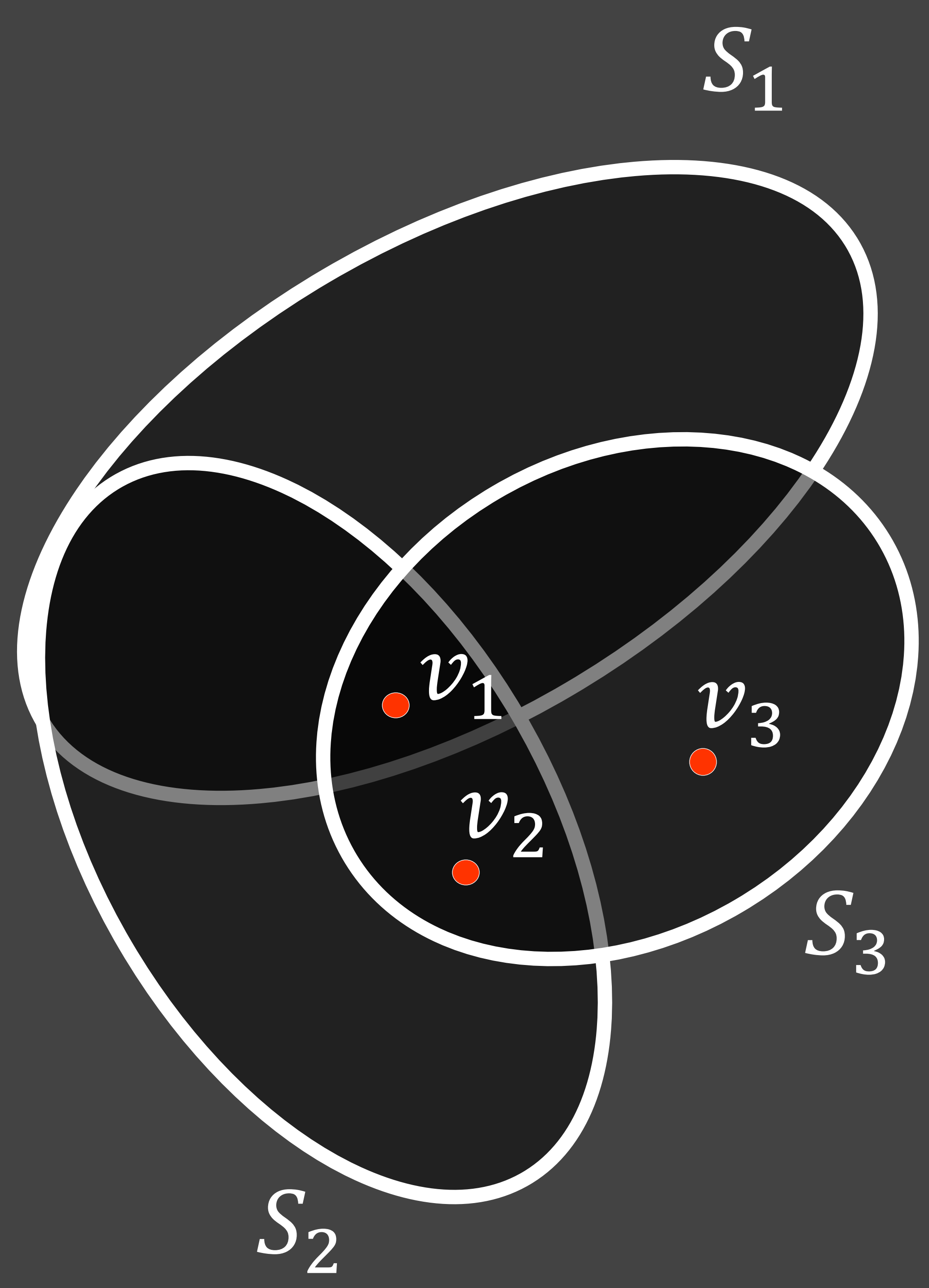
Set system. n elements arrive over time, want to maintain a cover.

Goal: minimize cost of sets picked

Competitive ratio of algorithm A:

$$\max_{\text{instances } I} \frac{\text{cost of algorithm } A \text{ on instance } I}{\text{optimal cost to serve } I}$$

Want to minimize the competitive ratio.

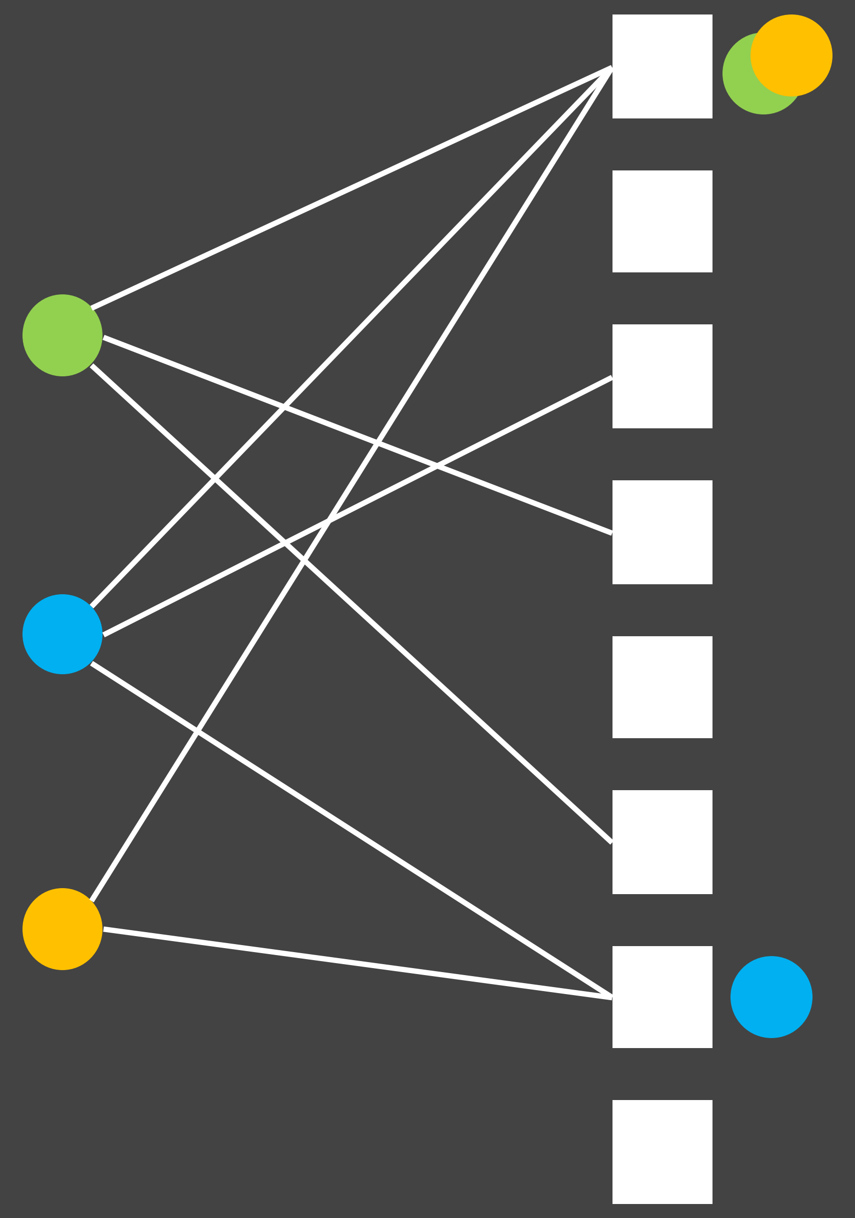


[Alon Awerbuch Azar Buchbinder Naor 03]

Online Load Balancing

m machines. Jobs arrive over time, can be assigned to subset of machines.

Goal: minimize maximum load of machines



Competitive ratio of algorithm A:

$$\max_{\text{instances } I} \frac{\text{cost of algorithm } A \text{ on instance } I}{\text{optimal cost to serve } I}$$

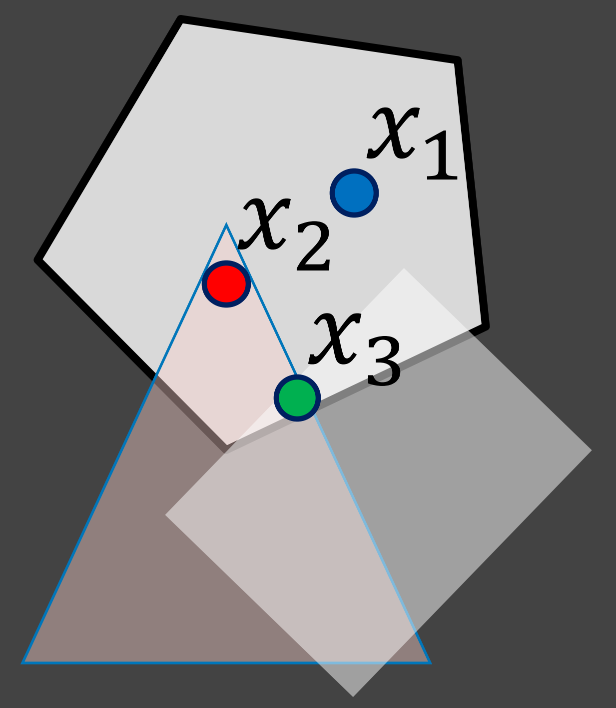
Want to minimize the competitive ratio.

[Azar Naor Rom 92]

Online Convex Body Chasing

Each time convex body K_t appears. Must output $x_t \in K_t$

Goal: minimize $\sum_t |x_t - x_{t-1}|$.



Competitive ratio of algorithm A:

$$\max_{\text{instances } I} \frac{\text{cost of algorithm } A \text{ on instance } I}{\text{optimal cost to serve } I}$$

Want to minimize the competitive ratio.

[Freedman Linial 94]

max-K finding

n people arrive over time, each has value v_i -- can pick at most K

Goal: (say) maximize sum of values of picked people

Competitive ratio of algorithm A:

$$\min_{\text{instances } I} \frac{\text{value of algorithm } A \text{ on instance } I}{\text{optimal value on instance } I}$$

Want to maximize the competitive ratio.

[Gardner 60, Dynkin 63]

goals for this week

What are the results?

What are the techniques?

What are the worst-case limitations?

How to bypass them beyond worst-case?

Connections to other sequential decision-making models/algos?

lecture plan

Lecture #1: Set Cover (worst case)

Lecture #2: Set Cover (beyond worst case), Network design (both)

Lecture #3: Resource Allocation (aka packing)

Lecture #4: Search Problems (aka chasing)

Two Online Models

competitive analysis vs regret minimization

optimization in the face of uncertainty

"Reactive" settings	"Predictive" settings
See request, take action	Predict next step, then see reality

Compare to the best solution in hindsight

Obj: Competitive ratio	Obj: Regret
CR = ALG/OPT	regret = ALG - OPT
Typically OPT = best dynamic solution	Typically OPT = best static solution

several techniques in common…

two canonical online problems

online linear optimization (OLO)

each day t = 1, 2, …, T

algorithm plays probability vector $p_t \in \Delta_n$

then sees cost vector $c_t \in [0,1]^n$

loss at time t is $\ell_t = \langle c_t | p_t \rangle$

Theorem: algorithm that achieves

$$\sum_t \langle c_t | p_t \rangle \leq (1+\varepsilon) \sum_t \langle c_t | p^* \rangle + O\left(\frac{\log n}{\varepsilon}\right)$$

for every $p^* \in \Delta_n$

"smoothed" OLO (aka uniform MTS)

each day t = 1, 2, …, T

sees cost vector $c_t \in [0, \infty)^n$

then algorithm plays probability vector $p_t \in \Delta_n$

loss at time t is $\ell_t = \langle c_t | p_t \rangle + | p_t - p_{t-1} |_1$

Theorem: algorithm that achieves

$$\sum_t \langle c_t | p_t \rangle + |p_t - p_{t-1}| \leq O(\log n) \left(\sum_t \langle c_t | p_t^* \rangle + \sum_t |p_t^* - p_{t-1}^*| \right)$$

for every $p_1^*, p_2^*, \ldots p_T^* \in \Delta_n$

two canonical online problems

online linear optimization (OLO)

each day t = 1, 2, …, T

algorithm plays probability vector $p_t \in \Delta_n$

then sees cost vector $c_t \in [0,1]^n$

loss at time t is $\ell_t = \langle c_t | p_t \rangle$

Theorem: algorithm that achieves

$$\sum_t \langle c_t | p_t \rangle \leq (1+\varepsilon) \sum_t \langle c_t | p^* \rangle + O\left(\frac{\log n}{\varepsilon}\right)$$

for every $p^* \in \Delta_n$

"smoothed" OLO (aka uniform MTS)

each day t = 1, 2, …, T

sees cost vector $c_t \in [0, \infty)^n$

then algorithm plays probability vector $p_t \in \Delta_n$

loss at time t is $\ell_t = \langle c_t | p_t \rangle + | p_t - p_{t-1} |_1$

Theorem: algorithm that achieves

$$\sum_t \langle c_t | p_t \rangle + |p_t - p_{t-1}| \leq (1+\varepsilon) \sum_t \langle c_t | p_t^* \rangle + O\left(\frac{\log n}{\varepsilon}\right) \sum_t |p_t^* - p_{t-1}^*|$$

for every $p_1^*, p_2^*, \ldots p_T^* \in \Delta_n$

[Blum Burch Kalai 99]

online algorithms vs online learning

optimization in the face of uncertainty

"Reactive" settings

See request, take action

"Predictive" settings

Predict next step, then see reality

Compare to the best solution in hindsight

Obj: Competitive ratio

CR = ALG/OPT

Typically OPT = best dynamic solution

Obj: Regret

regret = ALG - OPT

Typically OPT = best static solution

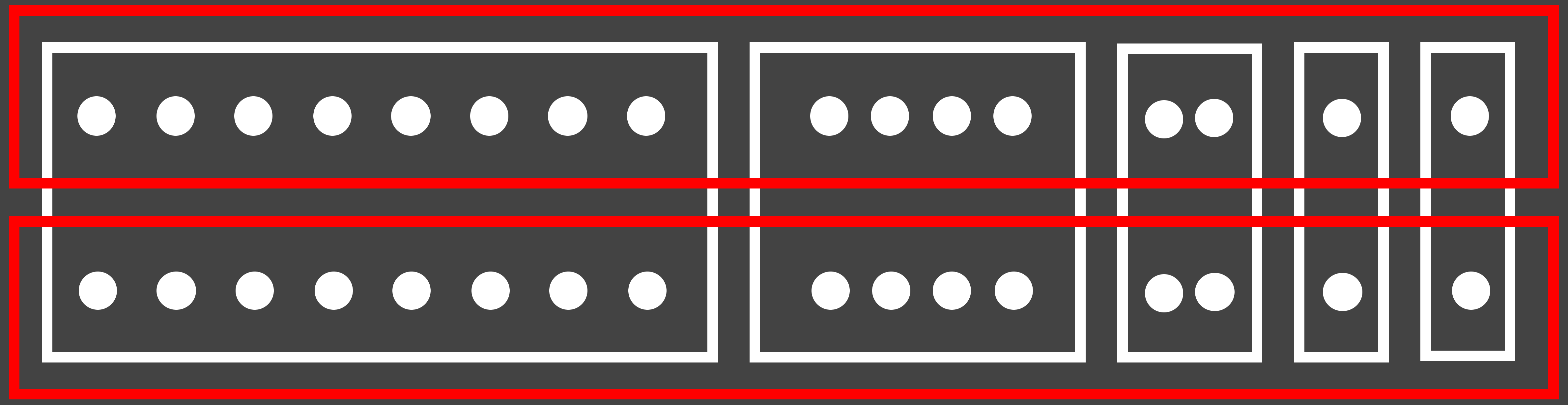
several techniques in common…

Set Cover

(minimum) set cover

$|\mathcal{U}| = n =$# elements
$|\mathcal{F}| = m =$# sets

$f = \max_{e}$ #(sets containing e)



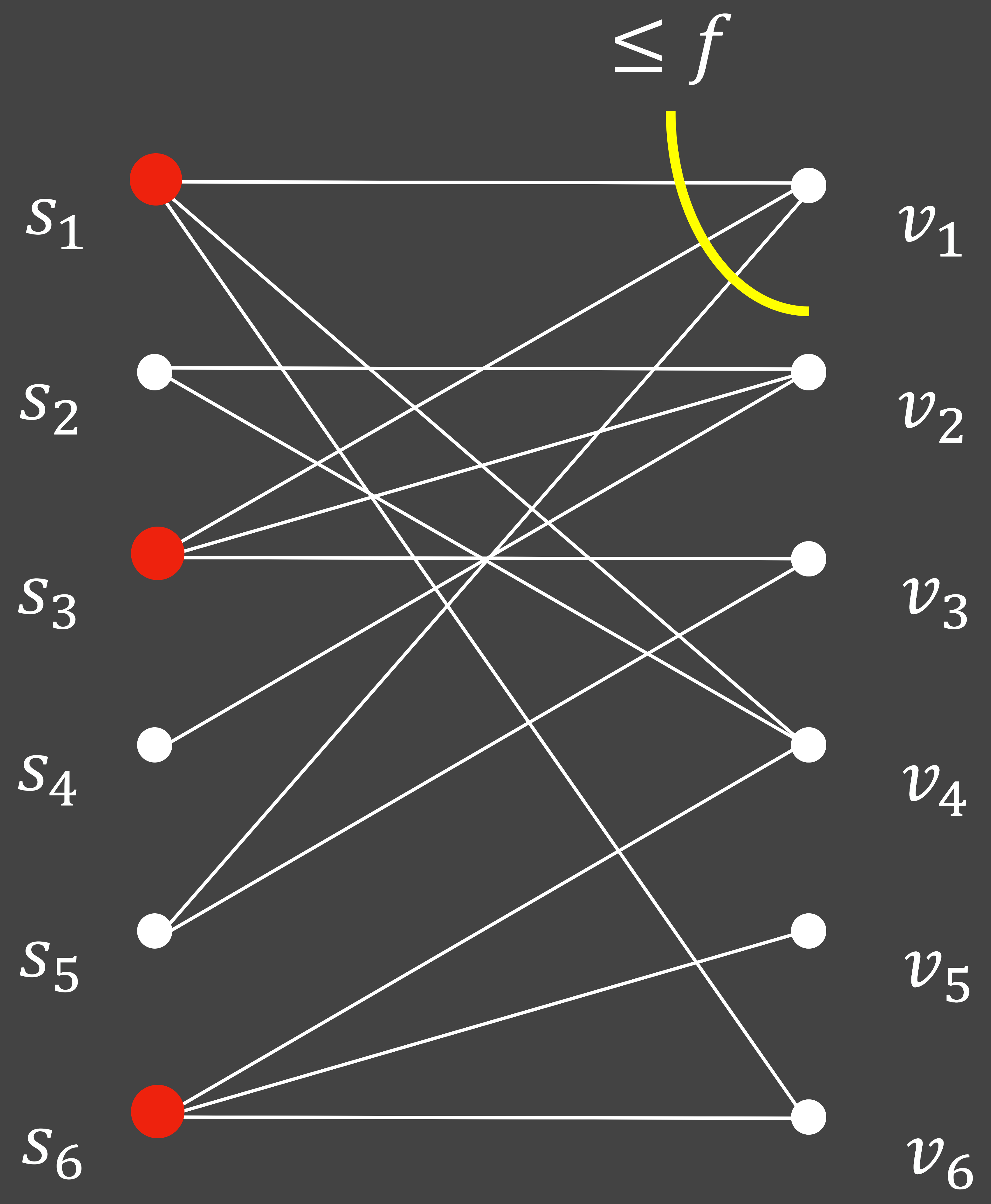
Goal: pick smallest # sets to cover all elements.

"weighted" problem: sets have costs, minimize cost of cover

set cover as bipartite graph

$\mathcal{F}$

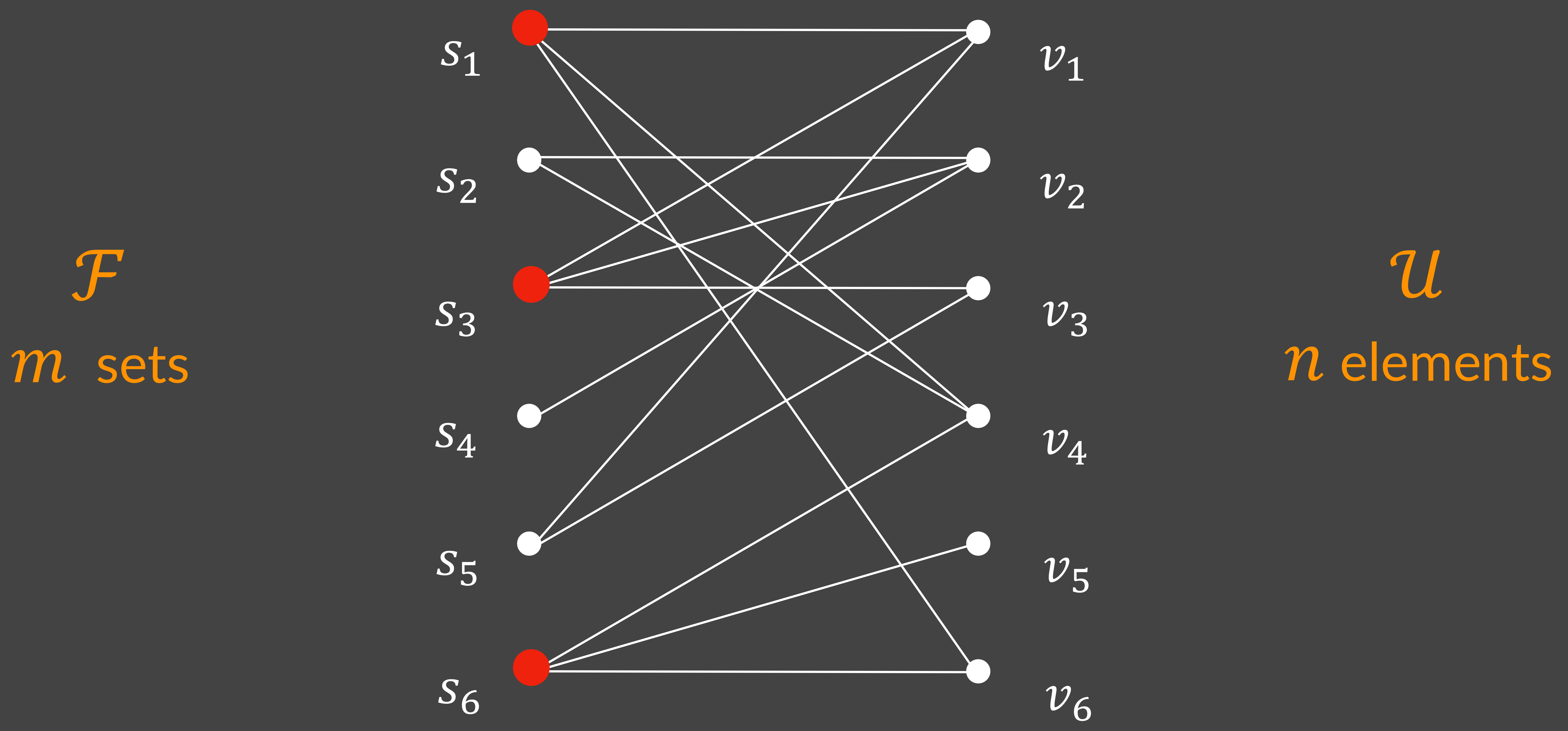
m sets



$\mathcal{U}$

n elements

Online Set Cover



online set cover: model choices

Including order of elements

Adversary fixes $(\mathcal{U}, \mathcal{F})$ and set costs (if applicable)

Algo sees elements of $\mathcal{U}$ one by one

When element e seen, then find out which sets contain e

If e not covered by current set cover, pick set(s) to cover e

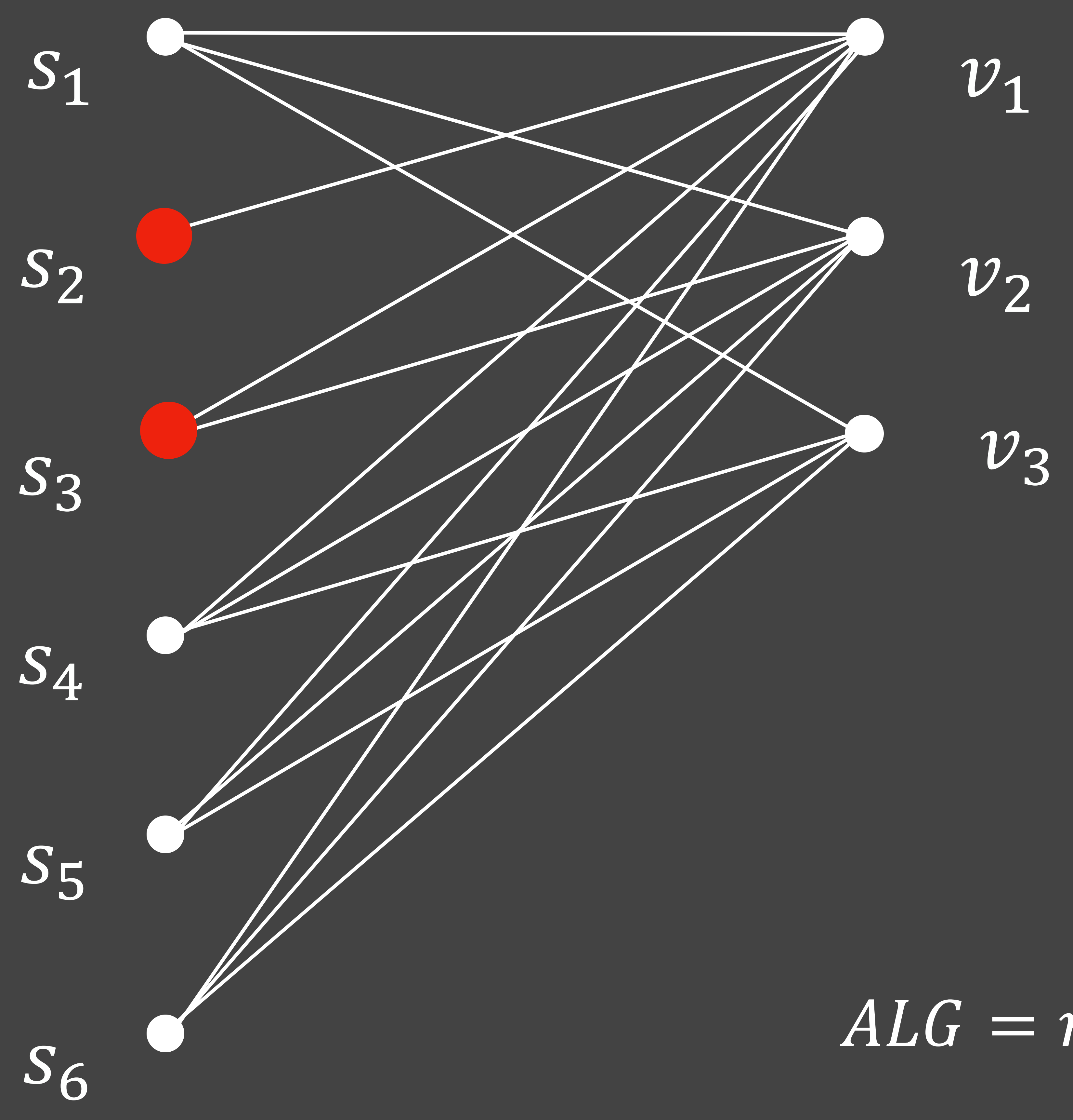
Instance is not "adaptive" to decisions of algorithm (relevant if randomized algo)

Called "oblivious adversary".

Deterministic Fail

$\mathcal{F}$

m sets



$\mathcal{U}$

n elements

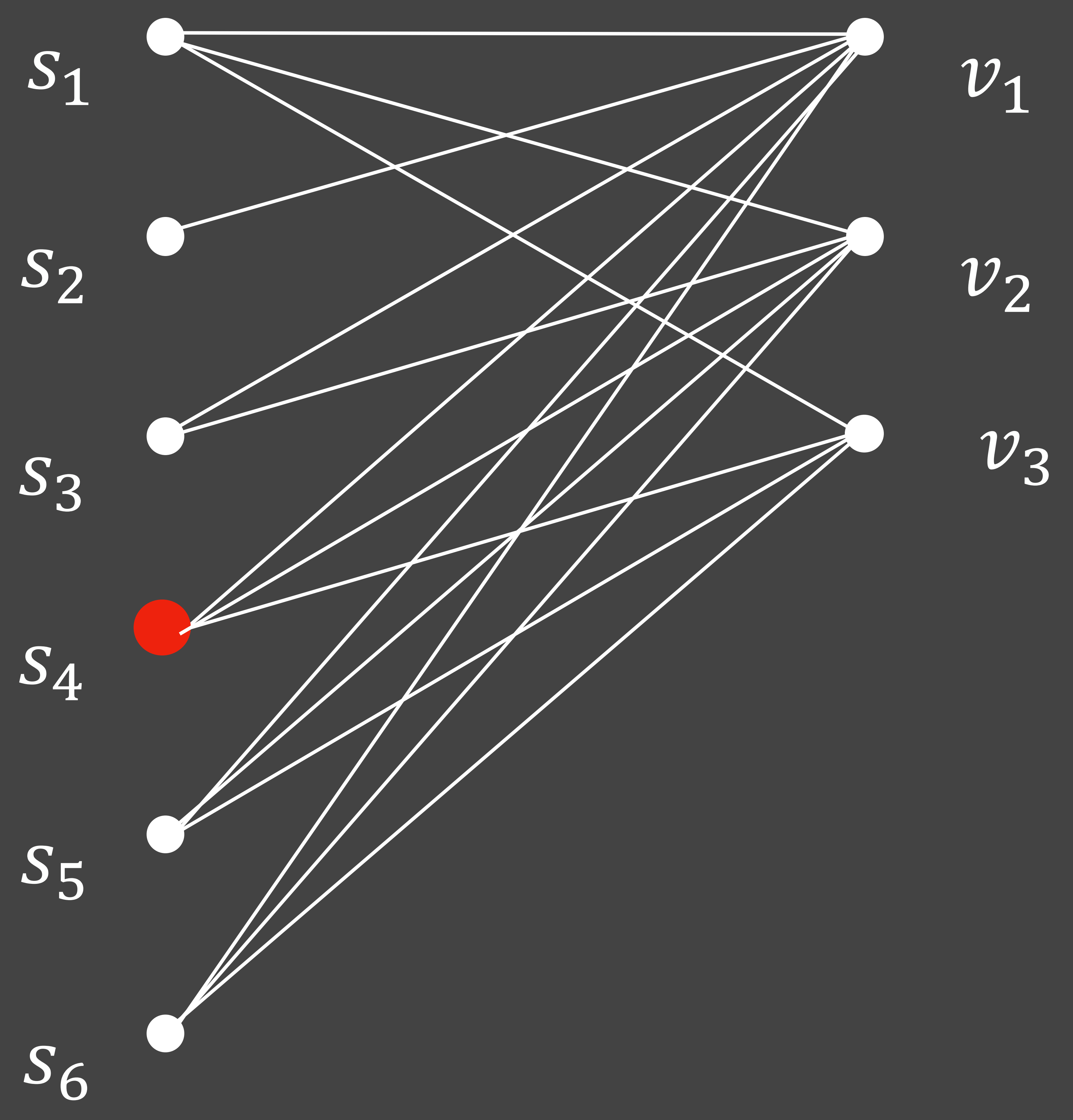
$ALG = n, OPT = 1$

$\Omega(n)$ competitive

Randomization: A New Hope

$\mathcal{F}$

m sets



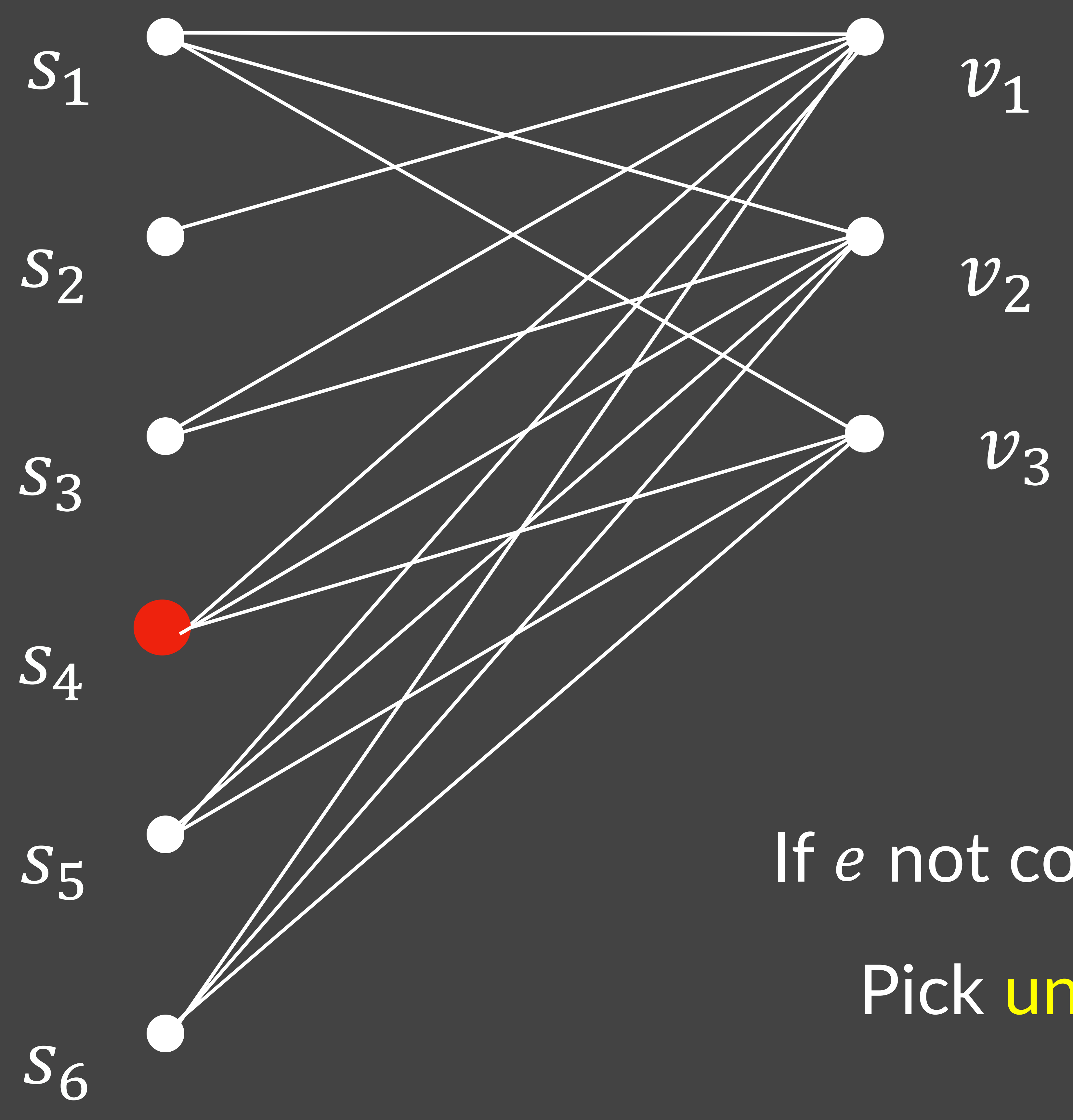
$\mathcal{U}$

n elements

What's a good strategy for randomizing?

Uniform Sampling (for unit costs)

f = max-degree of any element

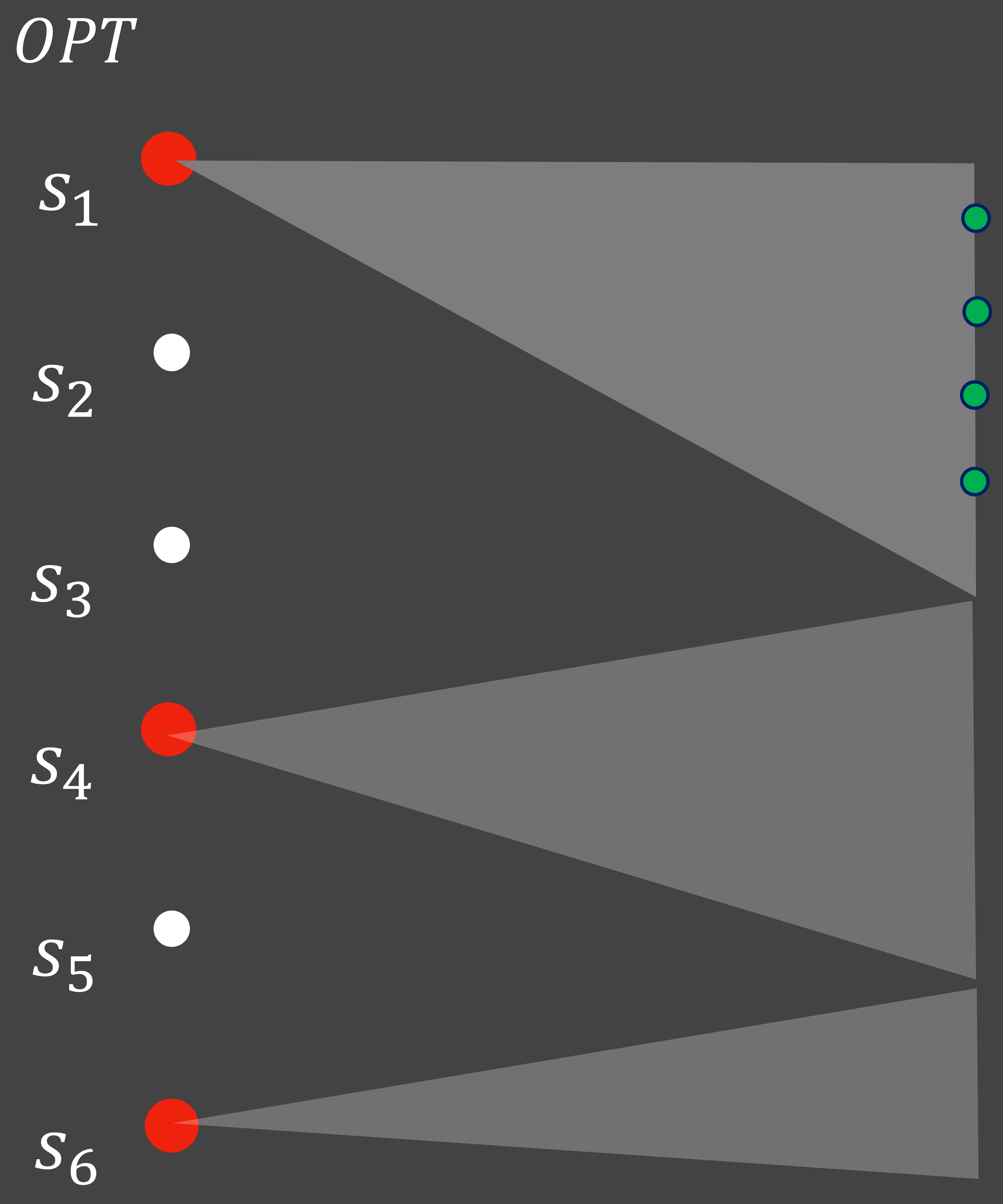


If e not covered

Pick uniformly random set covering e

Theorem [Pitt 75]: f-competitive

Proof



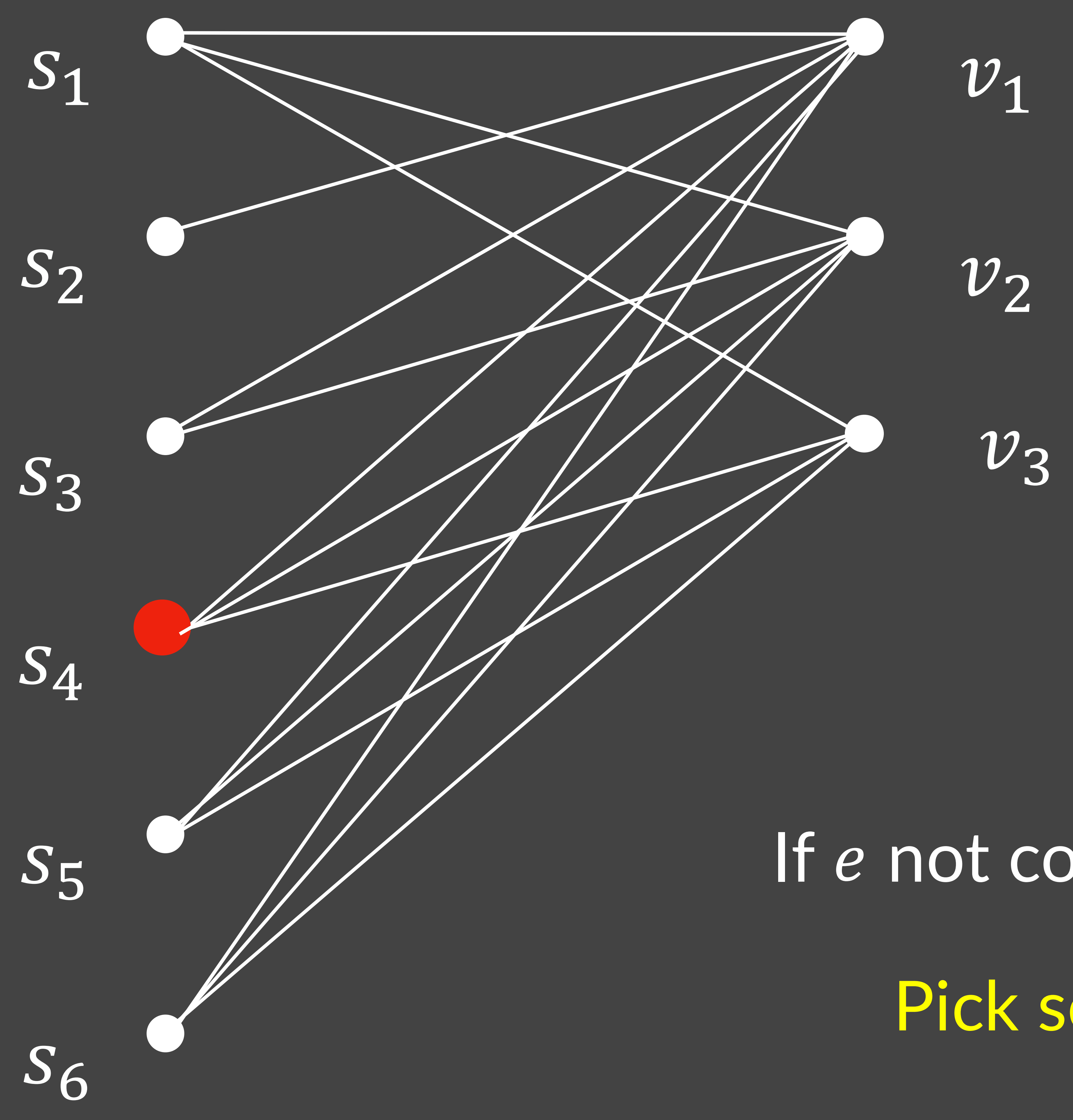
f = max-degree of any element

$\mathbb{E}[Geom(1/f)] = f$

Theorem [Pitt 75]: f-competitive

Extension for general costs

f = max-degree of any element



If e not covered

Pick set $S \ni e$ w.p. $\propto$ $1/c(S)$

Theorem [Pitt 75]: f-competitive

roadmap for today

Intro to Online Algorithms

Set cover

Online algo (using relax-and-round)

Some (almost) matching hardness results

[Prob Wednesday] How to go beyond worst-case?

When requests from known distribution

When requests from **unknown** distribution

Theorem: f-competitive using Pitt's algo

roadmap for today

Intro to Online Algorithms

Set cover

Theorem: f-competitive using Pitt's algo

Online algo (using relax-and-round)

Some (almost) matching hardness results

[Prob Wednesday] How to go beyond worst-case?

When requests from known distribution

When requests from **unknown** distribution

relax-and-round (offline)

~~Integer~~ linear program for set cover: min $\sum_S c(S)\, x_S$

$\sum_{S: e \in S} x_S \geq 1$ for all elements e

~~$x_S \in \{0,1\}$~~ "relax the problem" $LP \leq OPT$

$x_S \geq 0$

[folklore? Johnson?]

relax-and-round (offline)

~~Integer~~ linear program for set cover: min $\sum_S c(S)\, x_S$

$\sum_{S:e\in S} x_S \ \geq\ 1$ for all elements e

~~$x_S \in \{0,1\}$~~ "relax the problem" $LP \leq OPT$

$x_S \geq 0$

Randomized rounding: pick each set with probability $p_S = \min\{x_S \ln n\,, 1\}$

Alteration: For each some element not yet covered, pick cheapest set covering it.

Pr[e not covered by RR] = $\Pi_{S:e\in S}(1-p_S) \leq \mathrm{e}^{-\sum_{S:e\in S} p_S} \leq \mathrm{e}^{-\ln n \sum_{S:e\in S} x_S} \leq 1/n.$

Expected cost $\leq LP \cdot \ln n + \sum_e {}^1/_n \cdot \text{cheapest set covering } e \leq LP(1+\ \ln n).$

[folklore? Johnson?]

relax-and-round (online)

~~Integer~~ linear program for set cover:

min $\sum_S c(S)\, x_S$

$\sum_{S:e\in S} x_S \geq 1$ for all elements e

~~$x_S \in \{0,1\}$~~ "relax the problem" $LP \leq OPT$

$x_S \geq 0$

Solving: How to solve LP as elements arrive? We'll see soon.

N.b. want x_S values to only increase (captures online nature)

Online rounding: at each timestep, buy each set S w.p. $\min(\,\Delta x_S \ln n\,,1\,)$

Same analysis: total cost $O(\ln n)\; LP$.

[Alon Awerbuch Azar Buchbinder Naor 03]

solving set cover LP online

$$\min \sum_S c(S)\, x_S$$

$$\sum_{S:e\in S} x_S \geq 1 \quad \text{for all } e$$

$$x_S \geq 0$$

When element e arrives:

Until e fractionally covered

increase its fractional coverage "multiplicatively"

Theorem 1: get fractional set cover of cost $O(\log m)\ OPT$.

⇒ **Theorem 2:** Above+rand.round produces integer set cover with E[cost] $O(\log m \log n)\ OPT$.

solving set cover LP online

min $\sum_S x_S$

$\sum_{S:e\in S} x_S \geq 1$ for all e

$x_S \geq 0$

Algo 1: (unit cost sets)

$x_S \leftarrow 1/m$ for all S

When element e arrives with $\sum_{S:e\in S} x_S < 1$:

Repeatedly do $x_S \leftarrow 2x_S$ forall $S \ni e$ until $\sum_{S:e\in S} x_S \geq 1$.

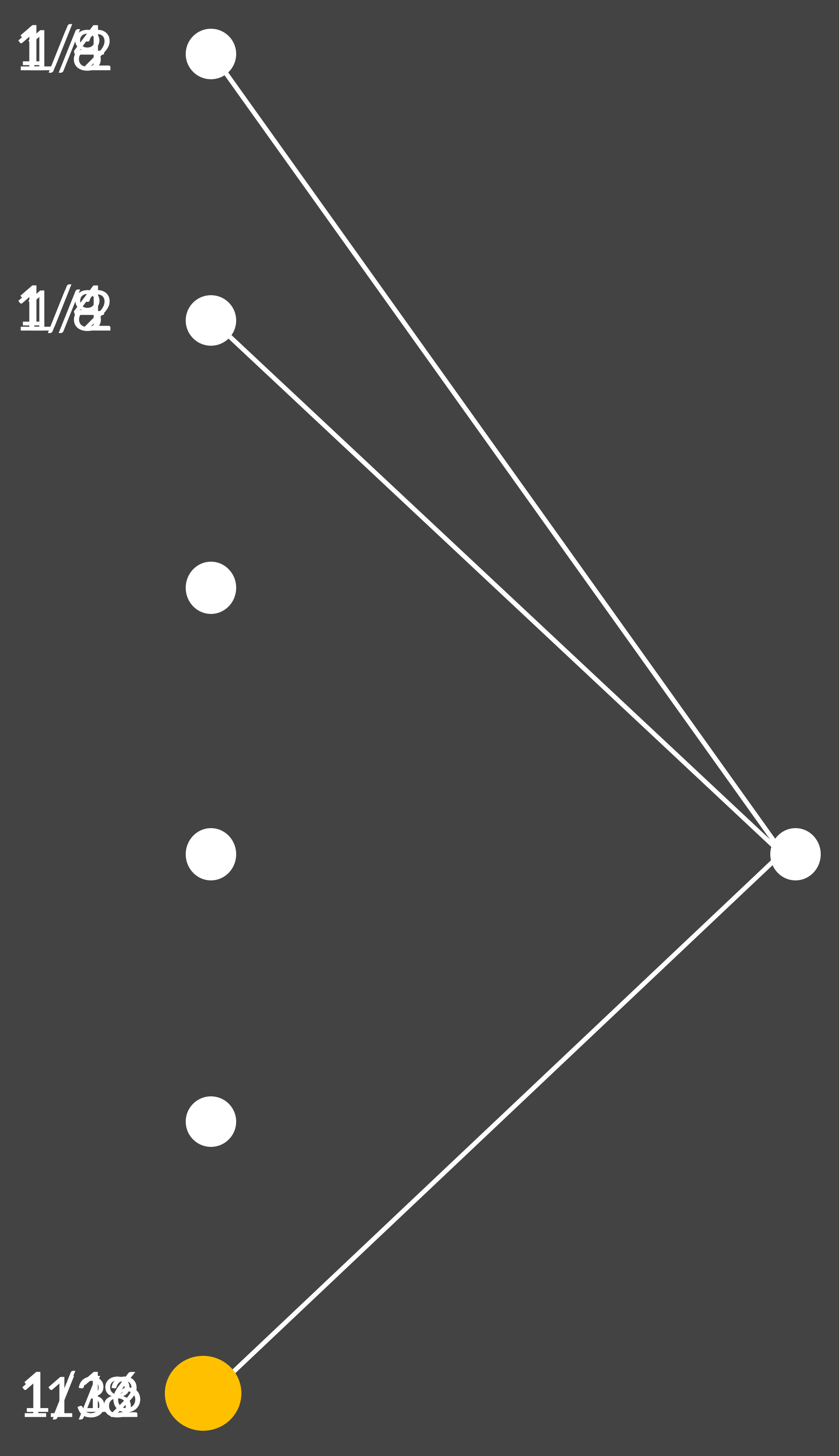
Theorem: Algo 1 has cost $O(\log m)$ OPT.

Proof:

If element e fractionally uncovered:
increase objective by ≤ 2. Also, increase x_{S^*} where S^* is OPT set covering e

"charge" cost increase to S^*

x_{S^*} doubled at most $\log m$ times



solving set cover LP online (costs)

min $\sum_S c(S)\, x_S$

$\sum_{S:e\in S} x_S \;\geq\; 1$ for all e

$x_S \geq 0$

Assume: $c(S) \leq OPT$ for all sets S

Algo 1: (general cost sets)

$x_S \leftarrow 1/m$

When element e arrives with $\sum_{S:e\in S} x_S < 1$:

Repeatedly do $x_S \leftarrow x_S \left(1 + \frac{1}{c(S)}\right)$ forall $S \ni e$ until $\sum_{S:e\in S} x_S \geq 1$.

new cost, old cost, old coverage

$$\sum_S c(S)\, x_S \left(1 + \frac{1}{c(S)}\right) = \sum_S c(S) x_S + \sum_S x_S$$

Theorem: Algo 1 has cost $O(\log m)$ OPT.

Proof:

If element e fractionally uncovered:

increase objective by ≤ 2. Increase x_{S^*} where S^* is OPT set covering e

"charge" cost increase to S^*

x_{S^*} increased at most $c(S) \log m$ times

remove assumption: “guess and double”

Maintain a “guess” G for value of OPT, say $OPT \in (G, 2G]$

Discard all sets with cost more than $2G$

Run α-competitive algorithm

If total cost incurred $> \alpha\ (2G)$, have proof that $OPT > 2G$

set $G \leftarrow 2G$

total cost = geometric sum, so at most $4\alpha\ G$

summary

$\min \sum_S c(S)\, x_S$

$\sum_{S:e\in S} x_S \;\geq\; 1$ for all e

$x_S \geq 0$

When element e arrives:

Until e fractionally covered:

increase its fractional coverage "multiplicatively"

Theorem: get fractional set cover of cost $O(\log m)\ OPT$.

Theorem: (being bit careful) fractional set cover of cost $O(\log f)\ OPT$.

$\Rightarrow$ **Theorem:** Above+rand.round produces integer set cover with E[cost] $O(\log f \log n)\ OPT$.

roadmap for today

Intro to Online Algorithms

Set cover

Theorem: f-competitive using Pitt's algo

Online algo (using relax-and-round)

Theorem: $O(\log m \log n)$-competitive using relax-and-round

Some (almost) matching hardness results

[Prob Wednesday] How to go beyond worst-case?

When requests from known distribution

When requests from **unknown** distribution

lower bounds (1)

n elements

one set for every $\sqrt{n}$ elements: $m = \binom{n}{\sqrt{n}} = \exp(\sqrt{n} \log n)$

Input: $\sqrt{n}$ random elements

$OPT = 1$

Until we pick $\leq \sqrt{n}/2$ sets, then Pr[next random element is uncovered] $\geq 1/2$.

$\Rightarrow \mathbb{E}[cost] \geq \sqrt{n}/2 = \frac{\log m}{\log \log m}$

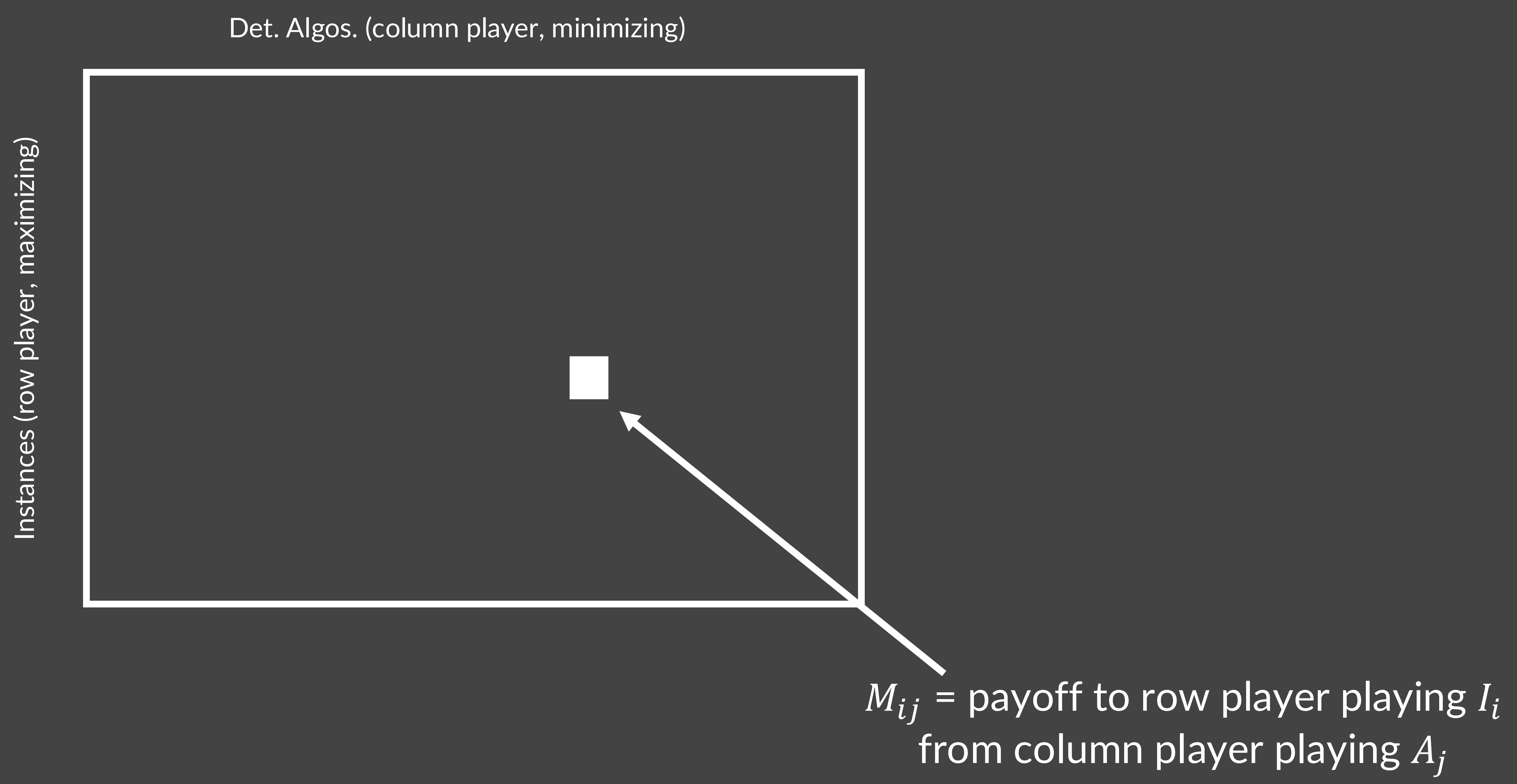
Lower bound holds against randomized algorithms.

Yao's lemma

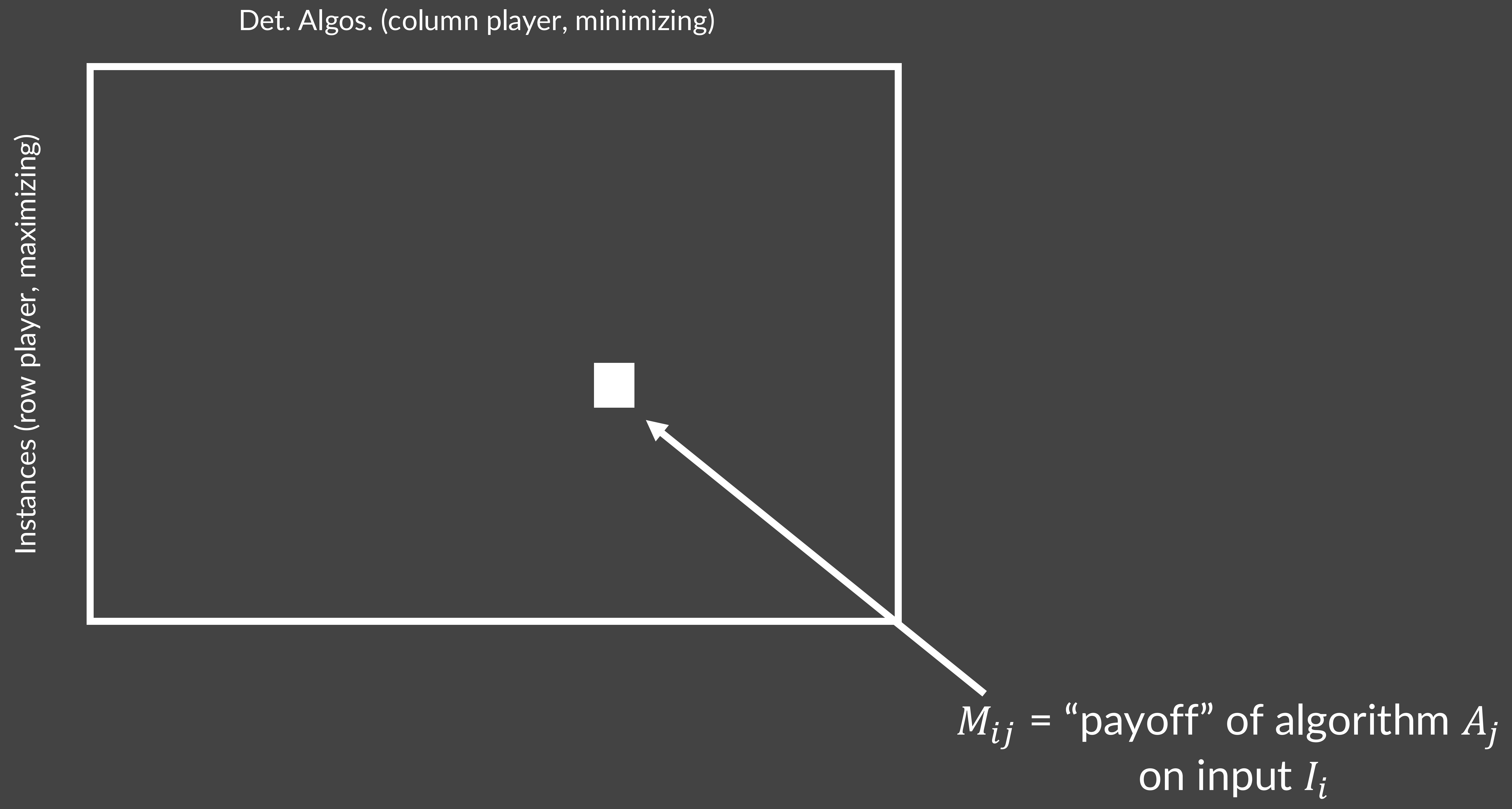
"exists distribution on inputs s.t.
best deterministic algo. has comp.ratio. $\geq blah$"

$\Rightarrow$ "best rand. algo. has comp.ratio. $\geq blah$"

Yao's lemma

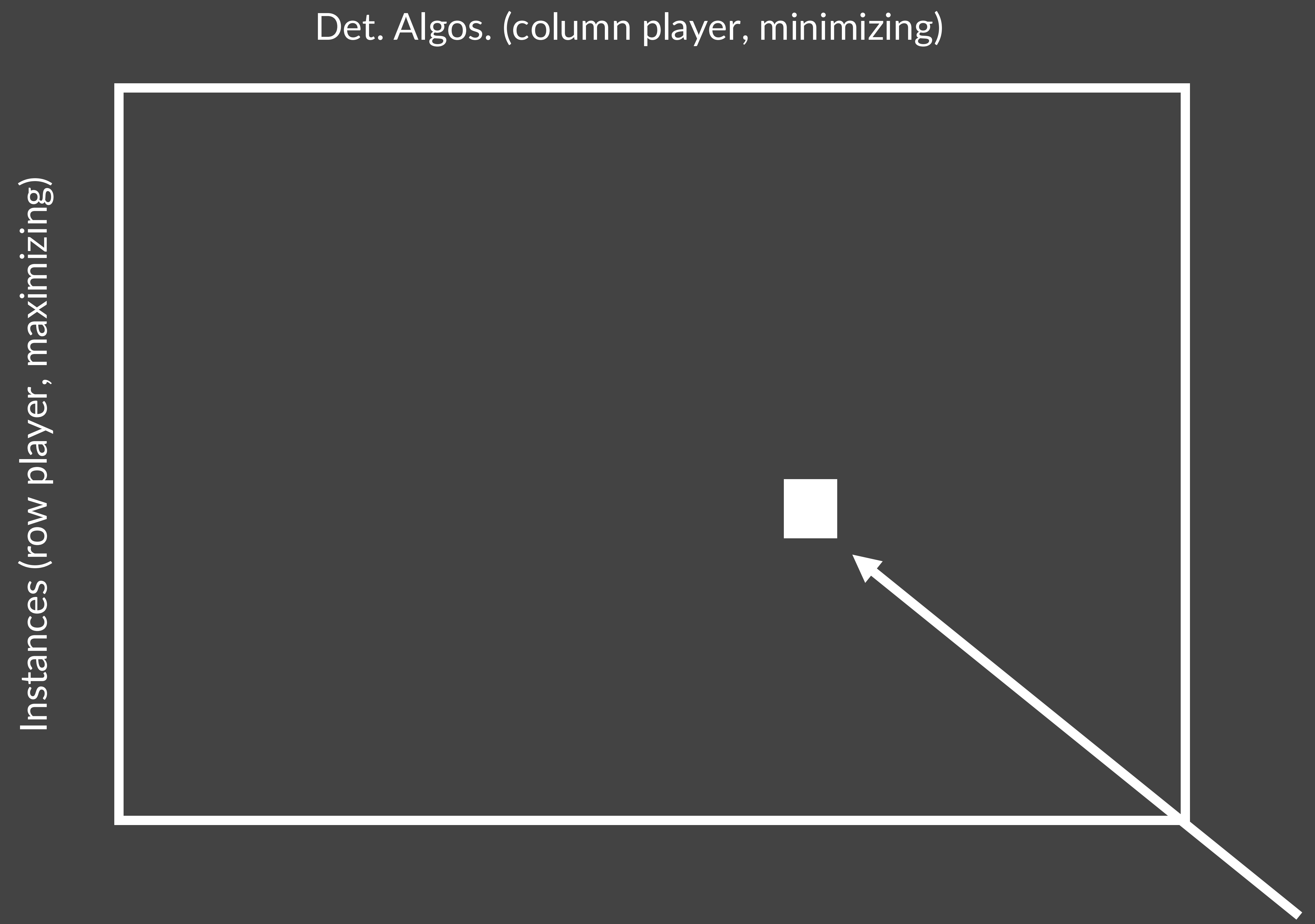

Det. Algos. (column player, minimizing)
Instances (row player, maximizing)
M_{ij} = payoff to row player playing I_i
from column player playing A_j

Yao's lemma



Det. Algos. (column player, minimizing)
Instances (row player, maximizing)
M_{ij} = "payoff" of algorithm A_j on input I_i

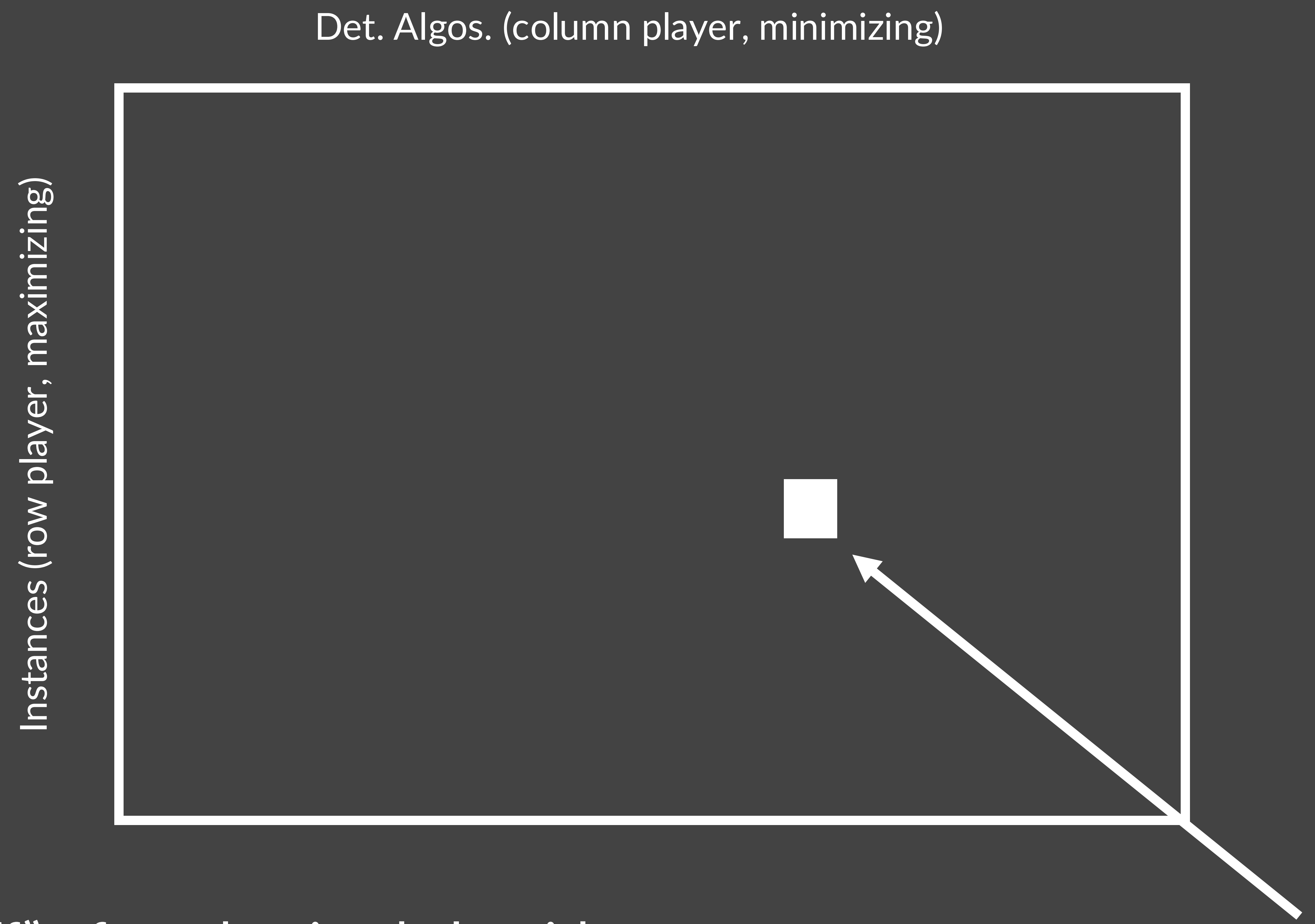
Yao's lemma



$(Mq)_i$ = "payoff" of randomized algorithm q on input I_i

M_{ij} = "payoff" of algorithm A_j on input I_i

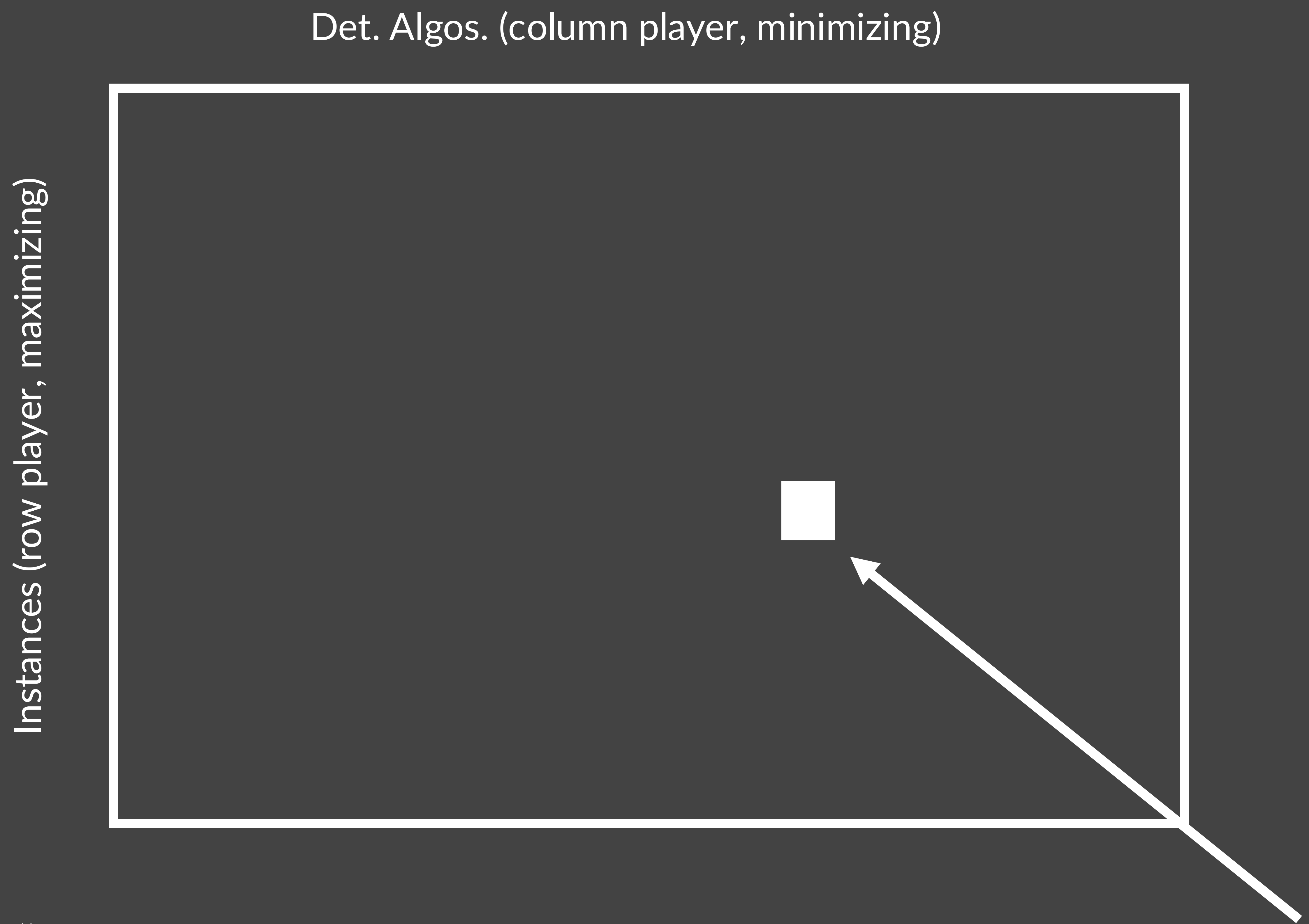
Yao's lemma



$\max_i (Mq)_i$ = "payoff" of randomized algorithm q on **worst-case** input I_i

M_{ij} = "payoff" of algorithm A_j on input I_i

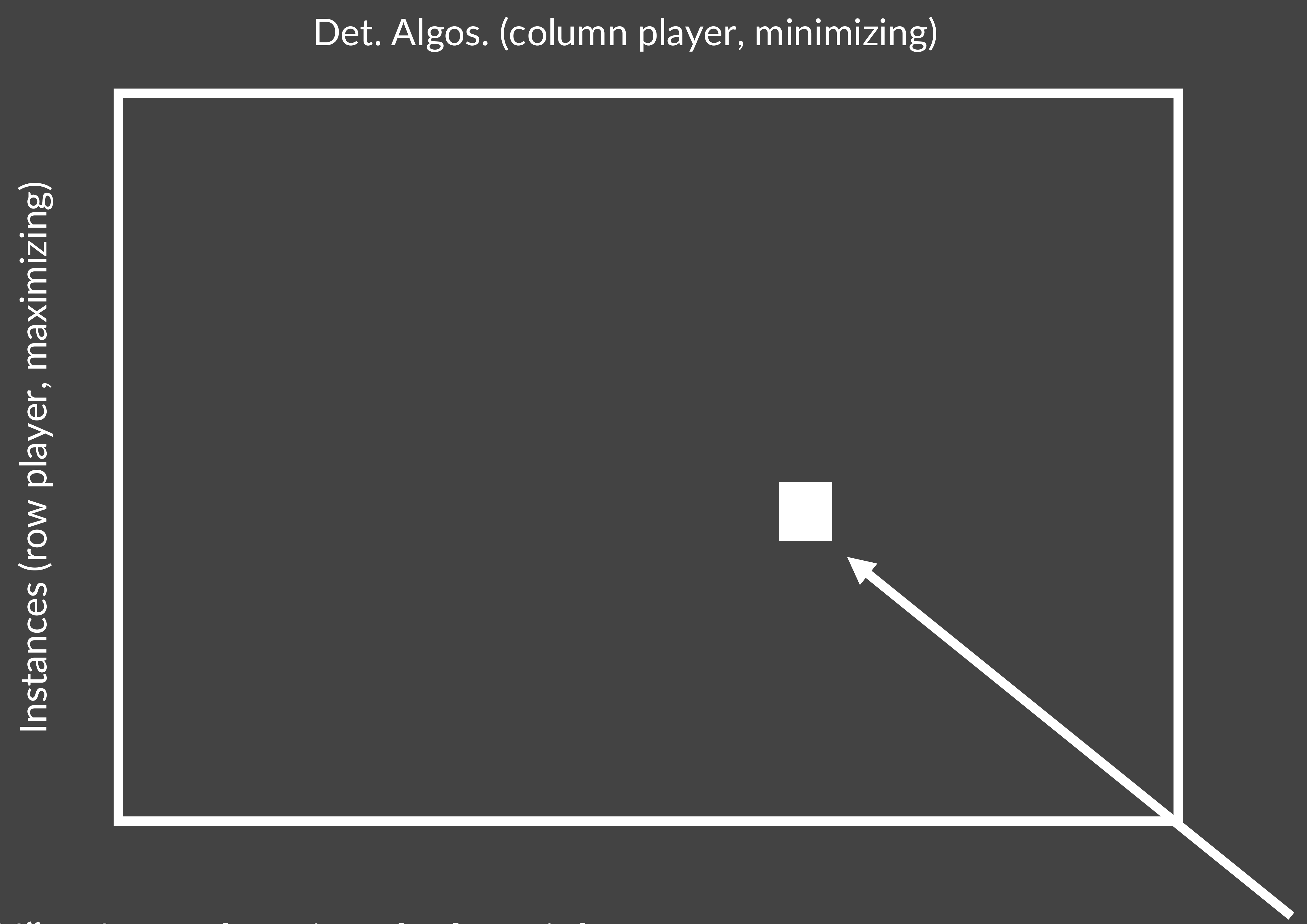
Yao’s lemma



$\max_i (Mq)_i$ = “payoff” of randomized algorithm q on **worst-case** input I_i

M_{ij} = “payoff” of algorithm A_j on input I_i

Yao's lemma



$\max_i e_i^T M q$ = "payoff" of randomized algorithm q on **worst-case** input I_i

M_{ij} = "payoff" of algorithm A_j on input I_i

Yao's Lemma

$$\min_{q \sim C} \max_{e_i} e_i^T M q$$

Yao's Lemma

$$\min_{q \sim C} \max_{e_i} e_i^T M q \geq \max_{p \sim R} \min_{e_j} p^T M e_j$$

"for every rand. algo. there exists input I_i which causes comp.ratio. $\geq blah$"

"exists distrib p on inputs s.t. best det. algo. A_j has comp.ratio. $\geq blah$"

"best rand. algo. has comp.ratio. $\geq blah$"

weak LP duality

in more detail…

for some $\mathcal{D}$, have $\min_A \mathbb{E}_{I\sim\mathcal{D}}[\ A(I)\ ] \geq blah$

"exists distrib on inputs s.t.
best det. algo. has comp.ratio. $\geq blah$"

$$\Rightarrow \max_{\mathcal{D}} \min_A \mathbb{E}_{I\sim\mathcal{D}}[\ A(I)\ ] \geq blah$$

$$\Rightarrow \min_A \max_I \mathbb{E}_A[\ A(I)\ ] \geq blah$$

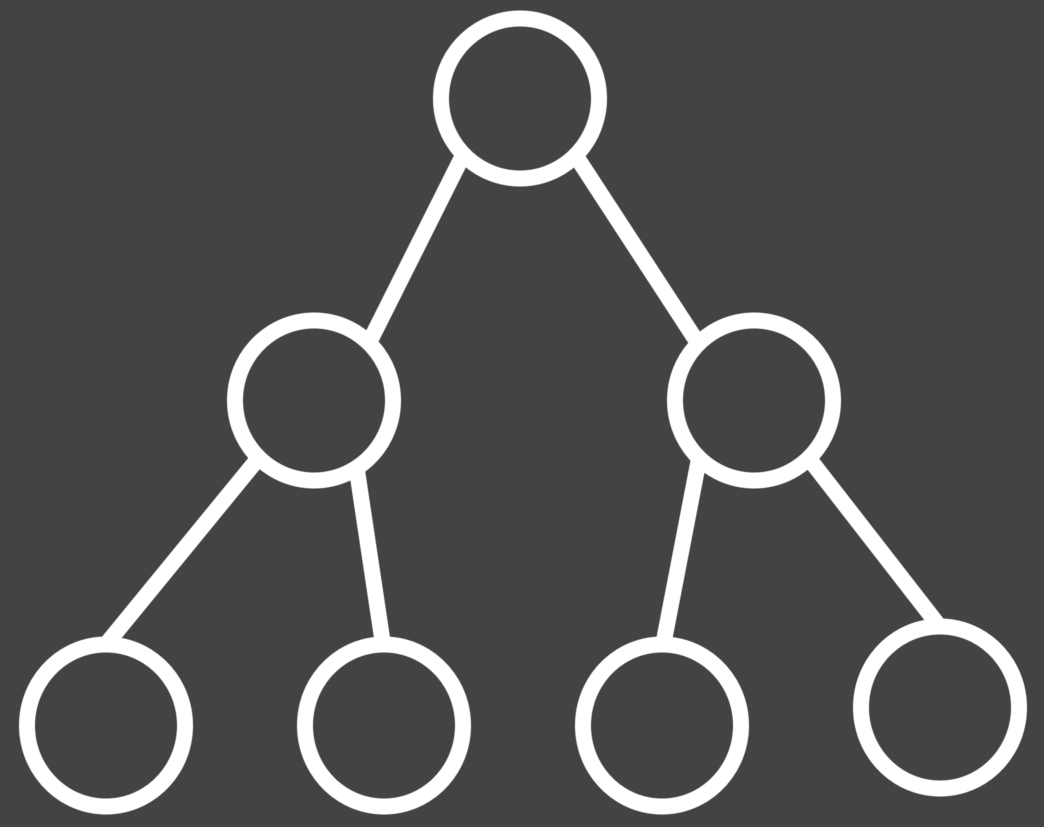
"best rand. algo. has comp.ratio. $\geq blah$"

$$\max_{\mathcal{D}} \min_A \mathbb{E}_{I\sim\mathcal{D}}\left[\frac{A(I)}{OPT(I)}\right] \geq blah$$

$$\max_{\mathcal{D}} \min_A \frac{\mathbb{E}_{I\sim\mathcal{D}}[\ A(I)\ ]}{\mathbb{E}_{I\sim\mathcal{D}}[OPT(I)]} \geq blah$$

$$\Rightarrow \min_A \max_I \frac{\mathbb{E}[A(I)]}{OPT(I)} \geq blah$$

another set cover lower bound



Adversary: pick random leaf, give elements top-down. Sets = leaves, cover ancestors

For any deterministic algorithm A, expected cost = $(k+1)/2 = \Omega(\log m + \log n)$

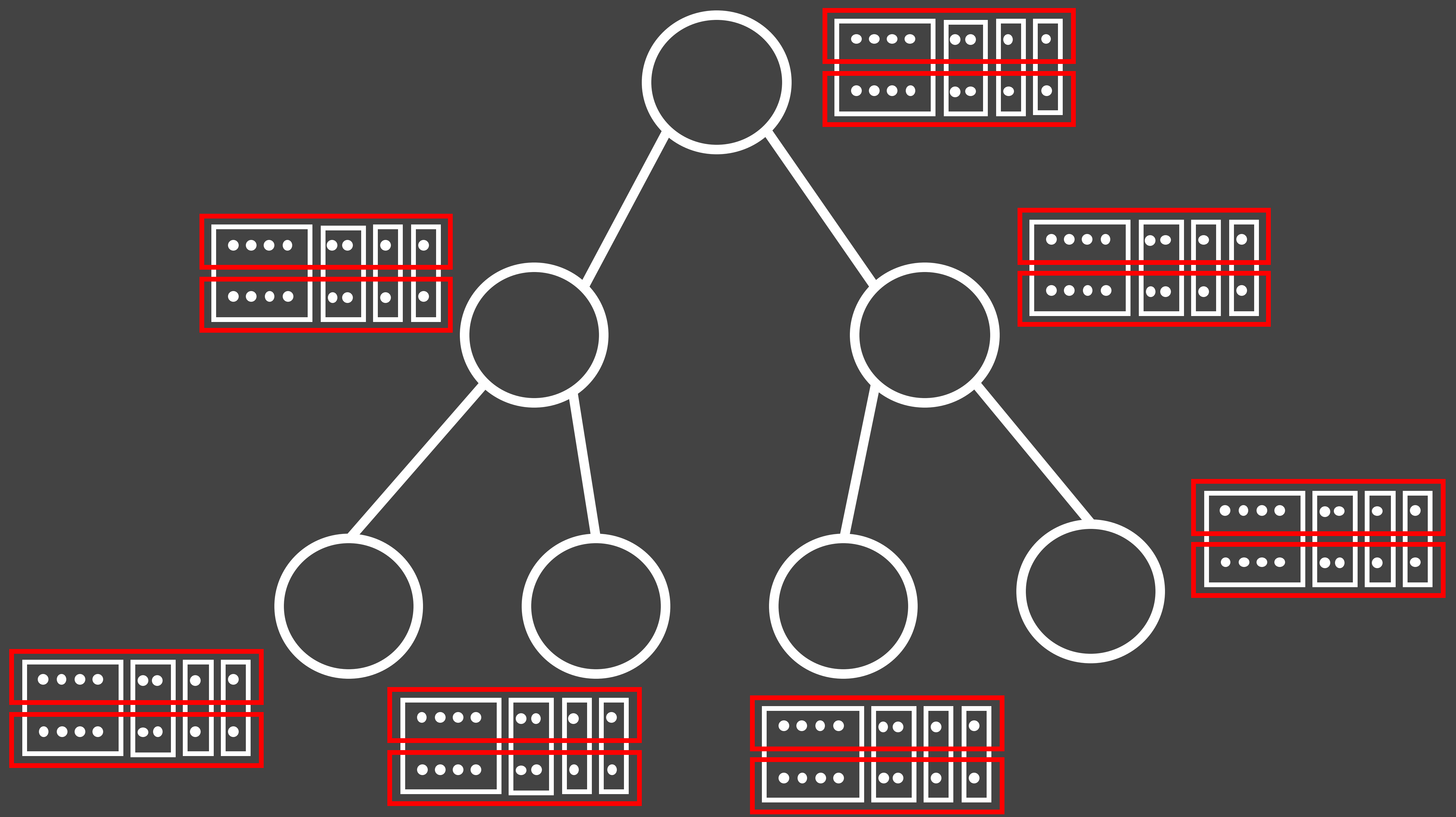
Now use Yao's lemma: exists $\mathcal{D}$ on instances s.t. any deterministic algo has competitive ratio $\geq blah$

$\Rightarrow$ for any randomized algo, exists instance s.t. has competitive ratio $\geq blah$

lower bounds (3)

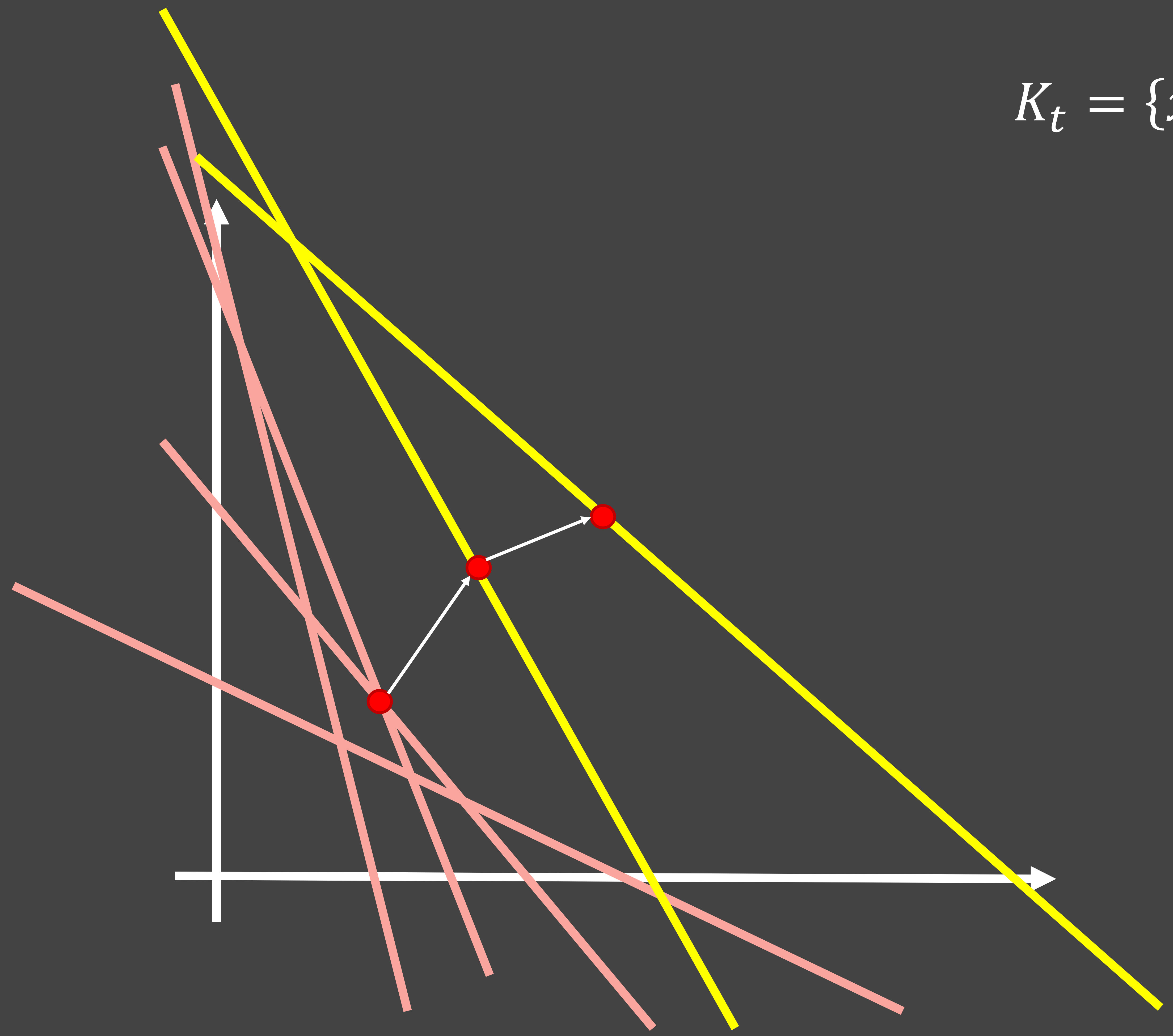
Theorem [Feige/Korman]: every poly-time (randomized) online algo has competititve ratio $\Omega(\log m \log n)$.

Theorem [Feige, Dinur/Steurer]: every poly-time offline (randomized) algo has approx. ratio $\Omega(\log n)$.



Different Perspective on Fractional Algo

unweighted set cover



$$K_t = \{x \geq 0 \mid \langle a^s, x \rangle \geq 1 \ \forall s \leq t, \qquad x \geq x^{t-1}\}$$

unweighted set cover

Request at time t:

$\langle a^t, x \rangle \geq 1$ for some $a^t \in \{0,1\}^n$.

Let $K_t = \{x \geq 0 \mid \langle a^s, x \rangle \geq 1 \; \forall s \leq t, \; x \geq x^{t-1}\}$

Output at time t: point $x^t \in K_t$

Cost at time t: $|| x^t - x^{t-1} ||_1 = \sum_i | x_i^t - x_i^{t-1} |$

Total cost until time t: $||x^t||_1 = \langle \mathbf{1}, x^t \rangle$

$$\min_{x \geq 0} \langle \mathbf{1}, x^t \rangle$$

$$x_1 + x_2 + x_3 + \ldots + x_n \geq 1$$

$$x_2 + x_3 + \ldots + x_n \geq 1$$

$$x_3 + \ldots + x_n \geq 1$$

$$x_n \geq 1$$

an algorithm

$x^0 = {}^1/_n \ \mathbf{1}$

$x^t = \arg\min_{x \in K_t} D(x \,||\, x^{t-1})$ where $D(p||q) = \sum_i (p_i \log {}^{p_i}/_{q_i} - p_i + q_i)$

Theorem #1:

For any algorithm giving $\{y^t \in K_t\}_t$, $||x^t|| \leq \log n \cdot ||y^t|| + 1.$

Colloquially, $Alg \leq \log n \cdot Opt + 1$

an algorithm

$x^0 = {}^1/_n\ \mathbf{1}$

$x^t = \arg\min_{x \in K_t} D(x\,||\,x^{t-1})$ where $D(p||q) = \sum_i (p_i \log {}^{p_i}/_{q_i} - p_i + q_i)$

Fact: $x^t = \arg\min_{x:\langle a^t, x\rangle \geq 1} D(x||x^{t-1})$

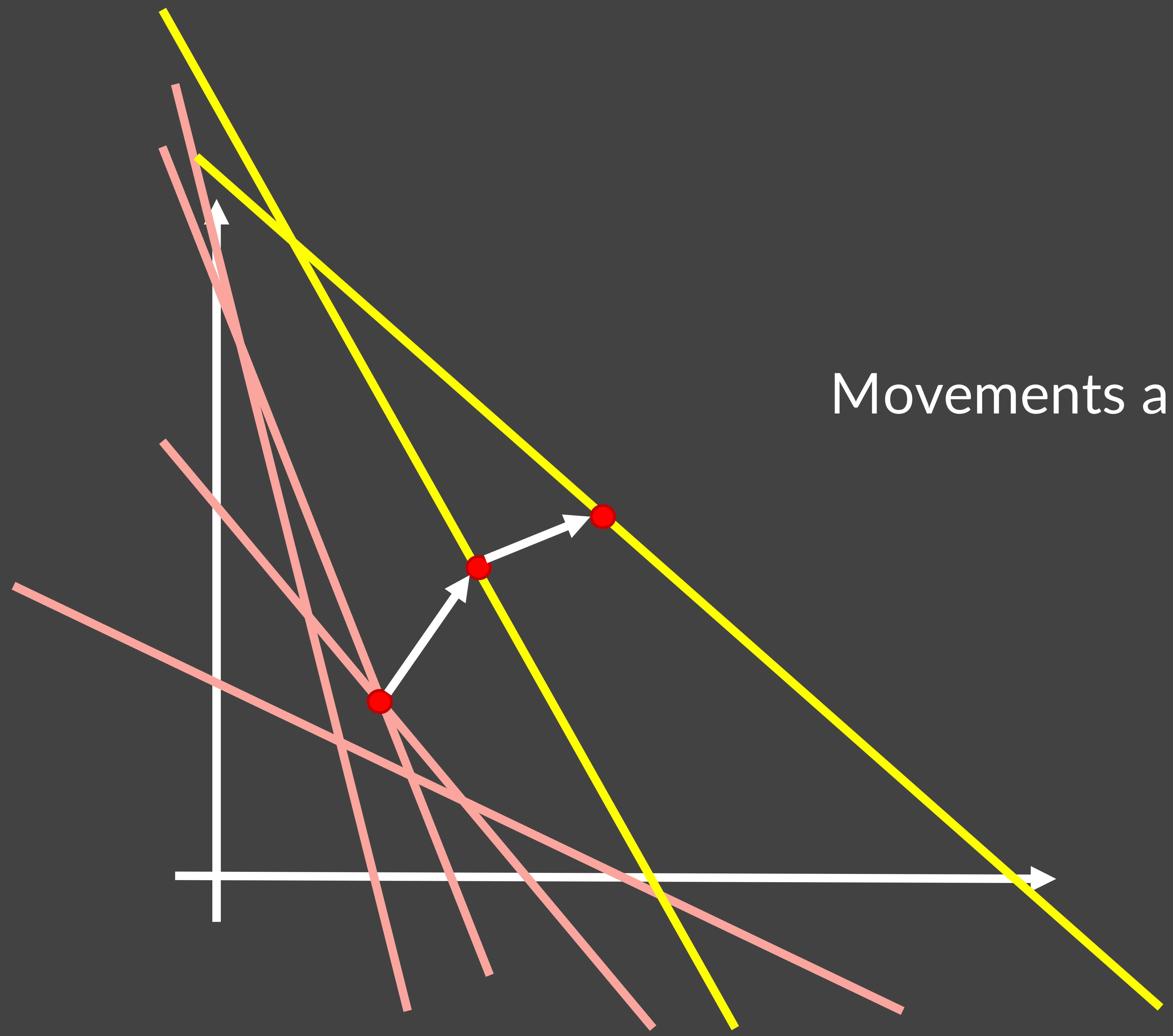
Pf: the Lagrangian is $\sum_i \left(x_i^t \log {}^{x_i^t}/_{x_i^{t-1}} - x_i^t + x_i^{t-1}\right) + \lambda_t (1 - \sum_i a_{ti} x_i^t)$ for $\lambda_t \geq 0$.

So KKT says: $\log {}^{x_i^t}/_{x_i^{t-1}} - \lambda_t a_{ti} = 0$.

$\Rightarrow x_i^t = x_i^{t-1}\, e^{\lambda_t a_{ti}}$.

Mult.weights!!! Since $a_{ti} \geq 0$, x monotone increasing and sat's older constraints too.

unweighted set cover



$$D(p||q) = \sum_i (p_i \log {}^{p_i}/_{q_i} - p_i + q_i)$$

Movements are projections according to D

Can be used to give
potential-function proof

The Price of Uncertainty

price of uncertainty

	Offline	Online
Set Cover	$\Theta(\log n)$	$\Theta(\log m \log n)$

can we do better in non-worst-case settings?

that's it for today...

Intro to Online Algorithms

Set cover

Theorem: f-competitive using Pitt's algo

- Online algo (using relax-and-round)

  Theorem: $O(\log m \log n)$-competitive using relax-and-round

- Some (almost) matching hardness results

[Wednesday] How to go beyond worst-case?

- When requests from **known** distribution

  Theorem: $O(\log m + \log n)$-competitive using universal maps

- When requests from **unknown** distribution

  Theorem: $O(\log m + \log n)$-competitive using learn-or-cover

lecture plan

Lecture #1: Set Cover (worst case)

Lecture #2: Set Cover (beyond worst case), Network design (both)

Lecture #3: Resource Allocation (aka packing)

Lecture #4: Search Problems (aka chasing)