Online Algos: Old and New

Set Cover: Beyond the Worst case

Anupam Gupta (NYU)

NYU | COURANT

analysis of algorithms

Ideally: want to get algorithms that are good for

worst-case ***and*** best-case ***and*** …. all cases.

Worst-case: robustness when data is unpredictable

Best-case: efficiency when data follows anticipated patterns

How to model these?

let’s see glimpse of ideas/techniques in context of **online algos**

price of uncertainty

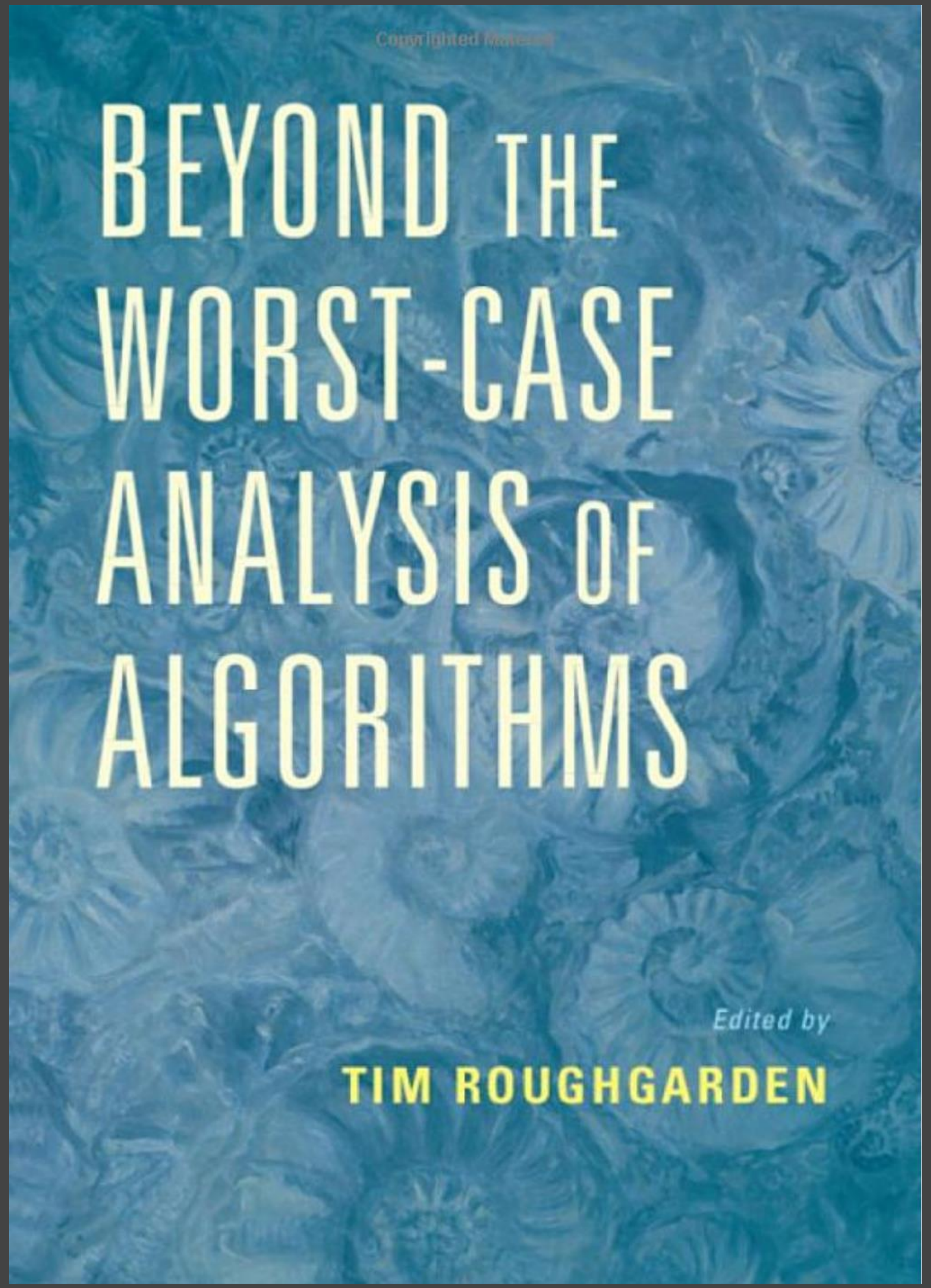
	Offline	Online
Set Cover	$\Theta(\log n)$	$\Theta(\log m \log n)$

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

going beyond the worst case

Ways to model non-worst-case instances

1. special structure to the instances
2. requests are “predictable”
3. arrival order is not worst-case?
4. train NN to find patterns, give predictions
5. …



Known Input Distributions

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

How to go beyond worst-case?

- When requests from known distribution
- When requests from **unknown** distribution

Stochastic Model

Given: set system $(U, \mathcal{F})$, possibly with set cost $c(S)$.

Request sequence $\sigma = \{x_1, x_2, \ldots, x_k\}$ arrives online.

Each x_t is drawn uniformly at random from U.

In general, element $e \in U$ drawn w.p. p_e

On seeing request x_t that is yet uncovered, output some set S_t covering it.

Want to minimize the **E**[cost of sets output on σ]

again, compared to **E**[cost of optimal solution for σ]

[Grandoni G. Leonardi Miettinen Sankowski Singh]

Special solutions: Universal Maps

Given: set system $(U, \mathcal{F})$, possibly with set cost $c(S)$.

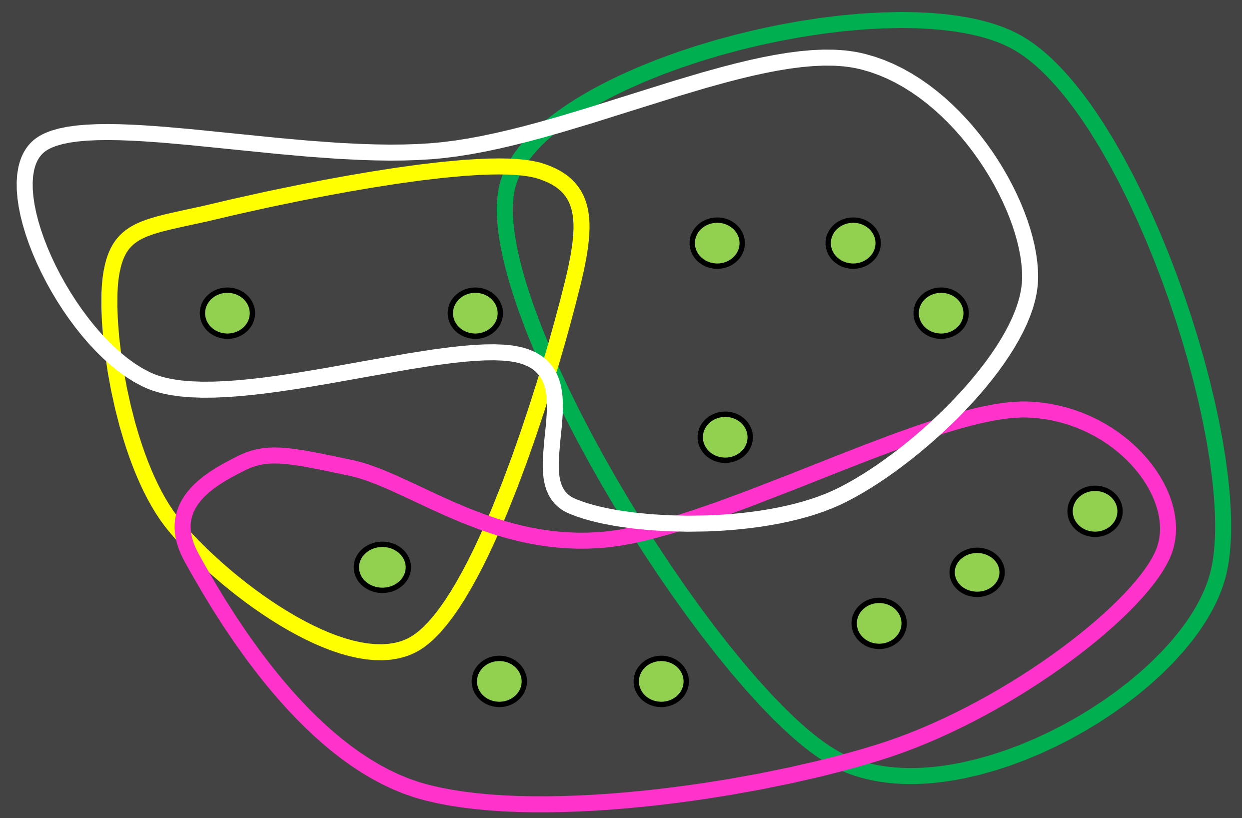
Request sequence $\sigma = \{x_1, x_2, \ldots, x_k\}$ arrives online.

Each x_t is drawn uniformly at random from U.

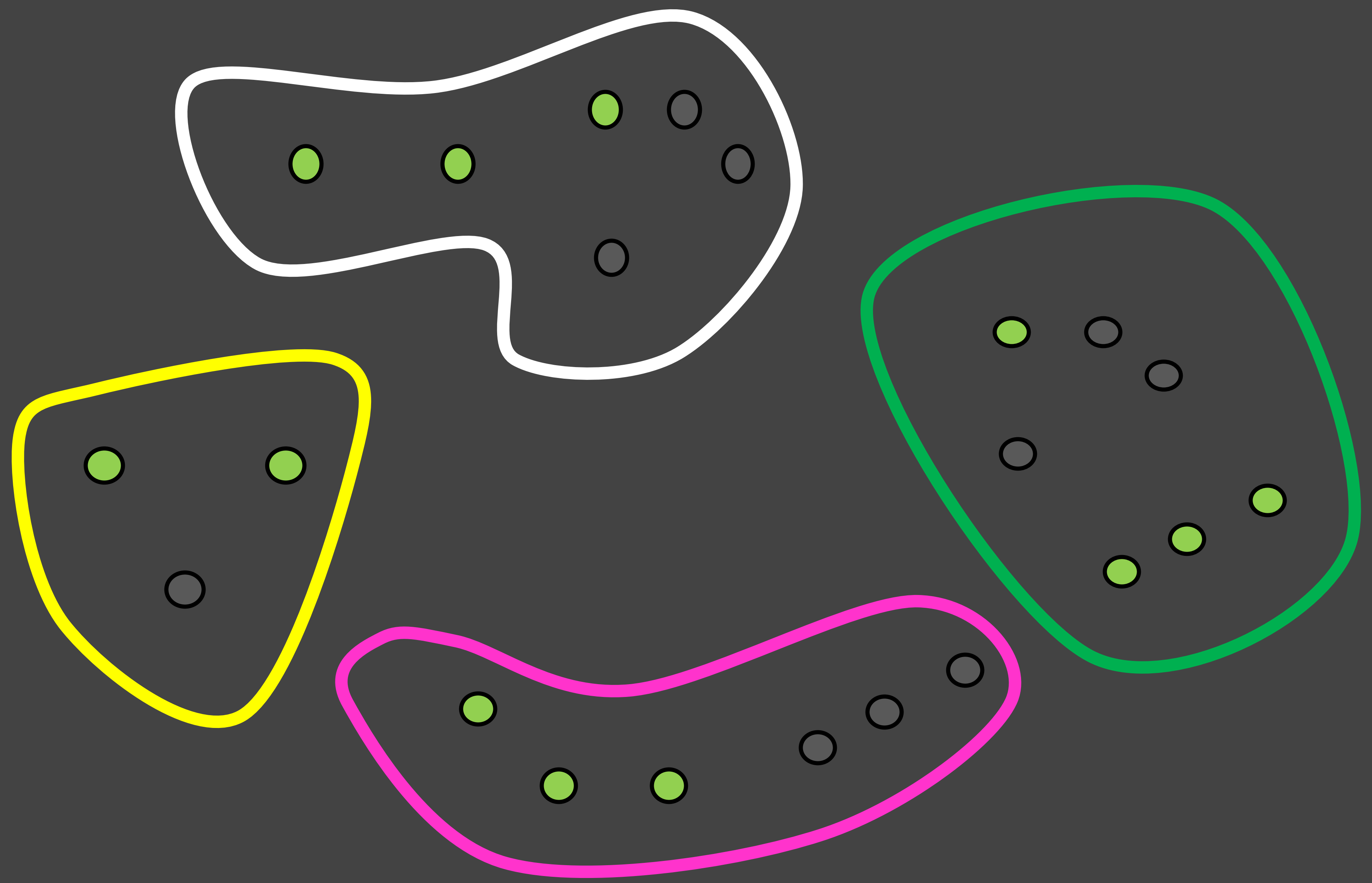
Up-front, give a "universal mapping"
associate set $S(e)$ in $\mathcal{F}$ with each element e in U.

Want to minimize the **E**[cost of sets output on σ]
again, compared to **E**[cost of optimal solution for σ]

Universal Solutions



A potential solution:



Universal Solutions

Theorem: Map given by greedy algorithm is O(log m + log n) competitive.

Algorithm 1: Universal mapping for unweighted set cover.

Data: Set system $(U, \mathscr{S})$.

while $U \neq \emptyset$ **do**

- let $S \leftarrow$ set in $\mathscr{S}$ maximizing $|S \cap U|$;
- $\mathbf{S}(v) \leftarrow S$ for each $v \in S \cap U$;
- $U \leftarrow U \setminus S$;

Each element is mapped to first set in greedy that covers it.

Exists Good Universal Map

Fix some sequence length k. Let μ = **E**[OPT cost on length k sequence]

Lemma: Exist 2μ sets in $\mathcal{F}$ that cover $(1-\delta)n$ elements of U, where $\delta = (3\mu \log m)/k$.

Proof: next slide

Note: Lemma is existential. But greedy covers as many elements using $2\mu \log n$ sets.

Exists Good Universal Map

Fix some sequence length k.   Let μ = **E**[OPT cost on length k sequence]

Lemma: Exist 2μ sets in $\mathcal{F}$ that cover $(1-\delta)n$ elements of U, where $\delta = (3\mu \log m)/k$.

Proof: Consider all the n^k sequences

Expected number of sets in OPT is μ
⇒ at least ½n^k "good" sequences can be covered by 2μ sets $L = m^{2\mu}$

Consider "bags" of elements $C_1, C_2, \ldots, C_L$ got by taking unions of 2μ sets.

For contradiction, suppose each bag has $\leq (1-\delta)n$ elements.
Another way to generate good sequences: pick C_i and pick k elements

So: $L \times [(1-\delta)n]^k \geq$ ½n^k $\Rightarrow m^{2\mu} e^{-\delta k} \geq 1/2.$ ∎

Wrap-up

Fix some sequence length k. Let μ = **E**[OPT cost on length k sequence]

Lemma: Exist 2μ sets in $\mathcal{F}$ that cover $(1-\delta)n$ elements of U, where $\delta = (3\mu \log m)/k$.

Recap: greedy covers $(1-\delta)n$ "happy" elements using $2\mu \log n$ sets.

Happy elements in our sequence covers by these many sets

E[sad elements in our sequence] = $\delta k = O(\mu \log m)$, one set for each

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

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

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

How to go beyond worst-case?

When requests from known distribution

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

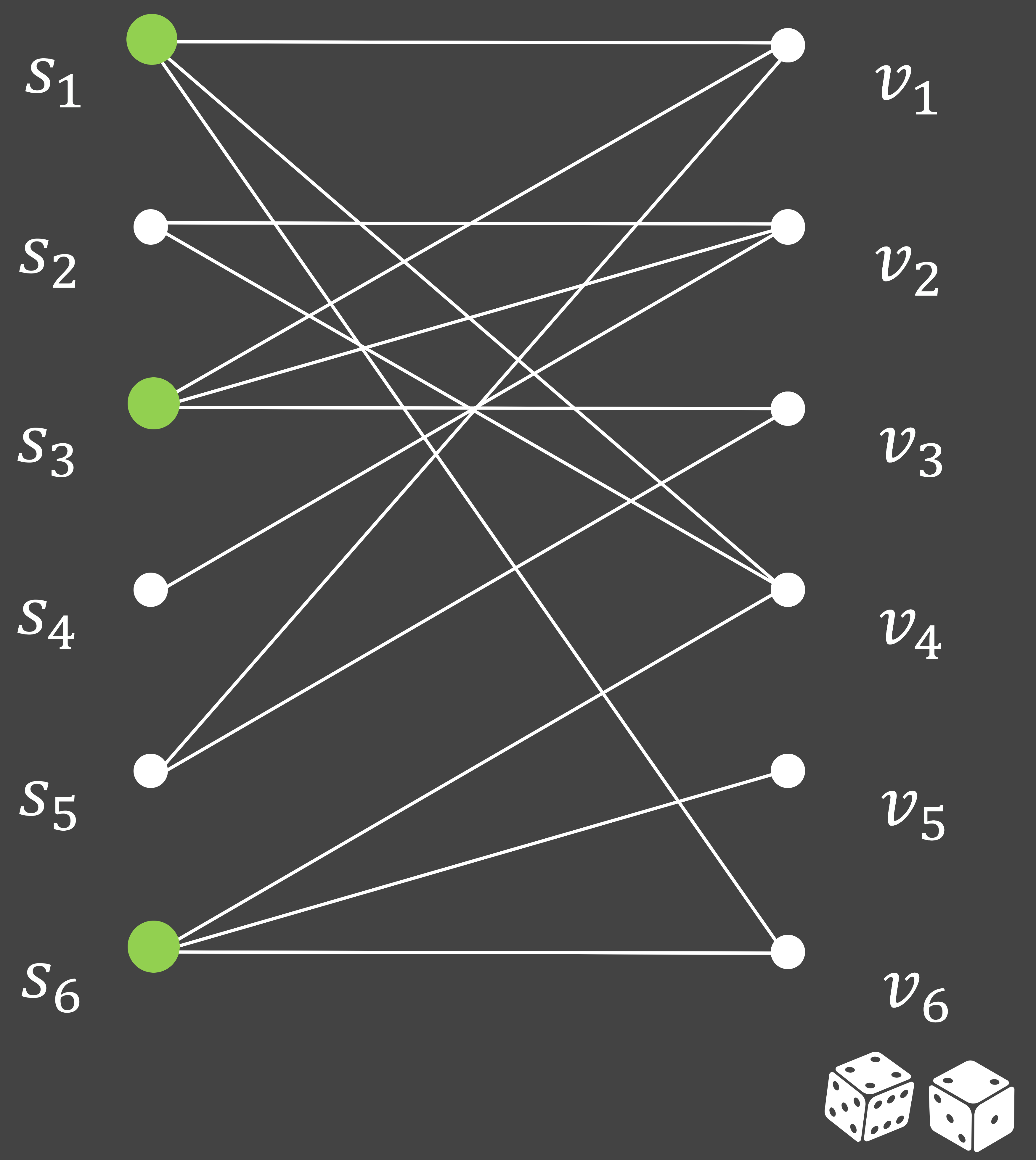
When requests from **unknown** distribution

Random Order

Random Order (RO) model

$\mathcal{F}$

m sets



$\mathcal{U}$

n elements

[G. Kehne Levin FOCS 21]

LearnOrCover

(Unit cost, exp time)

when random element v arrives

    if v not already covered, in parallel:

        1. select random remaining hand

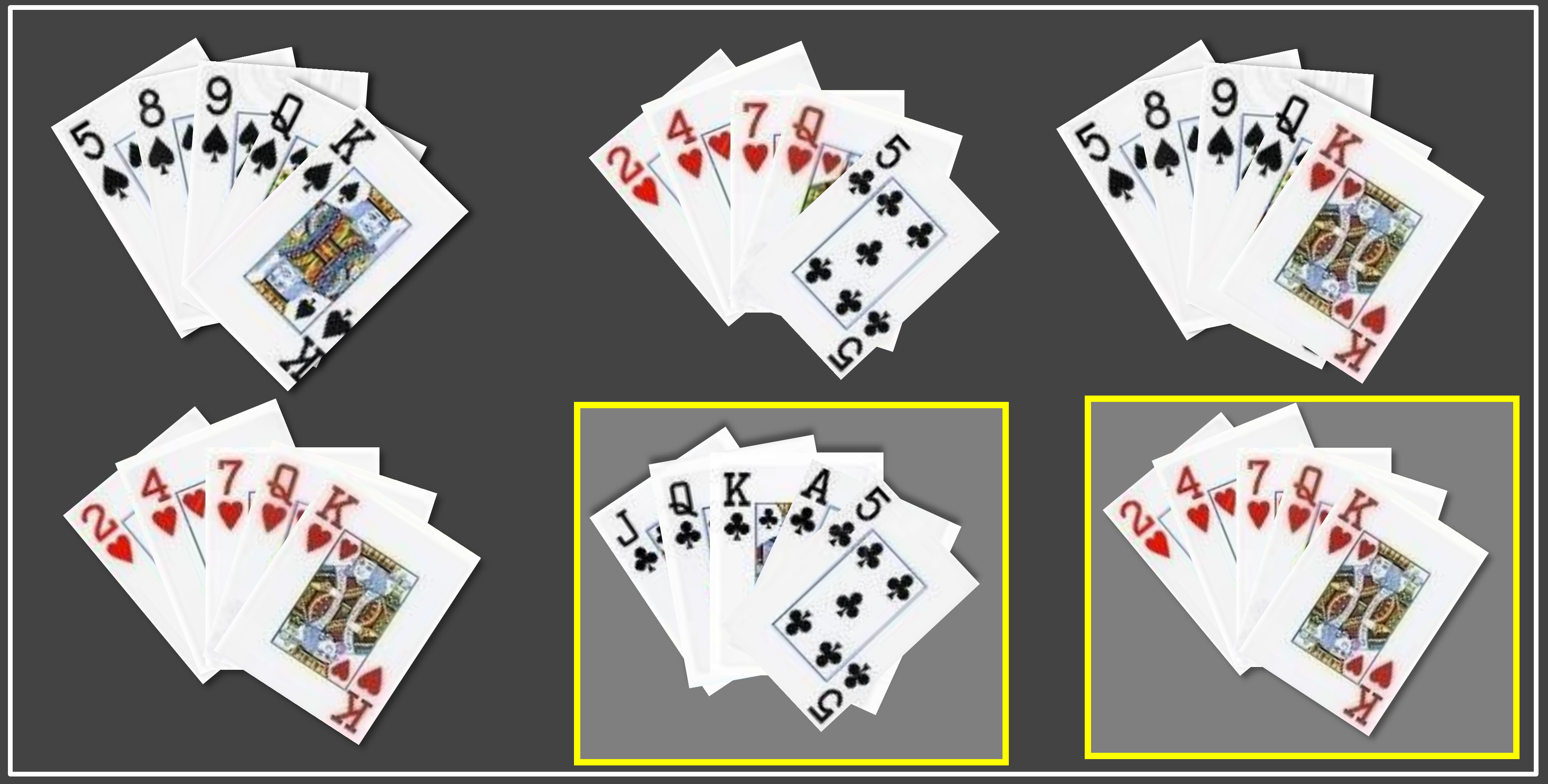
            pick random set from it

        2. remove sols that don't cover v

        pick any set covering v

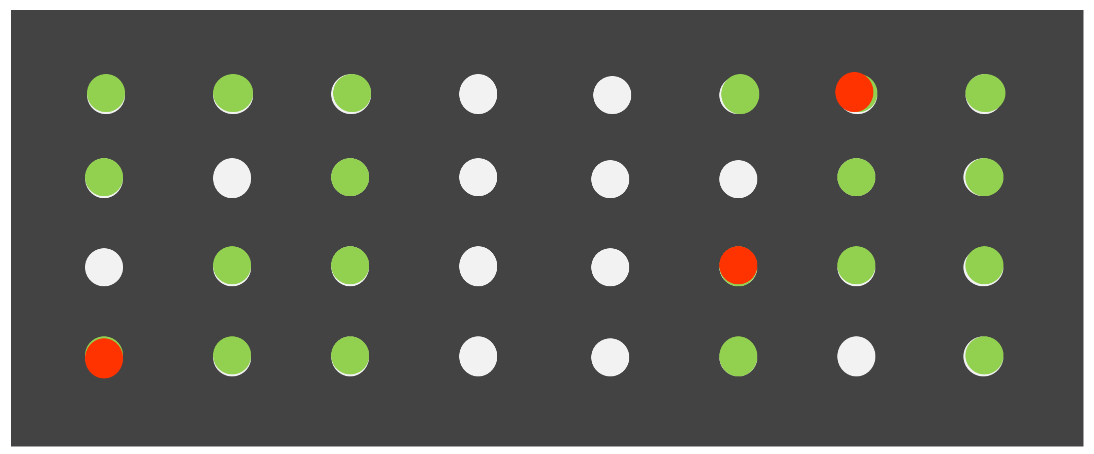
Main Q: how many elements uncovered on arrival?

"hands" of possible solutions



$\mathcal{P}$

$\binom{m}{k}$



$\mathcal{U}$

Sol R:

[G. Kehne Levin FOCS 21]

LearnOrCover

(Unit cost, exp time)

when random element v arrives

  if v not already covered, in parallel:

    1. select random remaining hand

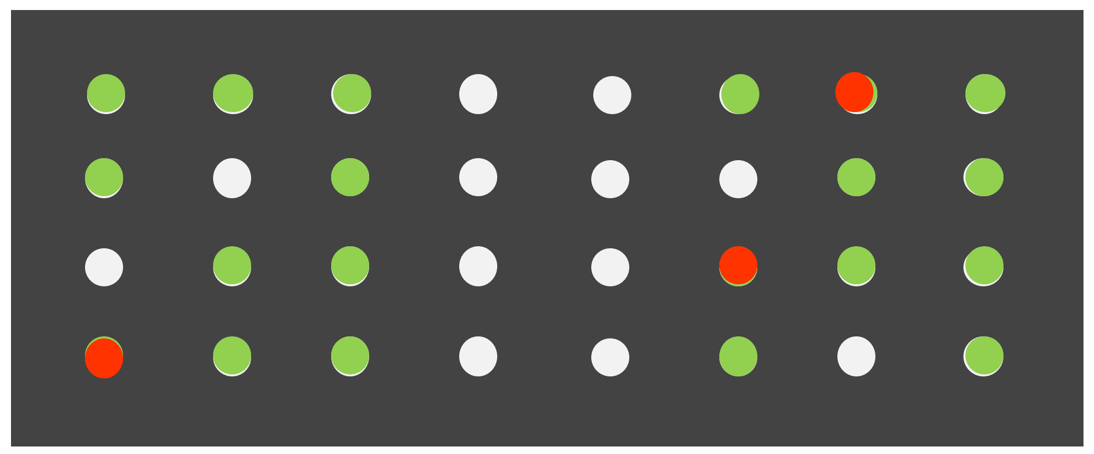
      pick random set from it

    2. remove sols that don't cover v

  pick any set covering v

“hands” of possible solutions

$\mathcal{P}$



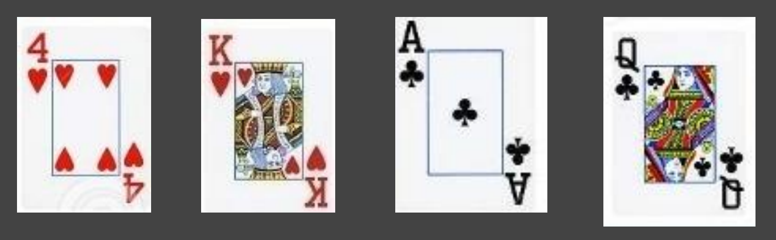
$\mathcal{U}$

Q: do ½ of remaining hands cover ½ of uncovered elements?

Yes: random set covers many uncovered elements!

No: random element removes many hands!!

Sol R:



Case 1: $\geq 1/2$ of $P \in \mathcal{P}$ cover $\geq 1/2$ of $\mathcal{U}$.

R covers $\frac{|\mathcal{U}|}{4k}$ in expectation.

$\mathcal{U}$ shrinks by $\left(1 - \frac{1}{4k}\right)$ in expectation.

$|\mathcal{U}|$ initially n

$\Rightarrow \quad O(k \log n)$ COVER steps suffice.

Case 2: $> 1/2$ of $P \in \mathcal{P}$ cover $< 1/2$ of $\mathcal{U}$.

$\geq 1/2$ of $P \in \mathcal{P}$ pruned w.p. $1/2$.

$\mathcal{P}$ shrinks by $3/4$ in expectation.

$|\mathcal{P}|$ initially $\binom{m}{k} \approx m^k$

$\Rightarrow \quad O(k \log m)$ LEARN steps suffice.

$\Rightarrow O(k \log mn)$ steps suffice.

LearnOrCover
(Unit cost, exp time)

Case 1: (COVER)

$\mathcal{U}$ shrinks by $\left(1 - \frac{1}{4k}\right)$ in expectation.

Case 2: (LEARN)

$\mathcal{P}$ shrinks by $3/4$ in expectation.

$$\Phi = \frac{1}{k}\log|\mathcal{P}| + \log|\mathcal{U}|$$

Claim 1: $\Phi(0) = O(\log mn)$ and $\Phi(t) \geq 0$.

Claim 2: If v uncovered, then $E[\Delta\Phi] \leq -\Omega\left(\frac{1}{k}\right)$.

How to make polytime?

Can we reuse LEARN/COVER intuition?

LearnOrCover

(Unit cost)

Init. $x \leftarrow 1/m$.

@ time t, element v arrives:

If v covered, do nothing.

Else:

(I) Buy random $R \sim x$.

(II) $\forall S \ni v$, set $x_S \leftarrow e \cdot x_S$.

Renormalize $x \leftarrow x/\| x \|_1$.

Buy arbitrary set to cover v.

$\sum_S x_S^* \log \frac{x_S^*}{x_S^t}$

Idea: Measure convergence with potential function

$$\Phi(t) = c_1 \, KL(x^* || x^t) + c_2 \log|\mathcal{U}^t|$$

$\mathcal{U}^t$:= uncovered elements @ time t

x^* := uniform distribution on OPT

Claim 1: $\Phi(0) = O(\log mn)$, and $\Phi(t) \geq 0$.

Claim 2: If v uncovered, then $E[\Delta\Phi] \leq -\frac{1}{k}$.

If $\mathbb{E}_v[\, x_v \,] > \frac{1}{4} \Rightarrow \mathbb{E}_R[k \, \Delta \log|\mathcal{U}^t|]$ drops by $\Omega(1)$.

Else $\mathbb{E}_v[k \, \Delta KL]$ drops by $\Omega(1)$.

(Recall $k = |OPT|$)

picture for set cover

Intro to Online Algorithms

Set cover

Online algo (using relax-and-round)

Some (almost) matching hardness results

How to go beyond worst-case?

When requests from known distribution

When requests from unknown distribution

Theorem: f-competitive using Pitt's algo

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

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

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

Online Algos: Old and New

Lecture 2b: Network Design, Worst-case and Beyond

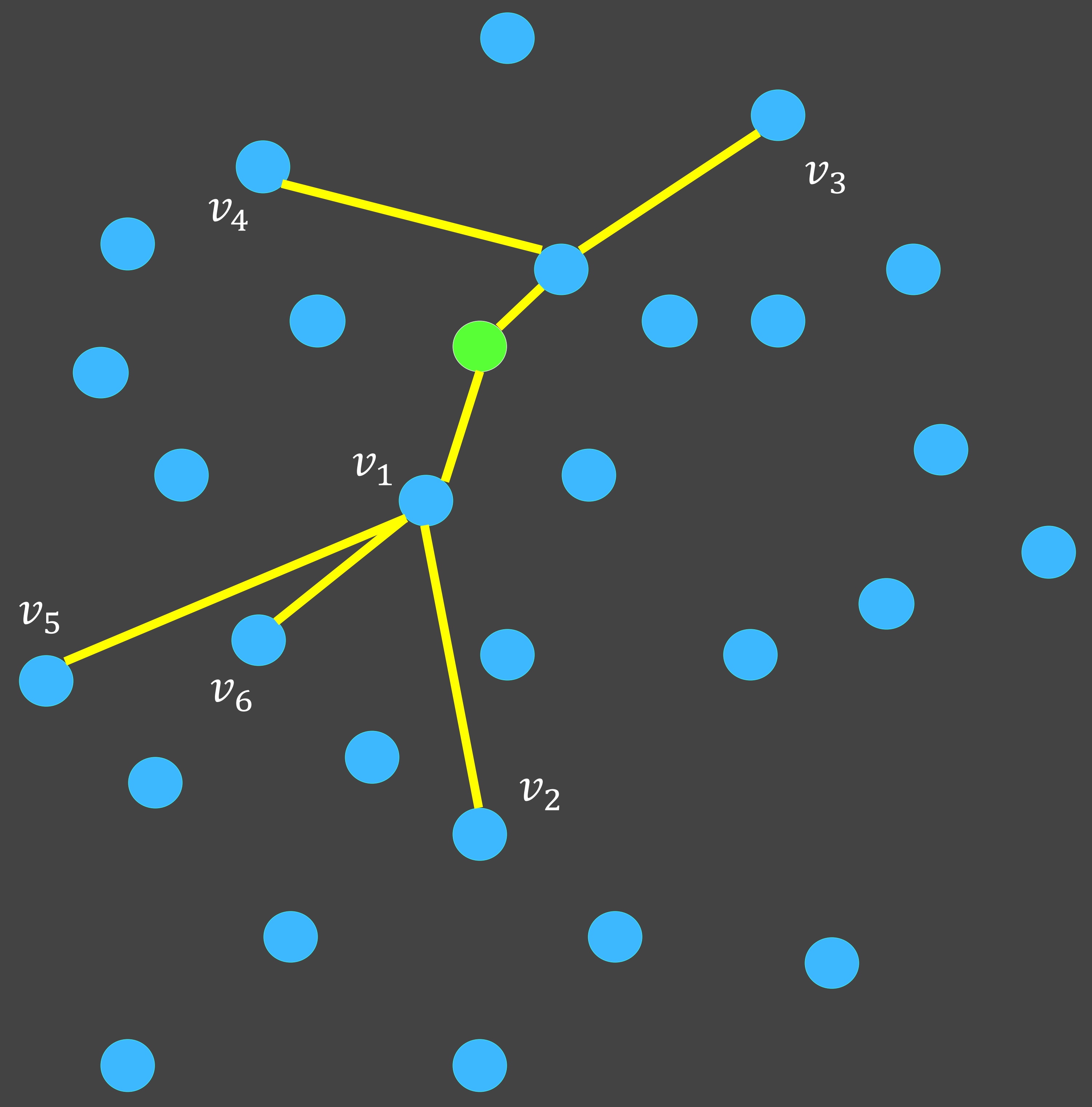
Anupam Gupta (NYU)

NYU | COURANT

Online Network Design

Metric space. n points arrive over time, maintain a connected graph.

Goal: minimize cost of tree



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.

[Imase Waxman 91]

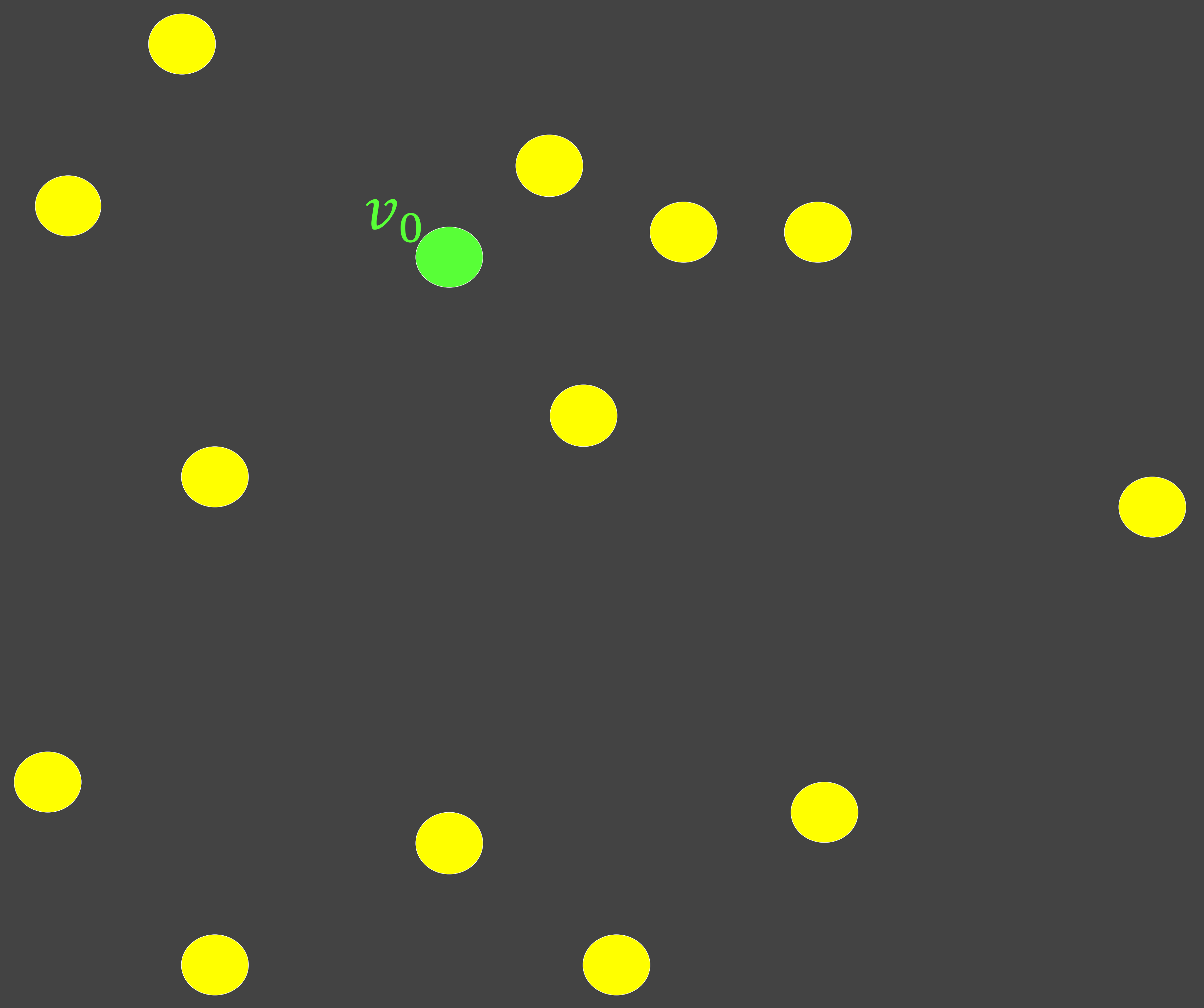
Steiner Tree

(steiner) tree offline

Underlying metric space $\mathcal{M}$, root vertex v_0

Given T terminals

find shortest tree connecting $T \cup \{v_0\}$ in $\mathcal{M}$



Thm 1: $MST(T \cup \{v_0\})$ is a 2-approximation

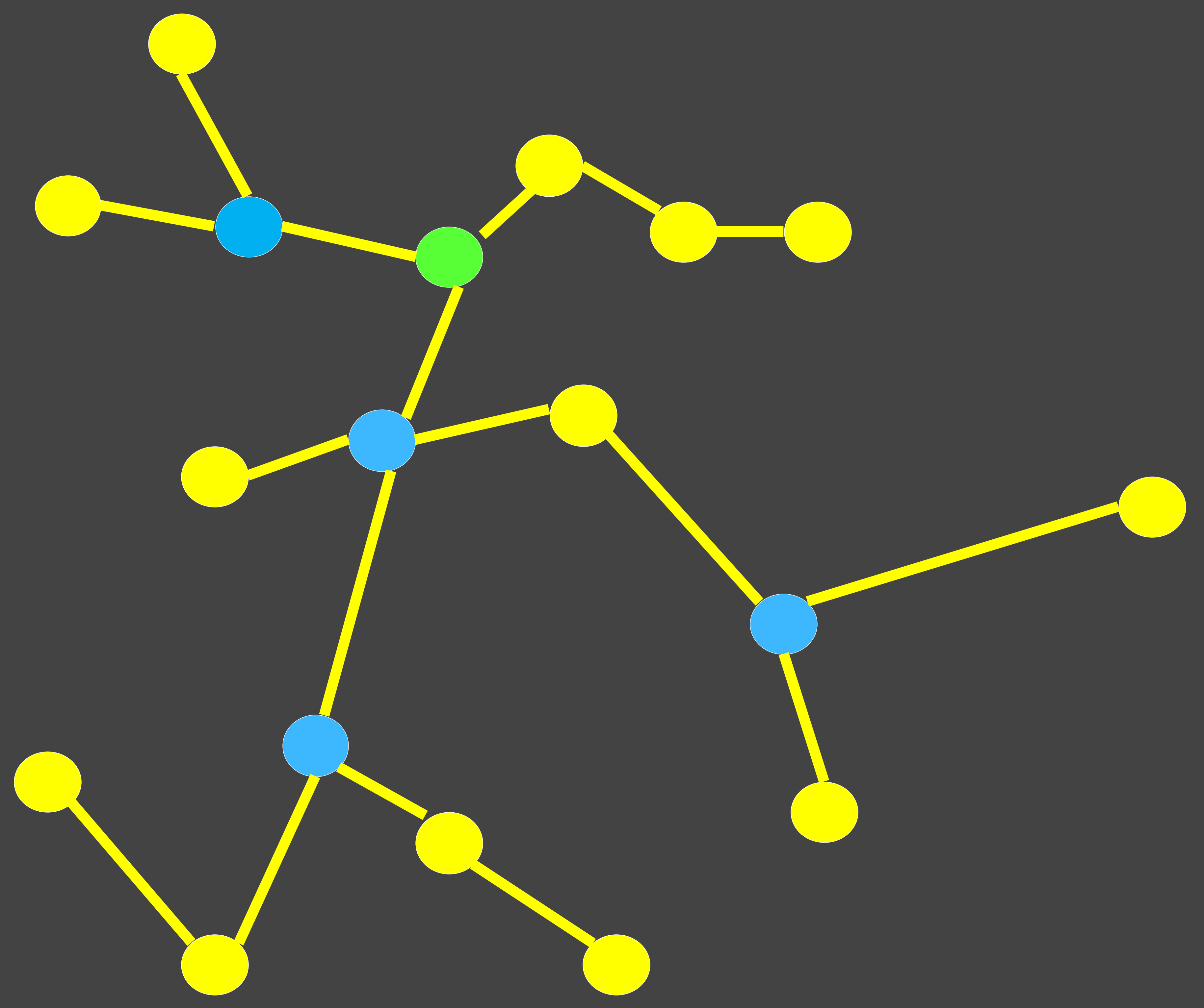
Proof:

(steiner) tree offline

Underlying metric space $\mathcal{M}$, root vertex v_0

Given T terminals

find shortest tree connecting $T \cup \{v_0\}$ in $\mathcal{M}$



Thm 1: $MST(T \cup \{v_0\})$ is a 2-approximation

Proof:

suppose this is OPT

(steiner) tree offline

Underlying metric space $\mathcal{M}$, root vertex v_0

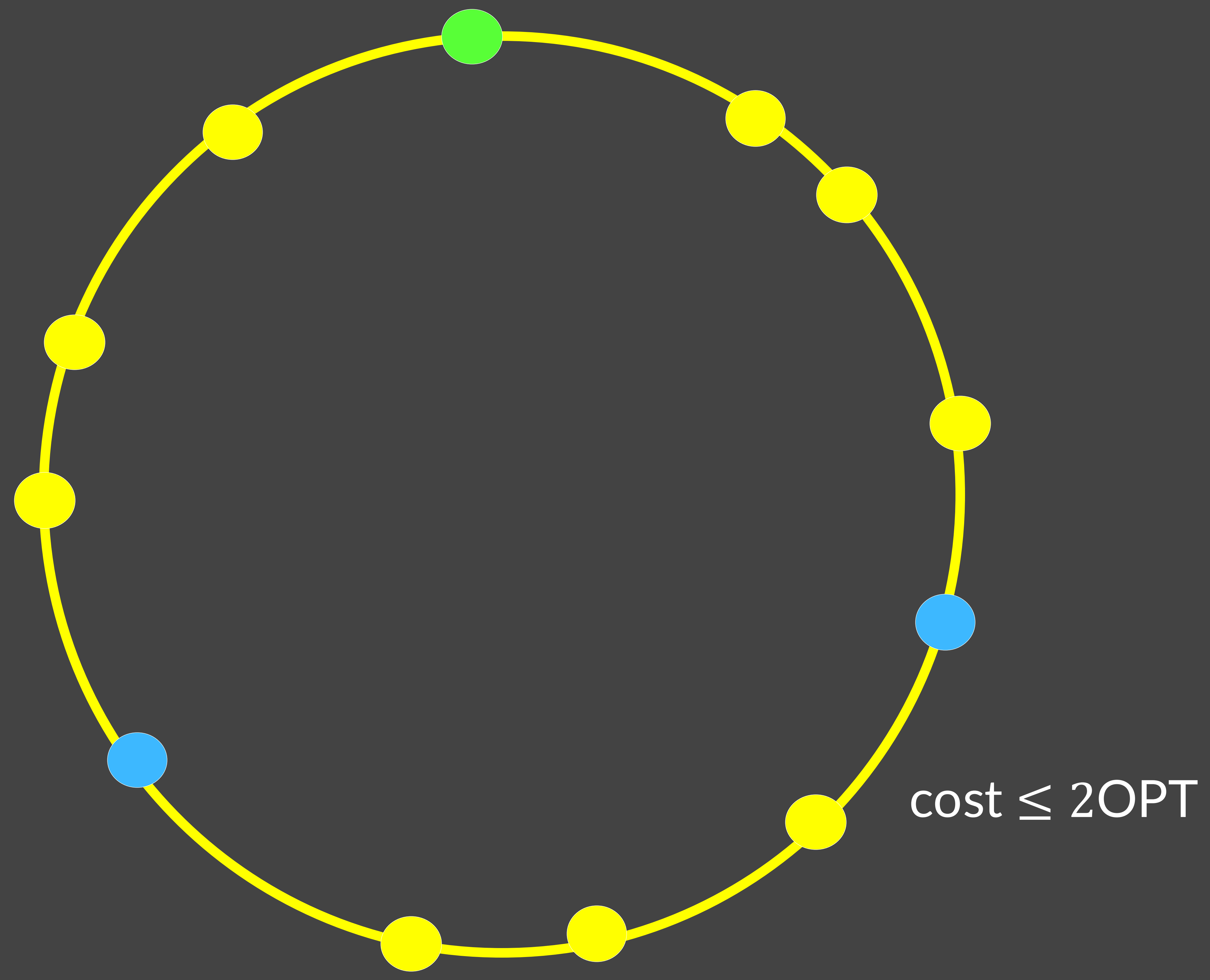
Given T terminals

find shortest tree connecting $T \cup \{v_0\}$ in $\mathcal{M}$

Thm 1: $MST(T \cup \{v_0\})$ is a 2-approximation

Thm 2: Exist $\ln(4)$-approx. (~1.386)

[Byrka Grandoni Rothvoss Sanita, Traub and Zenklusen]



online Steiner tree: model choices

Known Metric	Unknown Metric
Metric $\mathcal{M}$ and root v_0 is fixed and public	root v_0 is fixed and public
Adversary chooses T requests	Adversary chooses metric $\mathcal{M}$ and T requests
Algo sees requests $v_1, v_2, \ldots, v_T$ one-by-one	Algo sees requests $v_1, v_2, \ldots, v_T$ one-by-one
	When v_t seen, we learn $d(v_t, v_s)\ \forall s < t$

$\forall t$, when request v_t seen, must connect it to root component

online (steiner) tree

connect v_t to $\arg\min_{s<t} d(v_s, v_t)$

Thm 1: greedy is $O(\log T)$ competitive

number of requests

works in unknown metric

[Imase Waxman 91]

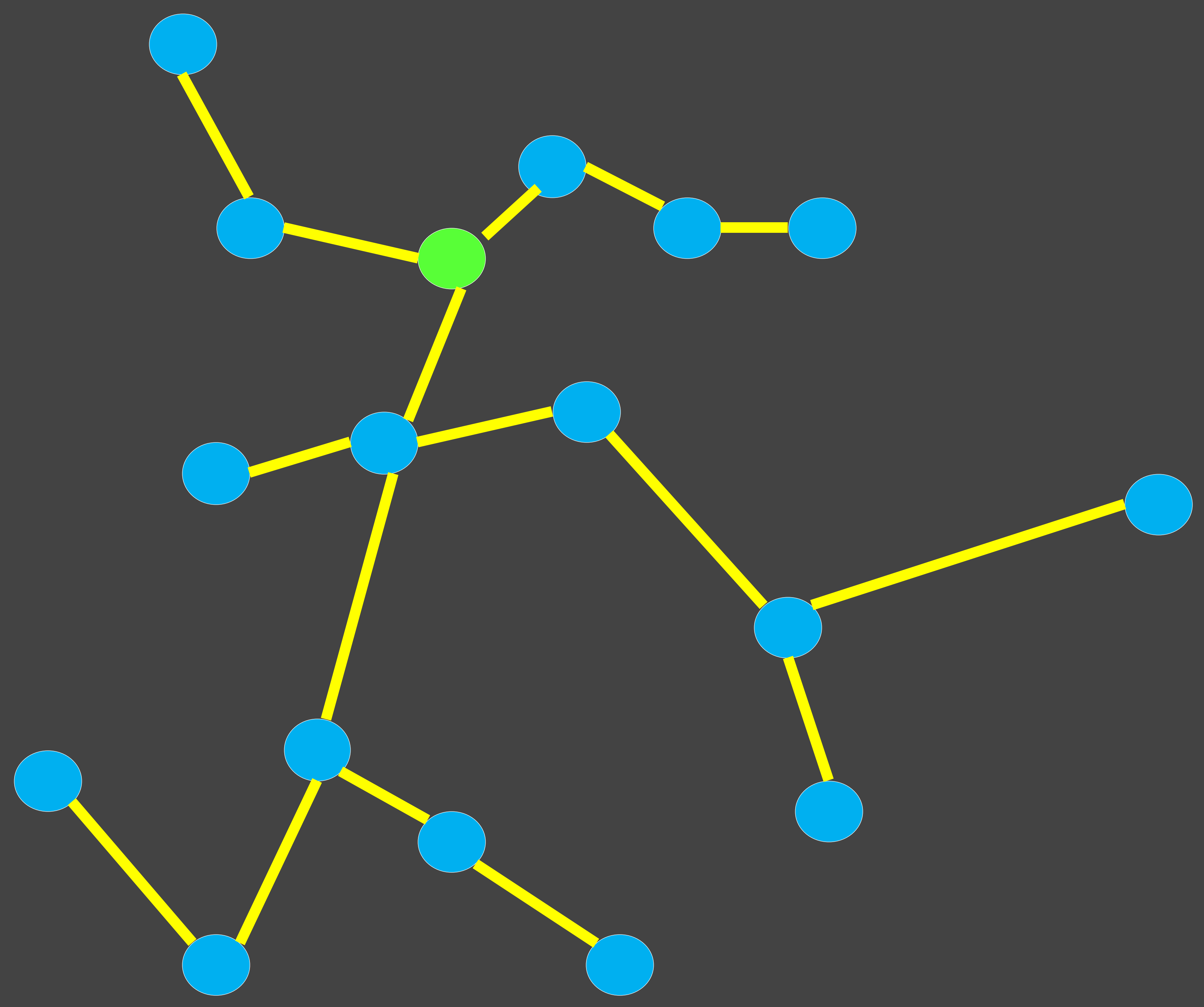
online (steiner) tree

connect v_t to $\arg\min_{s<t} d(v_s, v_t)$

Thm 1: greedy is $O(\log T)$ competitive

number of requests

works in unknown metric



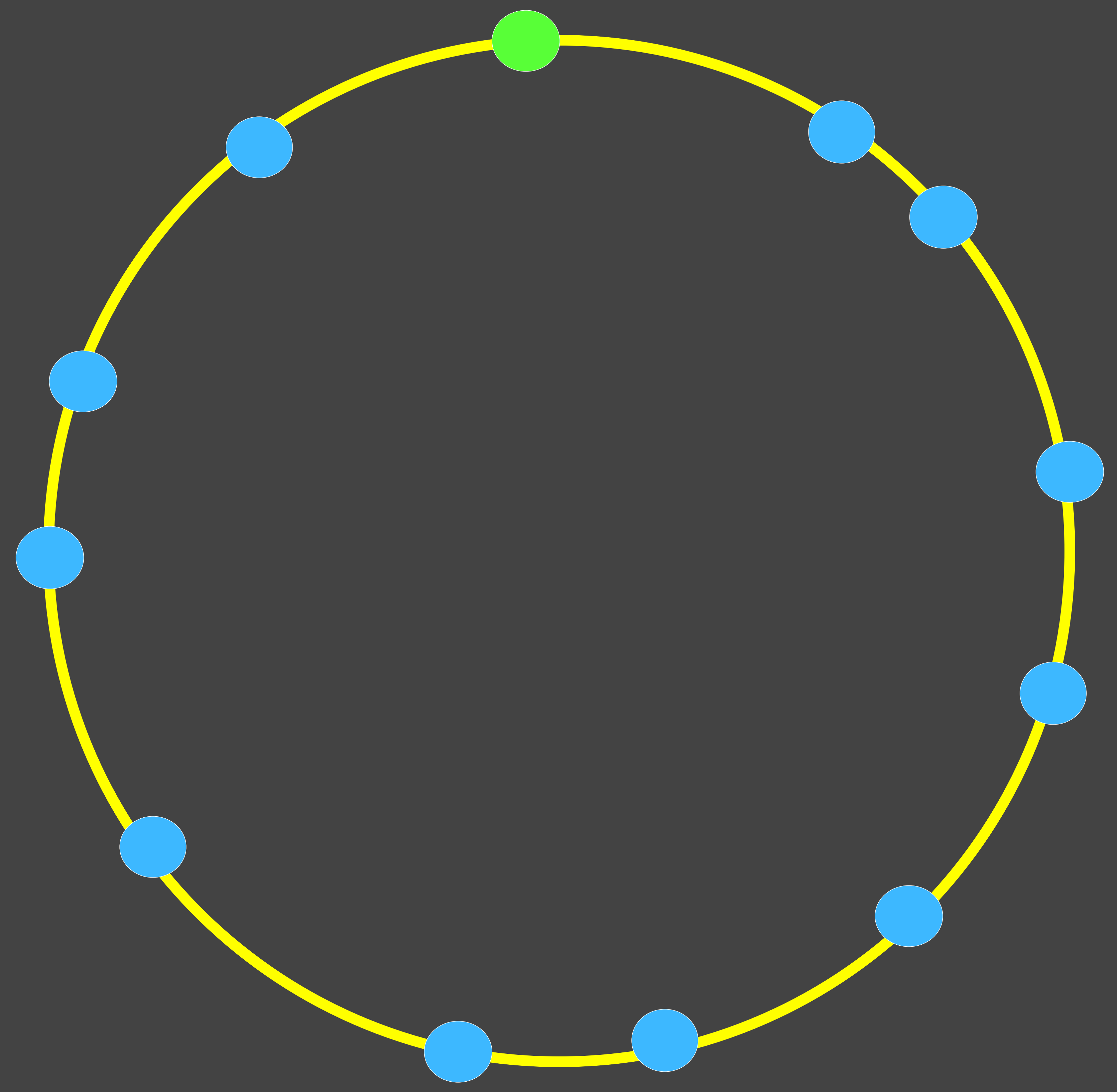
suppose this is OPT

[Imase Waxman 91]

online (steiner) tree

Thm 1: greedy is $O(\log T)$ competitive

number of requests



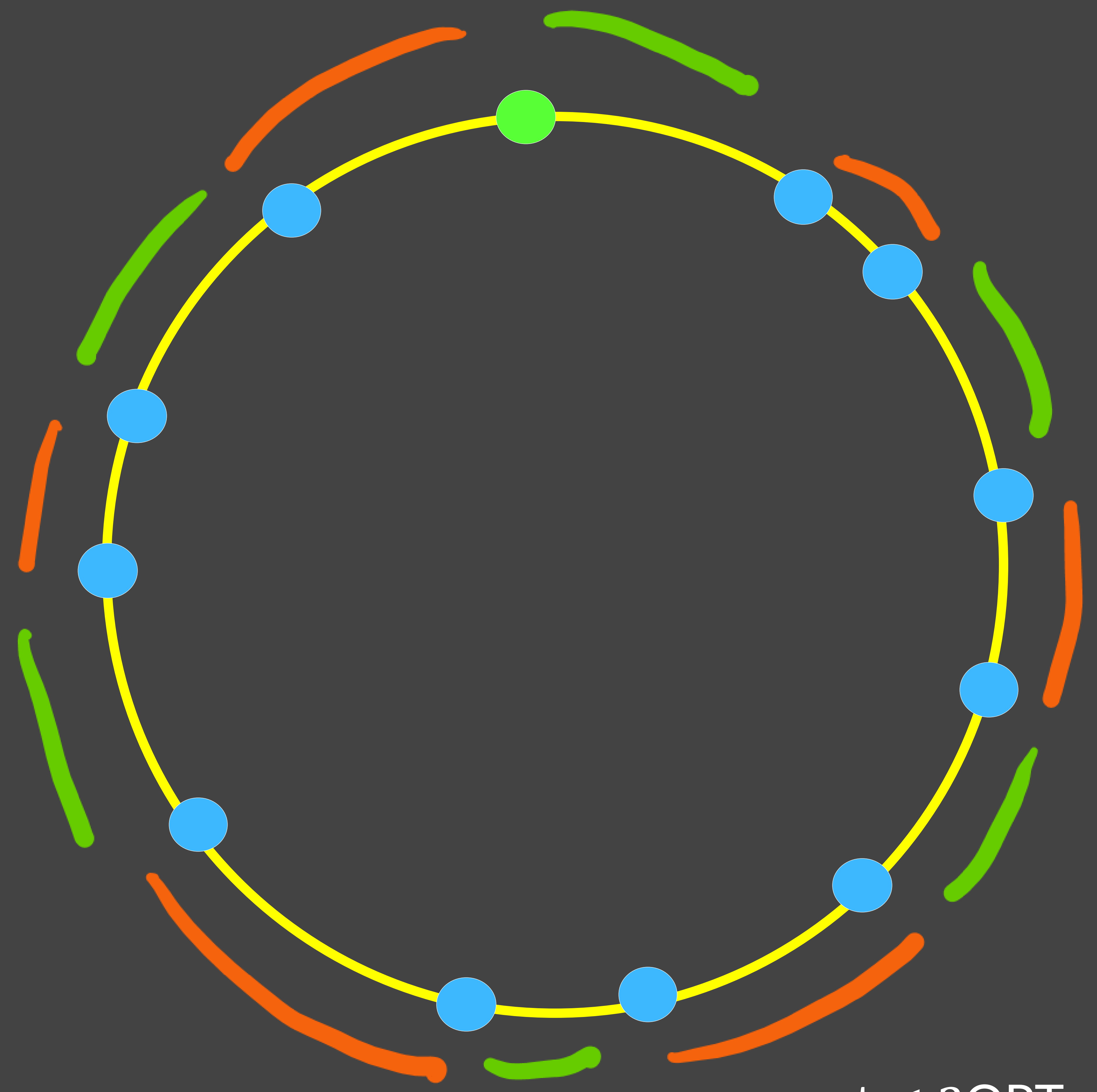
cost $\leq$ 2OPT

[Imase Waxman 91]

online (steiner) tree

Thm 1: greedy is $O(\log T)$ competitive

number of requests



cost $\lesssim$ 2OPT

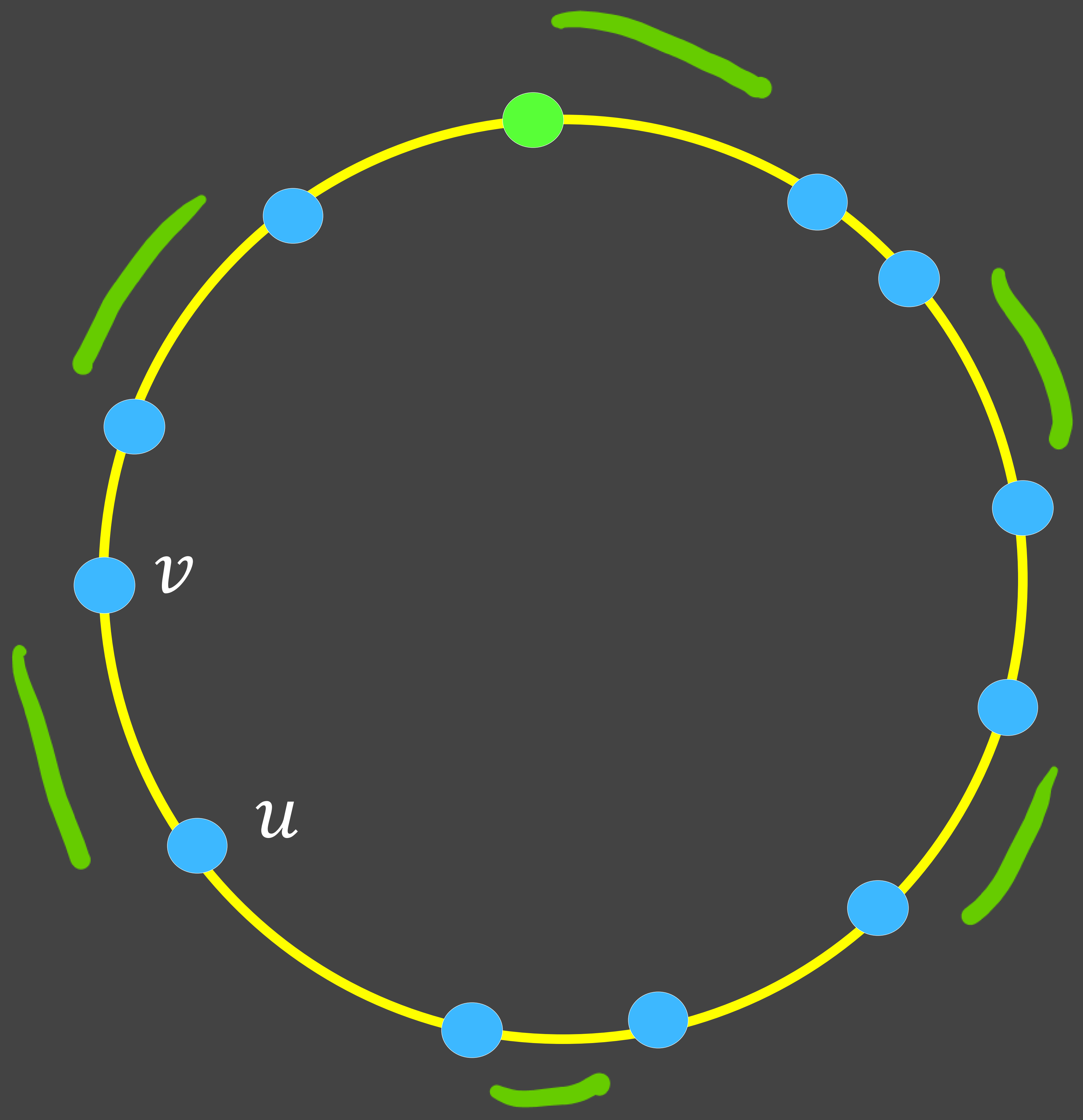
say, green cost $\leq$ OPT

[Imase Waxman 91]

online (steiner) tree

Thm 1: greedy is $O(\log T)$ competitive

number of requests



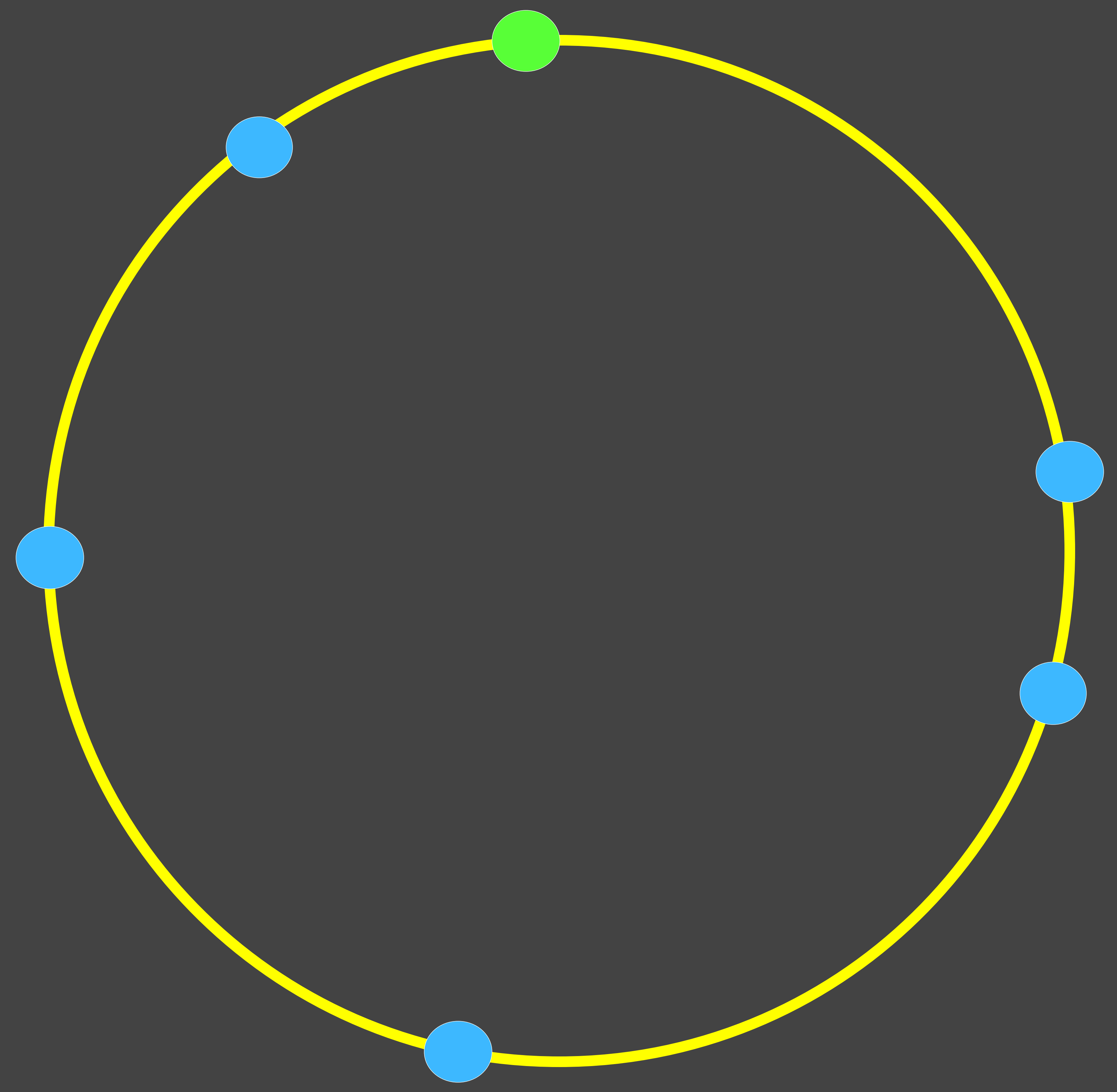
crucial observation:
total cost of these later requests $\leq OPT$

[Imase Waxman 91]

online (steiner) tree

Thm 1: greedy is $O(\log T)$ competitive

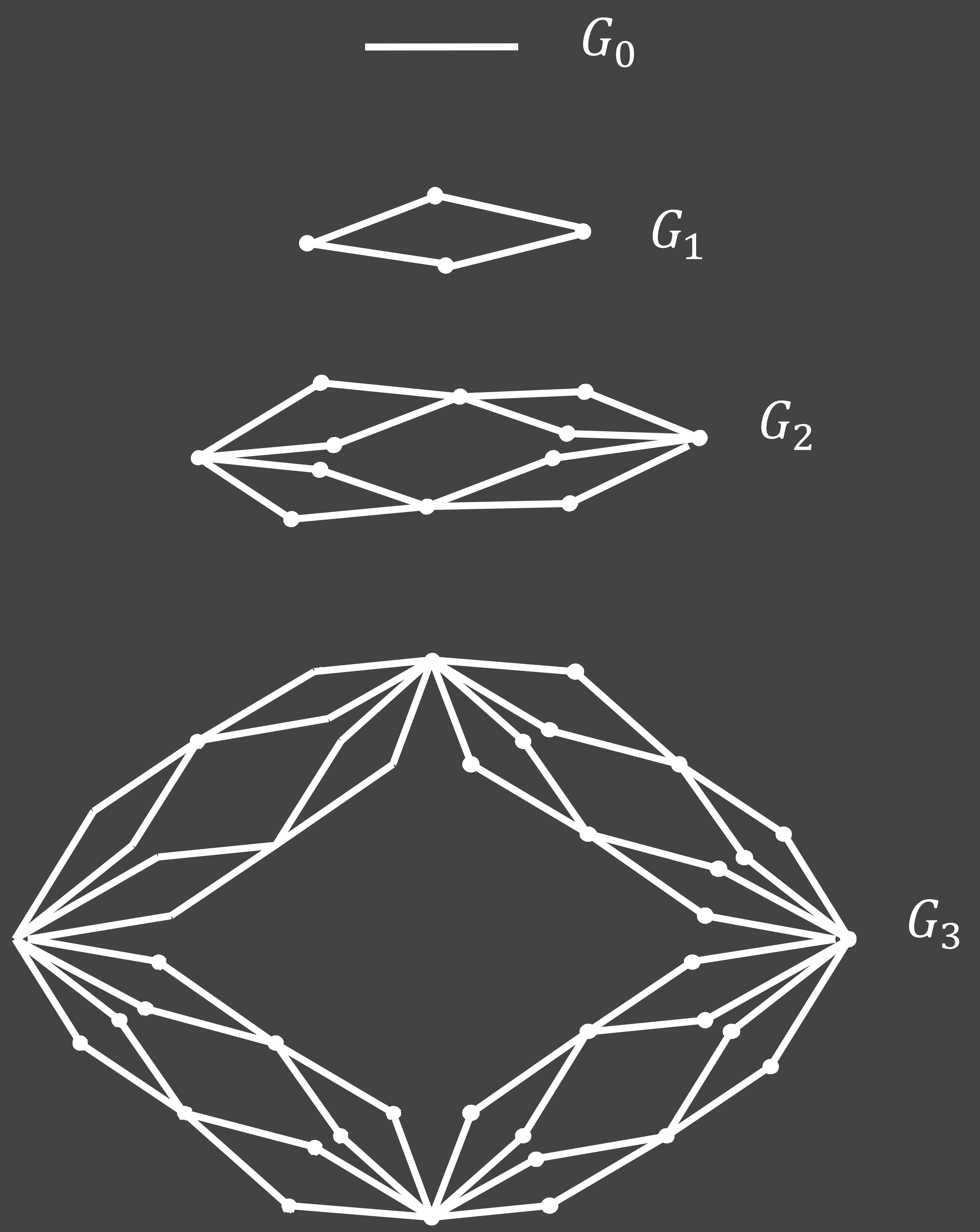
number of requests



Now recurse on other $T/2$ requests

[Imase Waxman 91]

online (steiner) tree



G_k: “diamond graph” or fractal of $K_{2,2}$

Thm 1: greedy is $O(\log T)$ competitive

Thm 2: no online algorithm can do better

[Imase Waxman 91]

roadmap for today

Steiner tree

Online algo (using greedy algo)

Theorem: $O(\log T)$-competitive using greedy algo

Some matching hardness results

Theorem: $\Omega(\log T)$ bound on diamond graphs

How to go beyond worst-case?

When requests from known distribution

Btw, approach #2

“Theorem”: Every n point metric space is “almost” a tree

in two senses:

approximation and **randomization**

Theorem:

Exists algo that takes any n point metric space $M = (V, d)$ and

outputs a random tree $T = (V, d)$ such that for all $x, y \in V$

a. $d_T(x, y) \geq d_M(x, y)$

b. $\mathbb{E}[\, d_T(x, y)\,] \leq \alpha\, d_M(x, y)$

where $\alpha = O(\log n)$

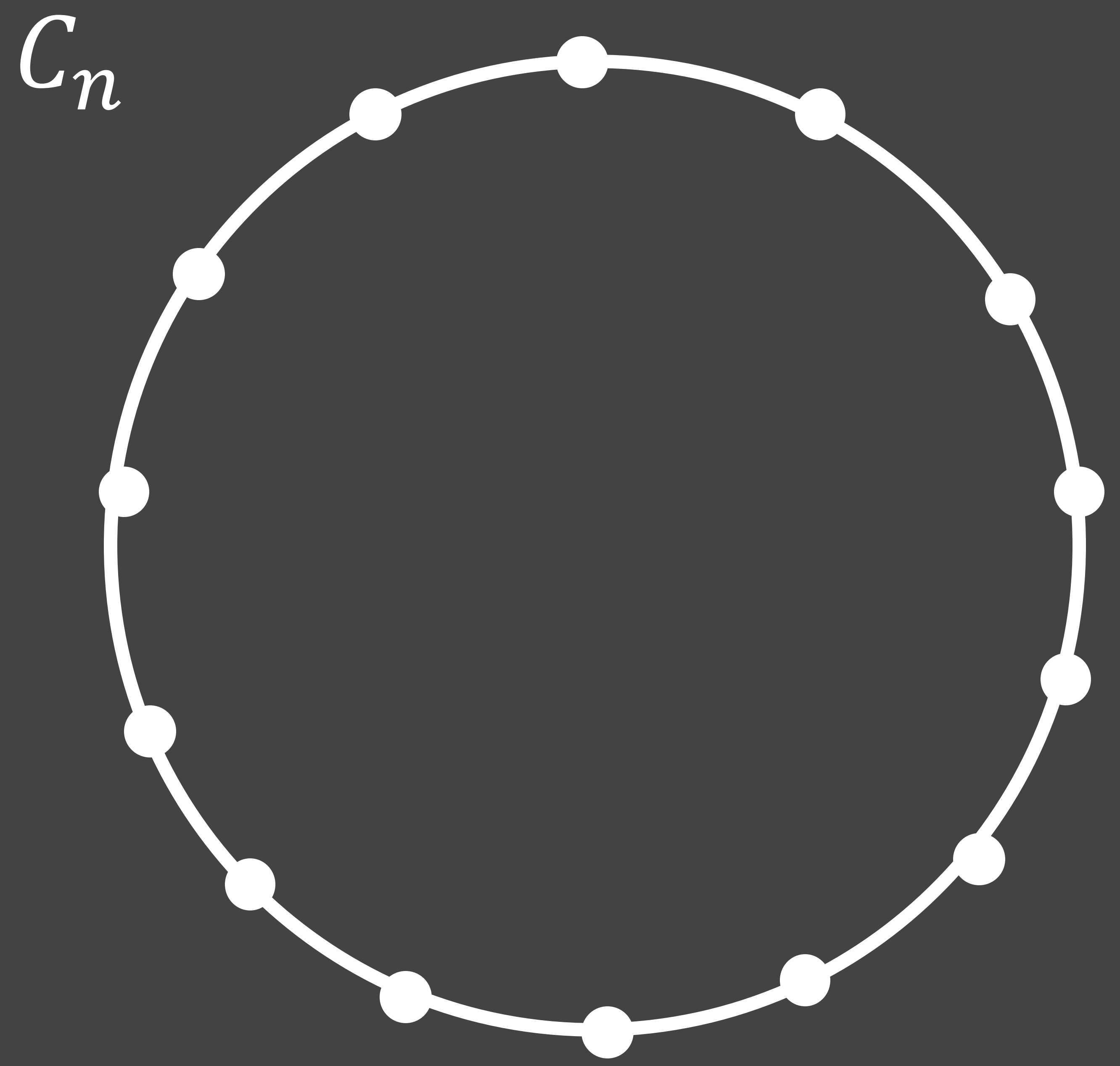
distances change

by only logarithmic factor

in expectation.

[Alon Karp Peleg West 94, Bartal 96, … , Fakcharoenphol Rao Talwar 04]

Evocative example:



[Alon Karp Peleg West 94, Bartal 96, … , Fakcharoenphol Rao Talwar 04]

Theorem:

Exists algo that takes any n point metric space $M = (V, d)$ and

outputs a random tree $T = (V, d)$ such that for all $x, y \in V$

a. $d_T(x, y) \geq d_M(x, y)$

b. $\mathbb{E}[\, d_T(x, y)\,] \leq \alpha\, d_M(x, y)$

where $\alpha = O(\log n)$

distances change

by only logarithmic factor

in expectation.

[Alon Karp Peleg West 94, Bartal 96, … , Fakcharoenphol Rao Talwar 04]

Algorithm #2

Underlying metric space, root vertex r

Sample a random tree T from the theorem

When request v_t comes, use unique path to r in T

Thm 1: algo is α-competitive (randomized)

Recall that $\alpha = O(\log n)$

only works in known metric!!

[Awerbuch Azar 96]

Algorithm #2

Underlying metric space, root vertex r

Sample a random tree T from the theorem

When request v_t comes, use unique path to r in T

Thm 1: algo is α-competitive (randomized)

Recall that $\alpha = O(\log n)$

Fact #1: $ALG_T = OPT_T$

Fact #2: $\mathbb{E}[cost(OPT_T)] \leq \alpha\, OPT_M$

Fact #3: $cost(ALG_M) \leq ALG_T$

$\mathbb{E}\, cost(ALG_M) \leq ALG_T = OPT_T \leq \alpha\, OPT_M$

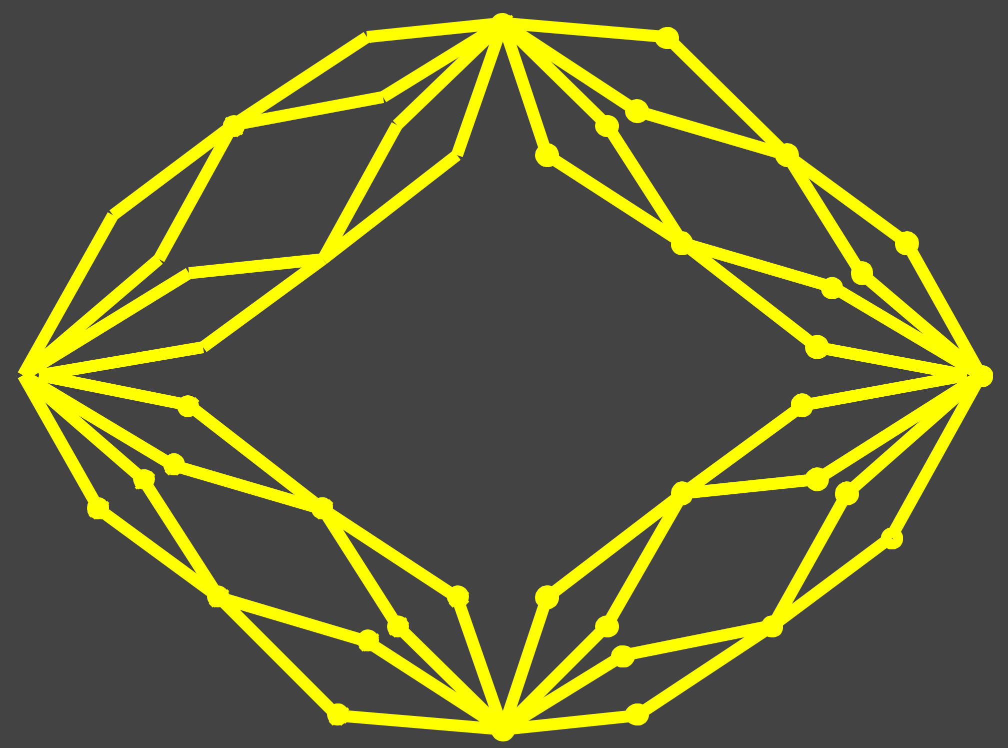
[Awerbuch Azar 96]

Algorithm #2

Underlying metric space, root vertex r

Sample a random tree T from the theorem

When request v_t comes, use unique path to r in T



Thm 1: algo is α-competitive (randomized)

Recall that $\alpha = O(\log n)$

Btw, lower bound shows $\Omega(\log T)$-competitive

On this example: $\log T = \Theta(\log n)$

$\Rightarrow$ embedding diamond graphs into random trees requires $\alpha = \Omega(\log n)$.

Theorem:

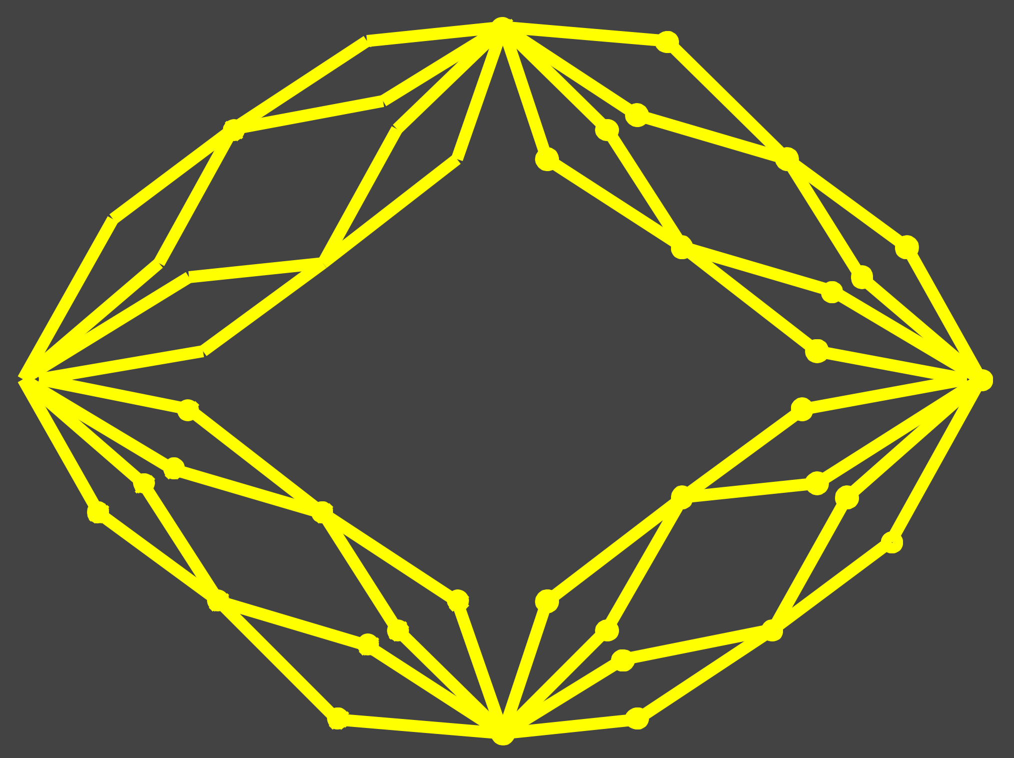
Exists algo that takes any n point metric space $M = (V, d)$ and

outputs a random tree $T = (V, d)$ such that for all $x, y \in V$

a. $d_T(x,y) \geq d_M(x,y)$

b. $\mathbb{E}[\, d_T(x,y)\,] \leq \alpha\, d_M(x,y)$

where $\alpha = O(\log n)$



⇒ gives matching lower bound of $\Omega(\log n)$ for α

Can we get a similar technique to work for unknown metric model?

"Online metric embeddings" (e.g., work by [Bartal Fandina Umboh 20])

[Alon Karp Peleg West 94, Bartal 96, … , Fakcharoenphol Rao Talwar 04]

“Theorem”: Every n point metric space is “almost” a tree

Gives randomized $O(\log n)$ competitive algo for Steiner tree

Approach useful for many network design problems as well!!

roadmap for today

Steiner tree

Online algo (using greedy algo)

Theorem: $O(\log T)$-competitive using greedy algo

Some matching hardness results

Theorem: $\Omega(\log T)$ bound on diamond graphs

Second Algorithm via tree embeddings

How to go beyond worst-case?

When requests from known distribution

Two-connected Network Design

Steiner Tree:

Requests from Known Distributions

stochastic (steiner) tree

Suppose n requests: vertex $R_i \sim \mathcal{D}_i$

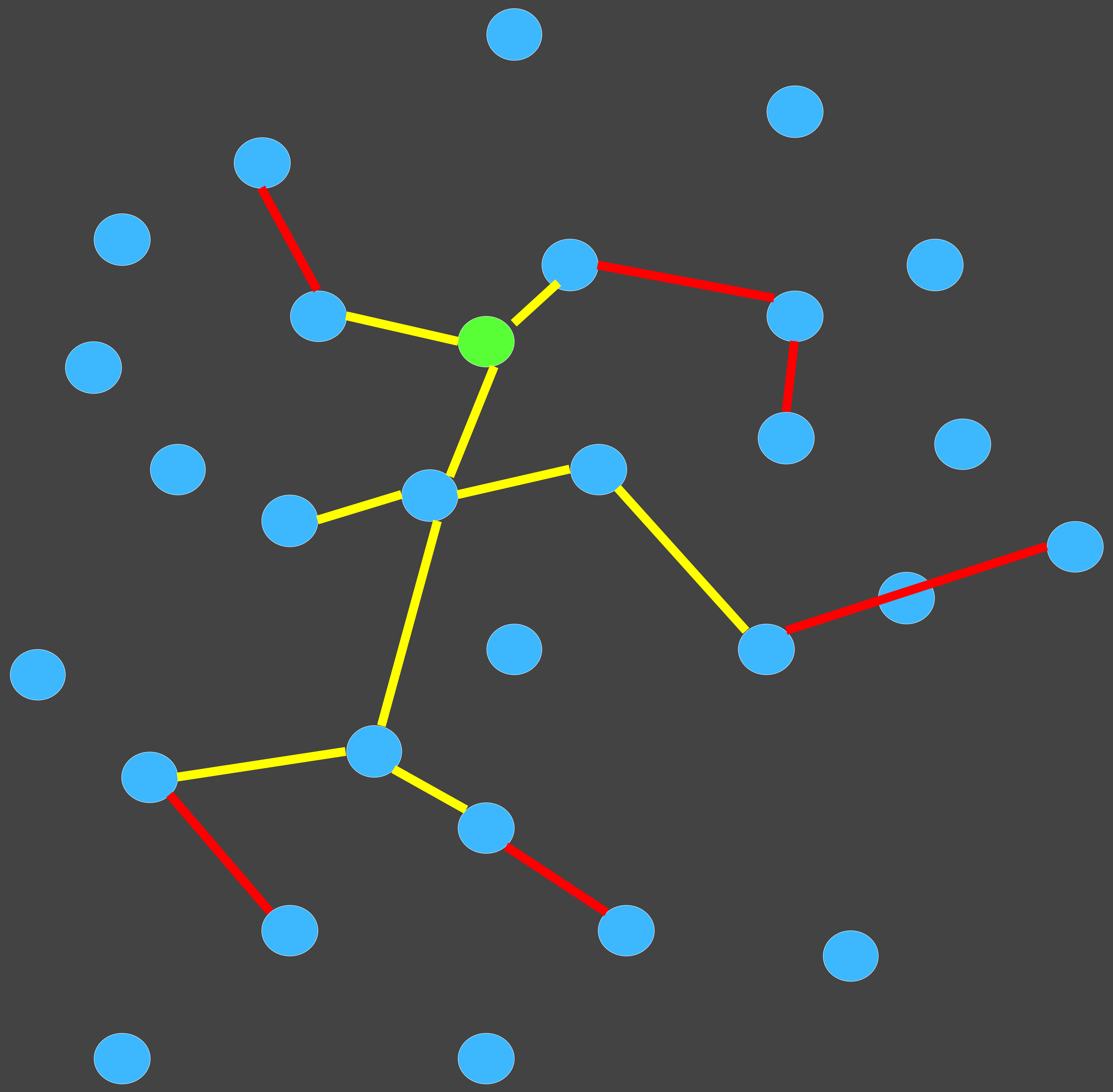
Connect each request on arrival

Algorithm:

For all i, take one sample $S_i \sim \mathcal{D}_i$ each

Build MST on $S_1, \ldots, S_n$

When actual requests $R_i \sim \mathcal{D}_i$ arrive:
connect to closest previous point



Goal: minimize total cost of edges

[Garg Gupta Leonardi Sankowski 08]

stochastic (steiner) tree

Suppose n requests: vertex $R_i \sim \mathcal{D}_i$

Connect each request on arrival

Theorem: $\mathbb{E}[Algo] \leq 2\, \mathbb{E}[MST(R_1, \dots, R_n)]$

Algorithm:

For all i, take one sample $S_i \sim \mathcal{D}_i$ each

Build MST on $S_1, \dots, S_n$

When actual requests $R_i \sim \mathcal{D}_i$ arrive:
connect to closest previous point

Proof: $\mathbb{E}[MST(S_1, \dots, S_n)] = \mathbb{E}[MST(R_1, \dots, R_n)]$

$$\begin{aligned}\mathbb{E}[cost(R_i)] &\leq \mathbb{E}[dist(R_i, S)] \\ &\leq \mathbb{E}[dist(R_i, S_{-i})] \\ &= \mathbb{E}[dist(S_i, S_{-i})]\end{aligned}$$

$$\Rightarrow\ \Sigma_i\, \mathbb{E}[cost(R_i)] \leq \Sigma_i\, \mathbb{E}[dist(S_i, S_{-i})] \leq \mathbb{E}[\, MST(S)\,]$$

roadmap for today

Steiner tree

Online algo (using greedy algo) — Theorem: $O(\log T)$-competitive using greedy algo

Some matching hardness results — Theorem: $\Omega(\log T)$ bound on diamond graphs

How to go beyond worst-case?

When requests from known distribution — Theorem: $O(1)$ bound for Stochastic inputs

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)