Online Algorithms using Samples

(a biased sample)

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**

[Graham 66, Sleator Tarjan 85]

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!

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

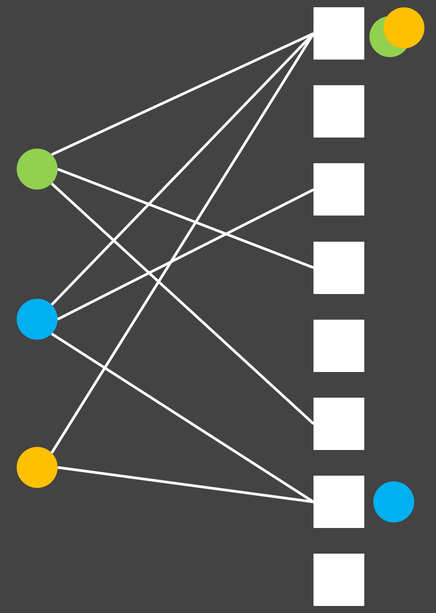
Want to minimize the competitive ratio.

[Azar Naor Rom FOCS 92]

Online Load Balancing

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

Goal: minimize maximum load of machines



[Awerbuch Azar Plotkin FOCS 93]

Online Packing-Covering

m resources

Each request i can be sat'd using any of K actions

Taking action k for request i:

consumes $P_{i,k} \in [0,1]^m$ of resources

gives $C_{i,k} \in [0,1]^m$ of benefit

Goal: want to get (almost-feasible) solution to the CPIP.

$$\begin{aligned} P\boldsymbol{x} &\leq B\mathbf{1} \\ C\boldsymbol{x} &\geq B\mathbf{1} \\ \boldsymbol{x} &\geq 0. \end{aligned}$$

the price of uncertainty

	Offline	Online
Steiner tree Facility location	$O(1)$	$\widetilde{\Theta}(\log n)$
Load Balancing (related m/c)	1	$\Theta(\log m)$
CPIPs	$(1+\varepsilon)$ if $B = \Omega(\log m/\varepsilon^2)$	$\Omega(\log m)$

Model: Online with a Sample

recent interest in using (machine learned) predictions

Question: where do predictions come from?

Answer #1: instances i.i.d. from some distribution $\mathcal{D}$

then use PAC learning to obtain predictions

Answer #2: online with a sample

Adversary fixes an instance I.

Today see p-sample S from I, see rest of instance tomorrow.

no distributional assumptions on the instance itself

[Campbell Samuels 1981]

[P. Azar Kleinberg Weinberg SODA 2014]

[Kaplan Naori Raz 2019]

A Learning Viewpoint

The sample S is "training data"

learn a good strategy using this training data

Use it on the "test data" $I \setminus S$.

Does the strategy generalize?

some results

	Offline	Online	OwaS	
Load Balancing (related m/c)	1	$\Theta(\log m)$	$(1+\varepsilon)$ if $OPT \gg poly(\log m, {}^1/_\varepsilon, {}^1/_p)$	[Argue Frieze G. Seiler Neurips 22]
CPIPs	$(1+\varepsilon)$ if $B = \Omega(\log m/\varepsilon^2)$	$\Omega(\log m)$	$(1+\varepsilon)$-feasible if $B \gg poly(\log m, {}^1/_\varepsilon, {}^1/_p)$	[G. Molinaro SODA 26]

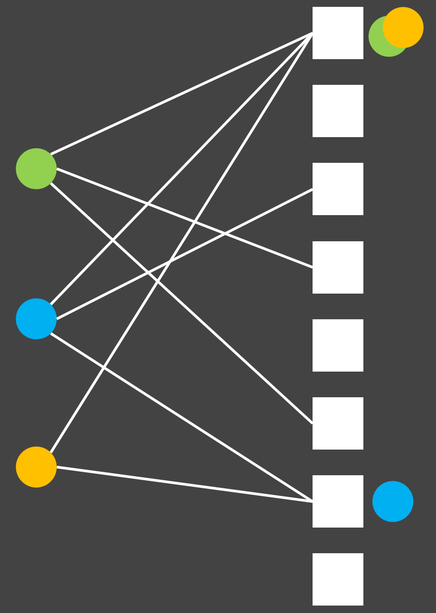
Load Balancing

load balancing (parallel machines, unit sizes)

m (identical) machines

job j arrives online

must assign job j to some "valid" machine



Load of machine i = number of jobs assigned to it

Goal: minimize maximum load over all machines

Thm 1: Greedy is $O(\log m)$ competitive algorithm.

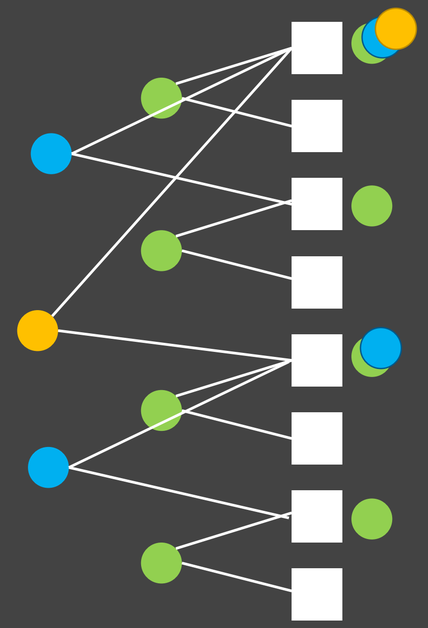
[Azar Naor Rom 92]

load balancing (parallel machines, unit sizes)

m (identical) machines

job j arrives online

must assign job j to some "valid" machine



Load of machine i = number of jobs assigned to it

Goal: minimize maximum load over all machines

Thm 1: Greedy is $O(\log m)$ competitive algorithm.

Thm 2: worst case $\Omega(\log m)$ lower bound.

[Azar Naor Rom 92]

load balancing with sample

Recall model: online with a sample

Tomorrow: instance I will arrive. Today: given epsilon (i.i.d.) sample from I.

Theorem: Can get load $(1+\varepsilon)\,OPT + {}^{1}/_{p}\,poly\left(\frac{\log m}{\varepsilon}\right)$ $\approx (1+\varepsilon)\,OPT$ if OPT is large

Using Uniform convergence: load $(1+\varepsilon)\,OPT + {}^{1}/_{p}\,poly\left(\frac{m}{\varepsilon}\right)$

explicitly learn a good strategy from samples!

Not possible with worst-case arrivals

[Argue Frieze G. Seiler Neurips 2022]

three algorithmic ideas

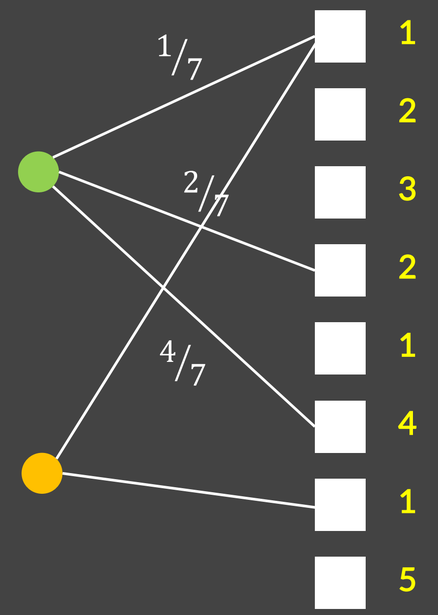
Fact 1: if OPT large, then integer assignment ≈ fractional assignment

Chernoff bound

Theorem 2: Exist machine "weights" so that proportional assignment is $OPT + \varepsilon$

Convex duality

Theorem 3: Exists algorithm to find machine weights using small sample



compute weights?

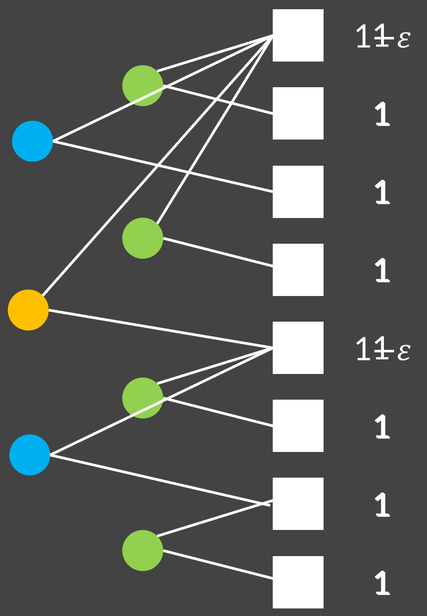
Theorem 3: Different algo to find machine weights $(\theta_1, \ldots, \theta_m)$ using small sample **(offline!!)**

Suppose we had entire instance!

Algorithm: start with all weights 1

Compute machine loads on instance, w.r.t current weights

if exist machines with load $\gg OPT$, reduce their weights by $(1-\varepsilon)$



N.b. guarantee good load,
not necc good soln to convex program

[Argue Frieze G. Seiler Neurips 2022]

compute weights?

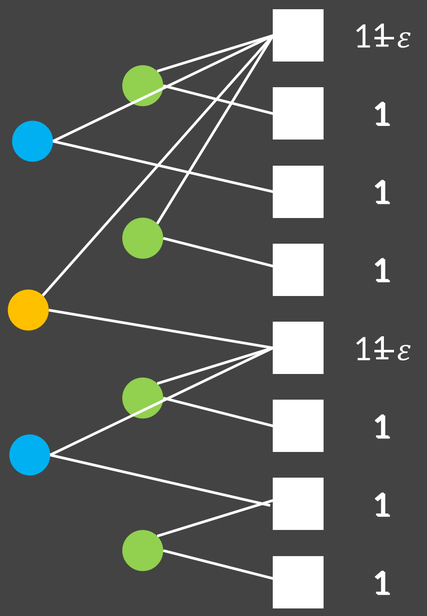
Theorem 3: Different algo to find machine weights $(\theta_1, \dots, \theta_m)$ using small sample **(offline!!)**

Suppose we had entire instance!

Algorithm: start with all weights 1

Compute machine loads on instance, w.r.t current weights

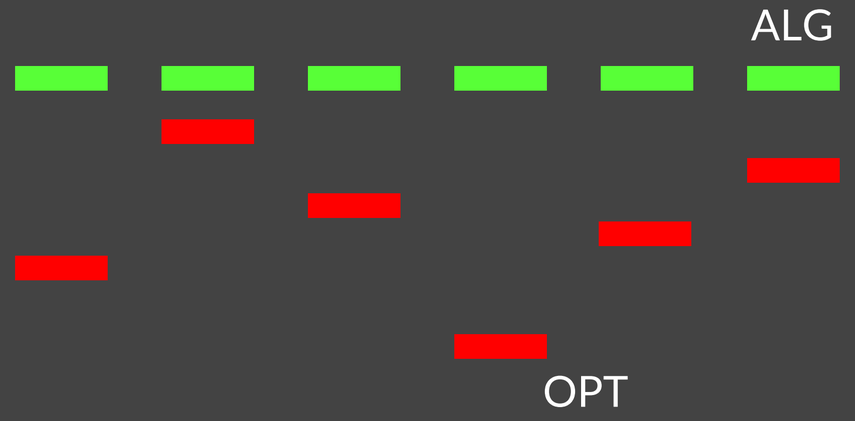
if exist machines with load $\gg OPT$, reduce their weights by $(1-\varepsilon)$



Does this terminate?	Yes, cute MW analysis!
How many rounds do we need?	$O\left(\frac{\log^2 m}{\varepsilon^3}\right)$ rounds!
But don't we need entire instance?!	Use samples to estimate loads!

[Argue Frieze G. Seiler Neurips 2022]

proof of termination



Lemma: algorithm terminates.

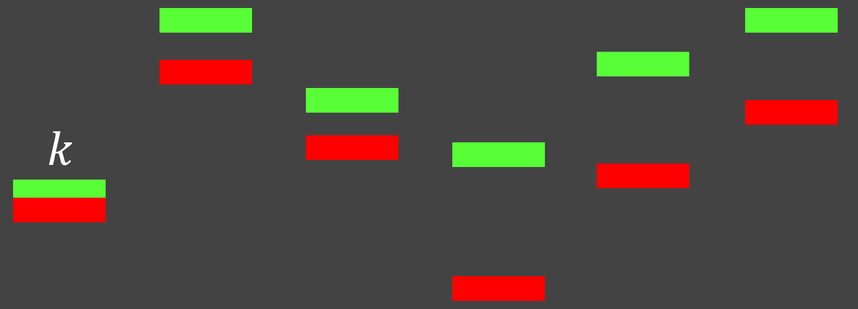
For simplicity: reduce if load > $2OPT$, reduce weights by ½

Let optimal weights θ^* be powers of 2

Algo is invariant to scaling, so imagine we start with equal weights $\geq \theta^*_{\max}$

[Lattanzi Lavastida Moseley Vassilvitskii SODA 20]

proof of termination



Lemma: algorithm terminates.

For simplicity: reduce if load > $2OPT$, reduce weights by ½

Let optimal weights θ^* be powers of 2

Algo is invariant to scaling, so imagine we start with equal weights $\geq \theta^*_{\max}$

Claim: θ_j never dips below θ^*_j for all j

Proof: Suppose some $\theta_k = \theta^*_k$, every other $\theta_j \geq \theta^*_j$.

By algo, k receives less load than in OPT. So will never reduce θ_k. ☺

So finitely many iterations.

[Lattanzi Lavastida Moseley Vassilvitskii SODA 20]

compute weights?

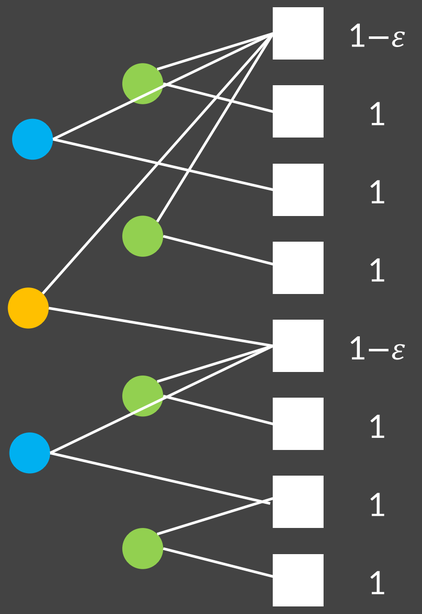
Theorem 3: Different algo to find machine weights $(\theta_1, \dots, \theta_m)$ using small sample **(offline!!)**

Suppose we had entire instance!

Algorithm: start with all weights 1

Compute machine loads on instance, w.r.t current weights

if exist machines with load $\gg OPT$, reduce their weights by $(1-\varepsilon)$



Does this terminate? Yes, cute MW analysis!

How many rounds do we need? $O\left(\frac{\log^2 m}{\varepsilon^3}\right)$ rounds!

But don't we need entire instance?! Use samples to estimate loads!

compute weights?

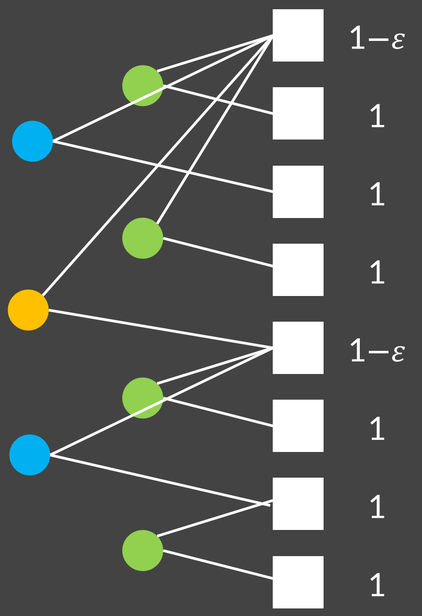
Theorem 3: Different algo to find machine weights $(\theta_1, \dots, \theta_m)$ using small sample **(offline!!)**

Suppose we had entire instance!

Algorithm: start with all weights 1

Compute machine loads on instance, w.r.t current weights

if exist machines with load $\gg OPT$, reduce their weights by $(1-\varepsilon)$



Does this terminate? Yes, cute MW analysis!

How many rounds do we need? $O\left(\frac{\log^2 m}{\varepsilon^3}\right)$ rounds!

But don't we need entire instance?! Use samples to estimate loads!

idea #3: estimate using the sample

Theorem 3: Algorithm to find machine weights $(\theta_1, \ldots, \theta_m)$ using small sample

Start with all weights 1

Break sample into $O\left(\frac{\log^2 m}{\varepsilon^3}\right)$ chunks

Estimate **machine loads w.r.t. current weights**, using next chunk

if exist machines with load $\gg OPT$, reduce their weights by $(1+\varepsilon)$

Use these weights on actual instance

…which completes the proof

Fact 1: if OPT large, then integer ≈ fractional assignment

Theorem 2: Exist machine weights so that proportional assignment is $OPT + \varepsilon$

Theorem 3: Algorithm to find machine weights using small sample

Theorem: load $(1 + \varepsilon)\, OPT + {}^{1}/_{p}\, poly\left(\frac{\log m}{\varepsilon}\right)$

Extending to Covering/Packing Problems

deeper connection to parallel algos

Consider parallel algo with form:

Local rounds: each agent does computation using private data

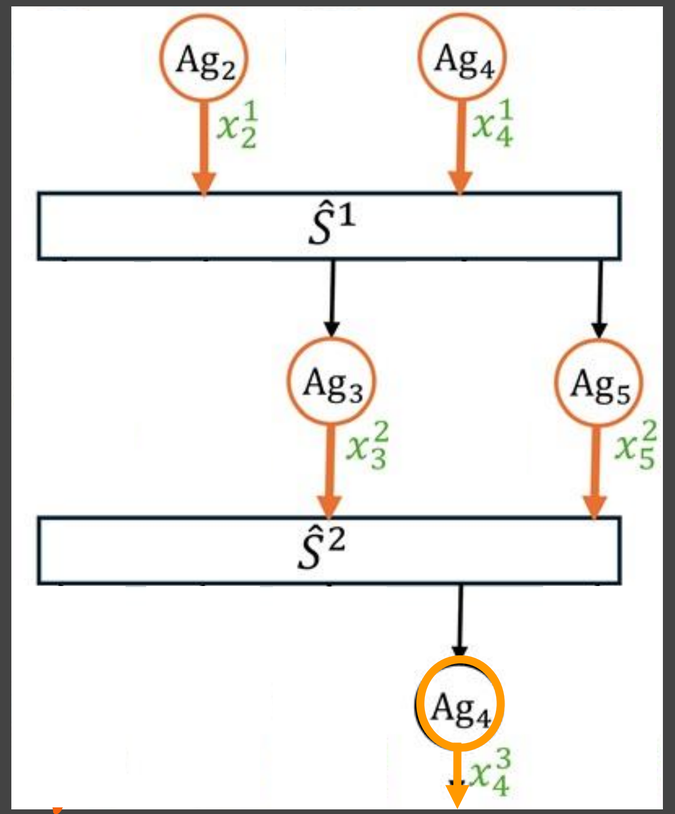
Global rounds: coordinator collects data from agents

L/G rounds alternate

"Few rounds": small number of rounds R

"Sample robust": coordinator can work
just using data of some random q fraction of agents

sample pR fraction of agents and simulate global coordinator



[G. Molinaro SODA 2026]

deeper connection to parallel algos

Consider parallel algo with form:

Local rounds: each agent does computation using private data

Global rounds: coordinator collects data from agents

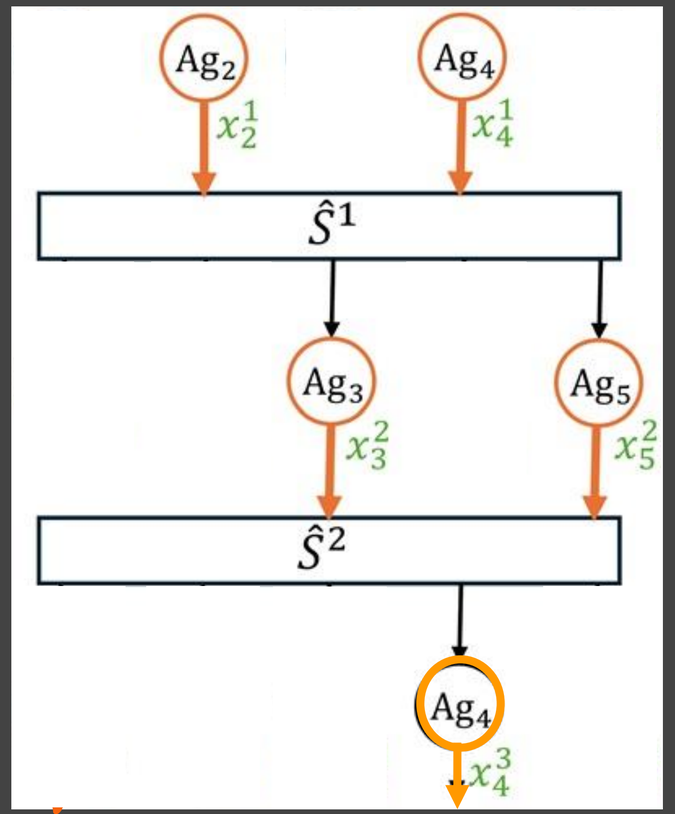
L/G rounds alternate

"Few rounds": small number of rounds R

"Sample robust": coordinator can work
just using data of some random q fraction of agents

sample pR fraction of agents and simulate global coordinator

Run algo for each other agent as they come in



[G. Molinaro SODA 2026]

Young's Parallel Algo for Packing/Covering

$$\begin{aligned} P\boldsymbol{x} &\leq B\mathbf{1} \\ C\boldsymbol{x} &\geq B\mathbf{1} \\ \boldsymbol{x} &\geq 0. \end{aligned}$$

$$f(\boldsymbol{x}) := \mathsf{smax}(P\boldsymbol{x} - B\mathbf{1}) \qquad \text{and} \qquad g_A(\boldsymbol{x}) := \mathsf{smin}((C\boldsymbol{x} - B\mathbf{1})_A),$$

$\exists j$ $\quad \nabla_j f(\boldsymbol{x}) \leq \nabla_j g_A(\boldsymbol{x}).$

raise this coordinate j

it covers faster than it consumes packing resources

our contribution: implement in "sample-robust" "local-global" framework

[Young FOCS 01, G. Molinaro SODA 2026]

Comparison of two “parallel-based” ideas

both approaches uses low-depth parallel algorithms

approach #1: use samples to learn good model parameters (machine weights)

approach #2: use samples to implement “good” global actions

each agent then replays the transcript to implement its actions

some results

	Offline	Online	OwaS	
Load Balancing (related m/c)	1	$\Theta(\log m)$	$(1+\varepsilon)$ if $OPT \gg poly(\log m, {}^1/_\varepsilon, {}^1/_p)$	[Argue Frieze G. Seiler Neurips 22]
CPIPs	$(1+\varepsilon)$ if $B = \Omega(\log m/\varepsilon^2)$	$\Omega(\log m)$	$(1+\varepsilon)$-feasible if $B \gg poly(\log m, {}^1/_\varepsilon, {}^1/_p)$	[G. Molinaro SODA 26]

online with a sample

Given a sample of the instance, explicitly learn good strategy

Model makes less stochastic assumptions than random order, or IID

Open Questions: Can we use samples to get better algos for other problems?

- other scheduling problems
- network design: e.g., Steiner forest, 2-connectivity
- optimal results for set cover

Thanks!