Approximation Algorithms for Optimization under Uncertainty

Anupam Gupta
Carnegie Mellon University

(Simons Uncertainty in Computation Workshop, Oct 7 2016)

the premise

Optimization problems are often defined on uncertain data.

e.g., data not yet available

have some predictions about inputs, actual data will arrive later

or, obtaining exact data is difficult/expensive/time-consuming

again, have predictions about all data,
based on it, we can ask for more precise values for some subset

…

the premise

Optimization problems are often defined on uncertain data.

Know-everything model:
deterministic algorithms

too optimistic?

Know-nothing-in-advance model:
online algorithms

online algorithms

Model: Instance is revealed slowly over time.

Need to make irrevocable decisions before the next arrival.

Measure of goodness: "competitive ratio"

$$\max_{I} \frac{\text{cost of our algorithm (instance } I)}{\text{cost of the best solution (instance } I)}$$

"compete with the best solution in hindsight"

[Sleator Tarjan 1985]

the premise

Optimization problems are often defined on uncertain data.

Know-everything model:
deterministic algorithms

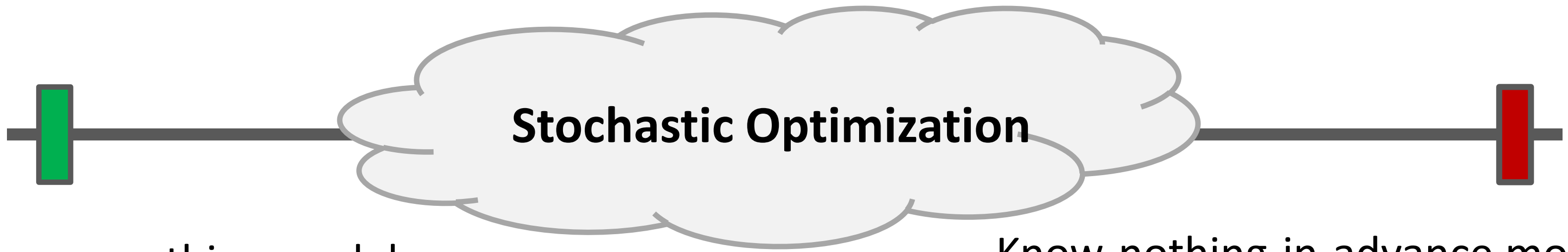
too optimistic?

Know-nothing-in-advance model:
online algorithms

too pessimistic?

the premise

Optimization problems are often defined on uncertain data.

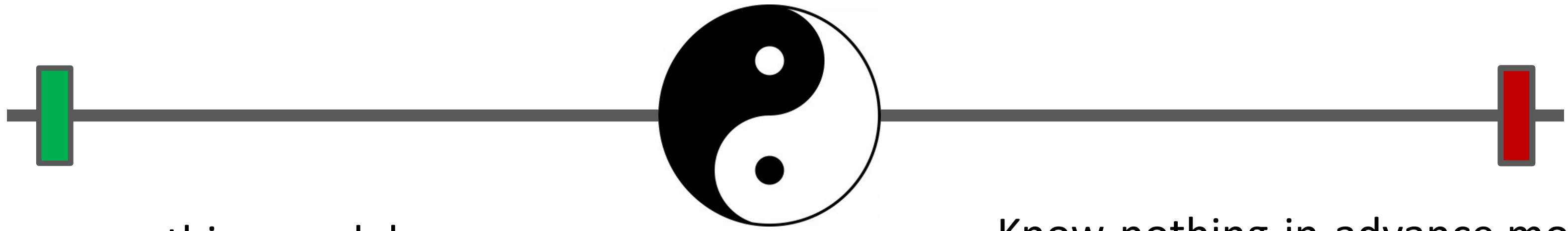


Know-everything model:
deterministic algorithms

Know-nothing-in-advance model:
online algorithms

the premise

Optimization problems are often defined on uncertain data.



Know-everything model:
deterministic algorithms

Know-nothing-in-advance model:
online algorithms

what kinds of problems?

1. want to pack items in a knapsack of bounded size
   but item sizes are random. What should we do?

2. want to build a network connecting customers
   each customer is an i.i.d. draw from a given prob.distrib.

3. want to find a large matching in a graph
   but each edge (when matched) fails with a certain probability

4. want to serve customers (one per timestep)
   but each waiting customer may quit with some probability

approx. algos. for stochastic optimization

Goal: design algorithms to make (near)-optimal decisions given some predictions (probability distribution on potential inputs).

Most of these problems are NP-hard (or worse)

So will give approximation algorithms for them.

(Still worst-case analysis, but inputs are distributions.)

Key Questions:

How to model uncertainty in the inputs?

How does the solution-space change?

How do the solution techniques (and analysis) change?

a sketch of a history…

Stochastic Optimization long-studied (~60 years)

Dantzig's paper on "Linear Programming under Uncertainty" in 1955.

Several textbooks, mainly from the OR perspective.

Lots of great heuristics: can we explain their effectiveness?

The approximation algorithms effort newer (since the 2000s)

some exceptions: stochastic scheduling, stochastic online paging, …

First approximation papers (~2003)

[Dye Stougie Tomasgaard], [Ravi Sinha], [Immorlica Karger Minkoff Mirrokni] [Dean Goemans Vondrak].

the plan for this talk

Stochastic Knapsack

the model, and solving it using basic LP techniques

Stochastic Steiner tree

how stochastic arrivals temper the pessimism of competitive analysis

Short takes:

Stochastic Matchings

Secretary Problems

Impatience

Two-stage problems

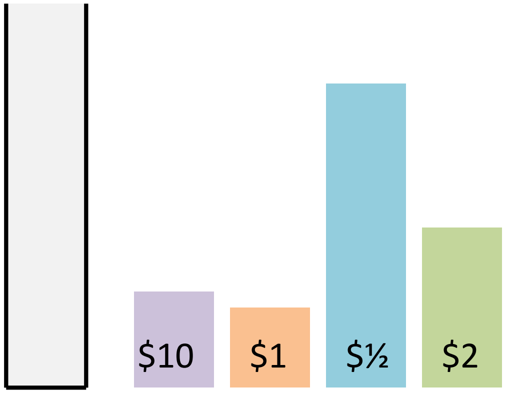
example I: knapsack

Input: a bag of size **B**

distribution on (size, reward) pairs

Output: set of objects that fit into bag,

maximize the reward



Stochastic Question:

sizes/rewards random, independent

only one operation allowed:

add an item to the bag,

then size/reward is revealed (drawn from the distribution)

stop when added item overflows bag.

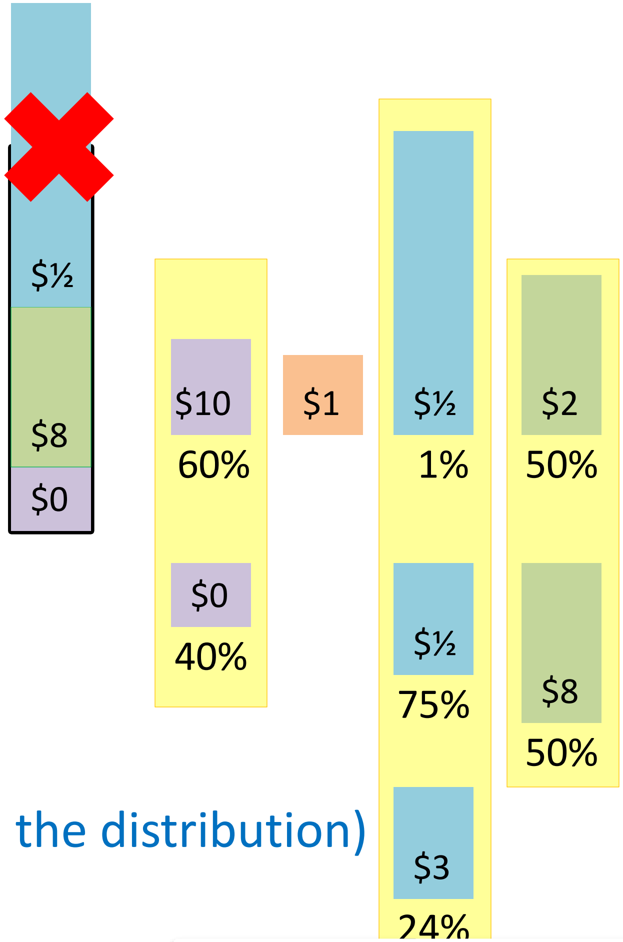
example I: knapsack

Input: a bag of size **B**

distribution on (size, reward) pairs

Output: set of objects that fit into bag,

maximize the reward



Stochastic Question:

sizes/rewards random, independent

only one operation allowed:

add an item to the bag,

then size/reward is revealed (drawn from the distribution)

stop when added item overflows bag.

algorithm "actively" causes uncertainty to be resolved

comparison to online algorithms

"online algorithms/competitive analysis" often too pessimistic!

Issue: competitive analysis compares

our algorithm's performance **(which cannot see the future)**

to an optimal algorithm that **sees the future perfectly**.

E.g., n identical items, taking size **2B** with probability 1-1/n, size 0 otherwise

Our algorithm: expected profit ~1/n

OPT: w.p. $\Omega(1)$, at least one small item exists, OPT gets at least its value.

comparison to online algorithms

"online algorithms/competitive analysis" often too pessimistic!

Issue: competitive analysis compares

our algorithm's performance **(which cannot see the future)**

to an optimal algorithm that **sees the future perfectly**.

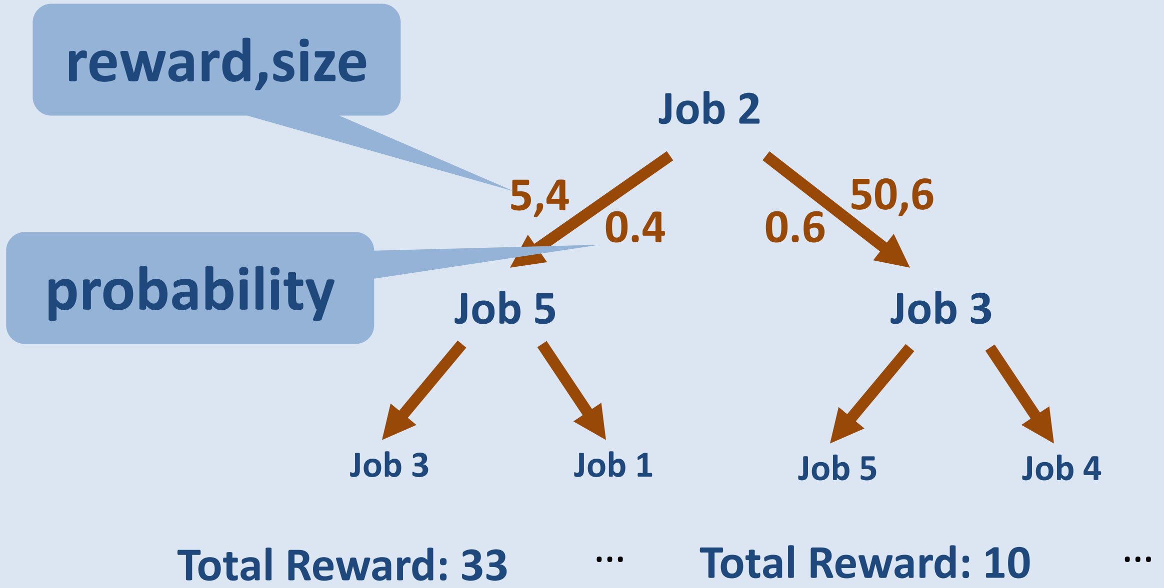
Stochastic Analysis: compare our algorithm's performance

to best possible algorithm with the same info.

i.e., compare our decision tree to the best decision tree.

Want to level the playing field…

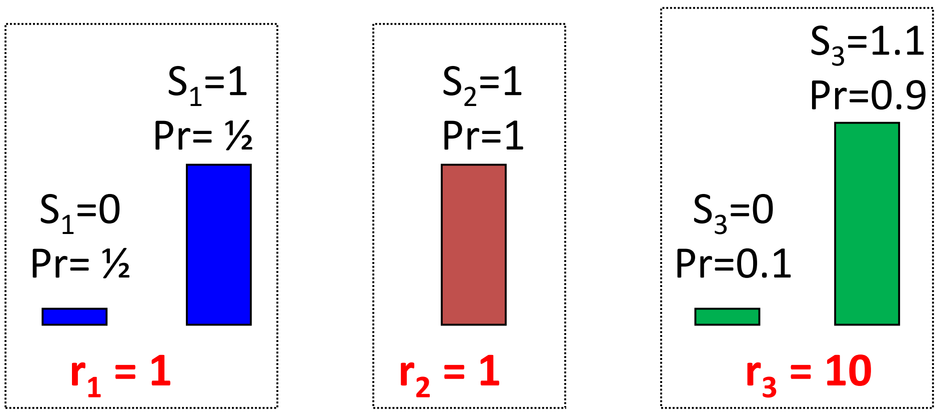
solution concept: decision tree



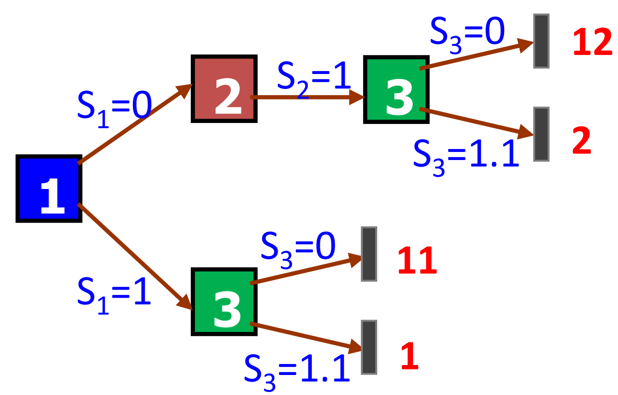
Quality: Expected total reward

optimal strategy (decision tree) may be exponential, also PSPACE hard.

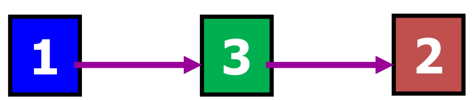
solution concept: decision tree



Budget B = 1

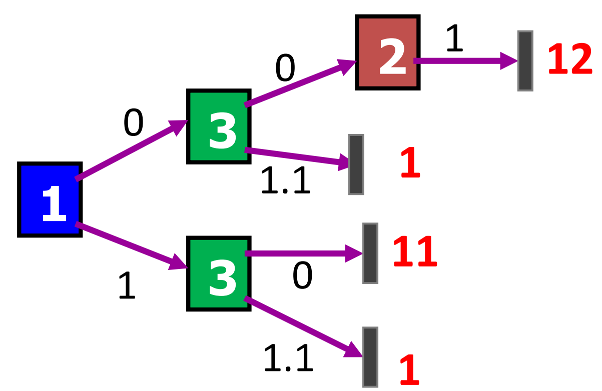


E [adaptive] = ½ * [0.1*12 + 0.9*2] + ½ * [0.1*11 + 0.9*1] = 2.5



E[non-adaptive] = 2.05

Adaptivity gap ≈ 1.25



how do we solve stochastic knapsack?

Assume for now: rewards are fixed, only sizes random.

Attempt #1: replace each job by its E[size].
Then run deterministic knapsack algorithm.

Problem: E[size] too sensitive a statistic.

Example: bin size B.
one item with size 0 wp 99%, B^2 wp 1%
another with size B wp 1.

Observe: if size more than B, does not matter if B+1 or B^2.

how do we solve stochastic knapsack?

Attempt #2: define the virtual size $\mu_k = E[\min(S_k, B^+)]$

virtual reward $\rho_k = r_k \Pr[S_k > B]$

Use deterministic algo. to find (approx) best set of jobs

w.p. ½ try these jobs in random order until we run out of space

w.p. ½ place the single best job

Theorem [Dean Goemans Vondrak 04]:

Expected reward is at least Ω(OPT).

OPT = reward of optimal decision tree.

how do we solve stochastic knapsack?

Attempt #2: define the virtual size $\mu_k = E[\min(S_k, B^+)]$

virtual reward $\rho_k = r_k \Pr[S_k > B]$

Use deterministic algo. to find (approx) best set of jobs

w.p. ½ try these jobs in random order until we run out of space.

w.p. ½ place the single best job

This is a (randomized) **non-adaptive** strategy. (simpler/faster/compact)

$\Rightarrow$ theorem bounds the "adaptivity gap"

$$\max_{\text{instance } I} \frac{\text{Best adaptive strategy on } I}{\text{Best non-adaptive strategy on } I}$$

an LP-based algorithm

$$\max \sum_i \mathsf{E}[r_i]\, x_i$$
$$\text{s.t} \sum_i \mathsf{E}[s_i]\, x_i \le B$$

This LP captures optimal strategy

Yes: up to a factor of 2 (this factor due to last item overflowing)

Rounding

Easy: basic LP solution has at most one fractional variable.

Interpreting solution as non-adaptive strategy

Not bad: scale down variables by ½.
Now violate budget with probability < ½ (by Markov's inequality).

To handle random rewards **and** sizes, need stronger LP. But similar idea.

[Dean Goemans Vondrak '05, G. Krishnaswamy Molinaro Ravi '11]

take-aways

Simple basic ideas:

Reduce stochastic problem to deterministic one.

Use a more robust statistic:

size = expected "truncated" means E[min(S_k, **B**)]

instead of just E[S_k]

Small Adaptivity gap:

Try to find a **non-adaptive** strategy comparable to best **adaptive** policy.

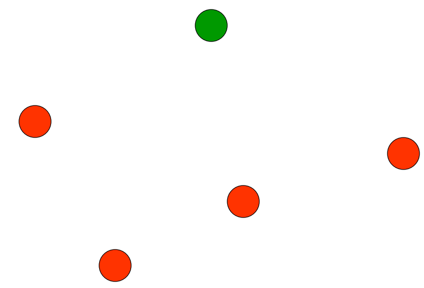
an extension: stochastic orienteering

Input: Metric space (V,d), start node s.

Each location has a job j

with random size S_j and (say, fixed) reward r_j

total time budget **B**.



Goal: Maximize expected total reward

subject to travel plus waiting ≤ B.

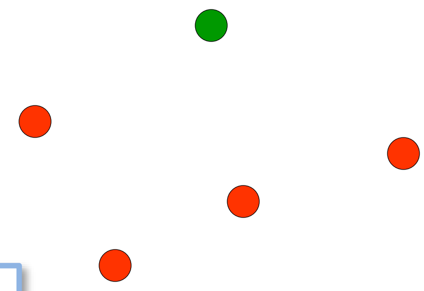
Theorem: [G. Krishnaswamy Nagarajan Ravi '12]

Gives non-adaptive strategy with adaptivity gap O(log log B).

Interestingly: [Bansal Nagarajan '14]

The adaptivity gap is $\Omega(\sqrt{\log \log B})$.

an extension: stochastic orienteering

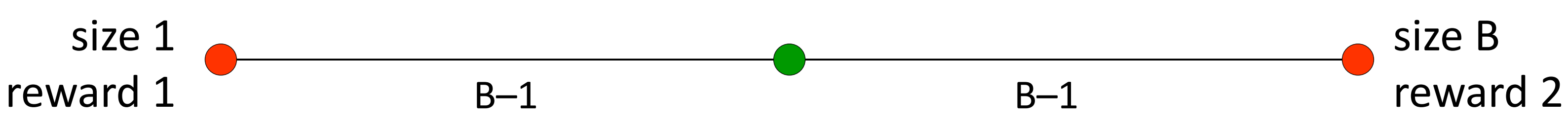


Attempt #1:

replace each job k by E[min(S_k, **B**)].

"B-truncated means"

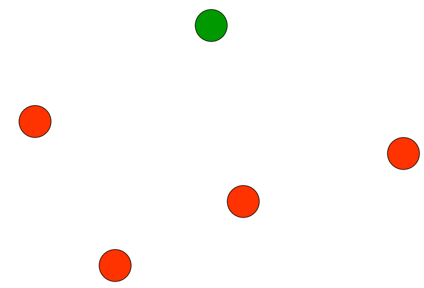
Bad example:



Moral:

Truncate item sizes at its distance from start point?

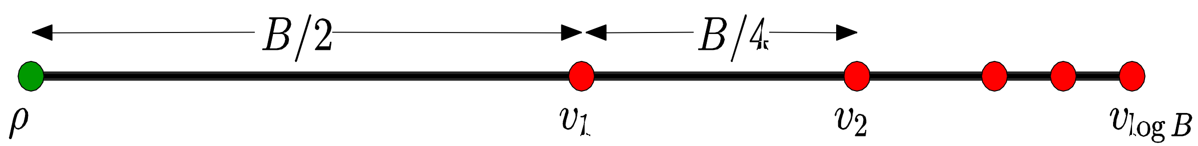
an extension: stochastic orienteering



Attempt #2:

use **(B – distance from start)**-truncated means

Still bad example: ☹



Each job has size (B – distance from start) wp. 1/(log B)

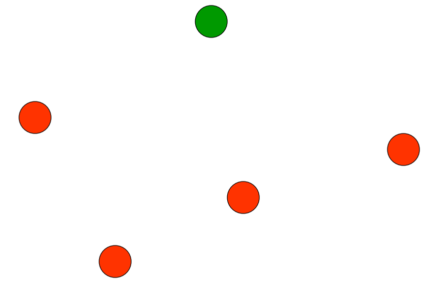
size 0 otherwise

W.p. Ω(1), all sizes are 0, can get all jobs.

But with truncated means, only pack in loglog B jobs.

⇒ gap of about Ω(log B/loglog B).

an extension: stochastic orienteering



Attempt #3:

"Guess" the ideal waiting time **W**, travel time **T = B – W**.

use **W**-truncated means.

Find best tour that travels **T**, and deterministically waits **W**.

Theorem: [G. Krishnaswamy Nagarajan Ravi '12]

Gives non-adaptive strategy with adaptivity gap $O(\log \log B)$.

Can't get a Constant Theorem: [Bansal Nagarajan '14]

The adaptivity gap is $\Omega(\sqrt{\log \log B})$.

take-aways

Simple basic ideas:

Reduce stochastic problem to deterministic one.

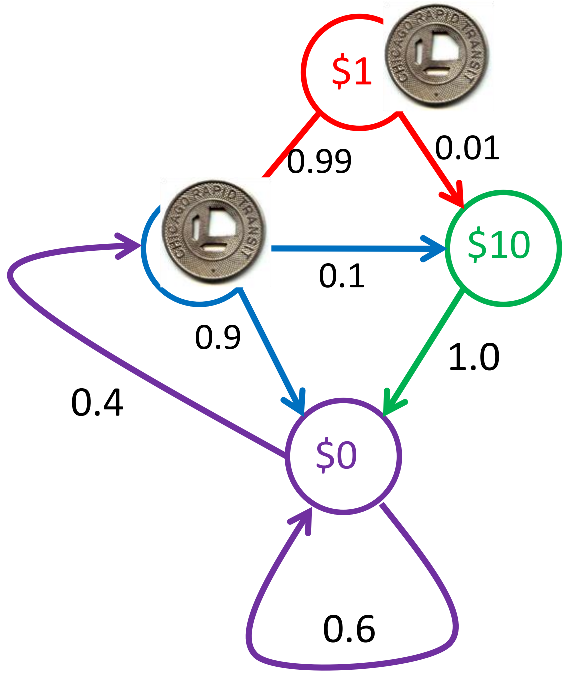
Might not be the most natural deterministic problem.

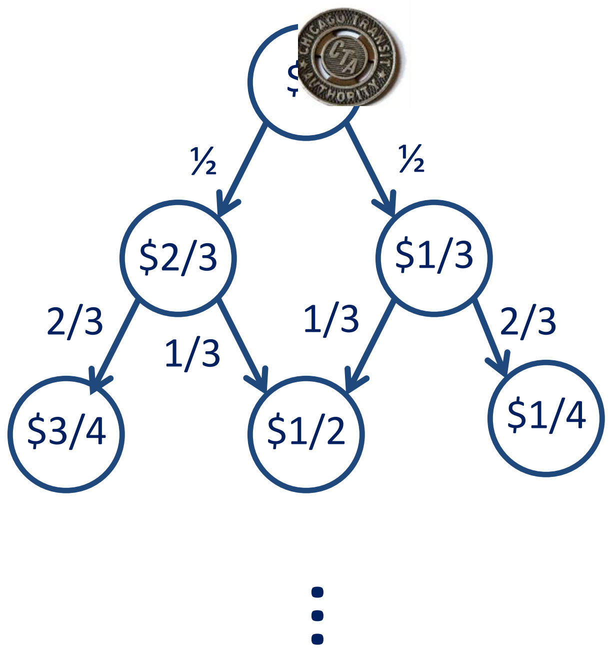
May need to use more robust statistic.

Small Adaptivity gap:

Found **non-adaptive** strategy comparable to best **adaptive** policy.

extension: playing bandits with a budget



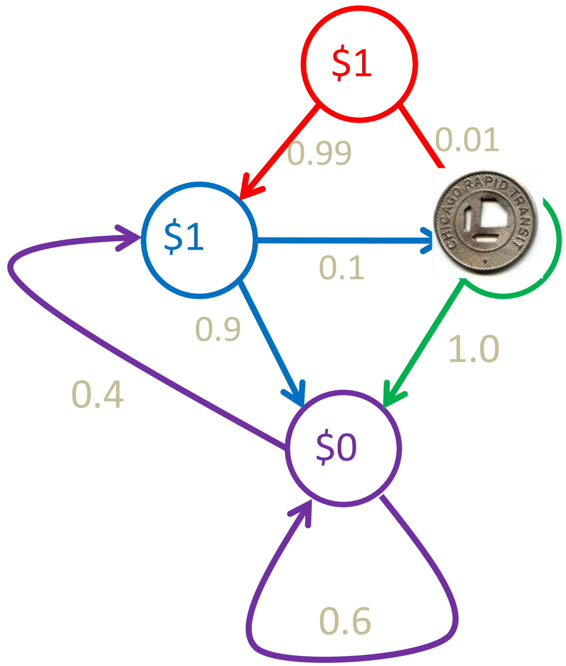


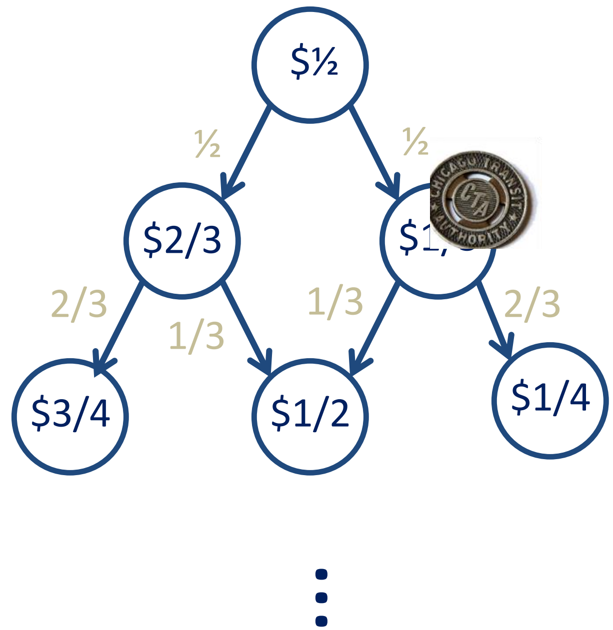
At each step, choose one of the Markov chains

that chain's token moves according to the probability distribution

Get the total payoff accrued over **B** steps

extension: playing bandits with a budget





Discounted rewards: the Gittins index is optimal [Gittins Jones 1974]

Fixed horizon:

O(1)-approx: [Guha Munagala] for "martingale rewards", non-adaptive

[G. Krishnaswamy Molinaro Ravi] for "non-martingale rewards", need adaptivity

active vs. passive uncertainty resolution

In above problems:

uncertainty was resolved by actions of the algorithm

algo chose what to learn (like "active learning")

Now, a different set of problems:

where information revelation process is independent of algo.

"same information revealed, no matter what our actions"

called "multi-stage stochastic optimization"

LINEAR PROGRAMMING UNDER UNCERTAINTY

GEORGE B. DANTZIG

The Rand Corporation, Santa Monica, Cal.

Summary

The essential character of the general models under consideration is that activities are divided into two or more stages. The quantities of activities in the first stage are the only ones that are required to be determined; those in the second (or later) stages can not be determined in advance since they depend on the earlier stages and the random or uncertain demands which occur on or before the latter stage. It is important to note that the set of activities are assumed to be *complete* in the sense that, whatever be the choice of activities in the earlier stages (consistent with the restrictions applicable to their stage), there is a possible choice of activities in the latter stages. In other words *it is not possible to get in a position where the programming problem admits of no solution.*

the **online** Steiner tree problem

Input: a metric space

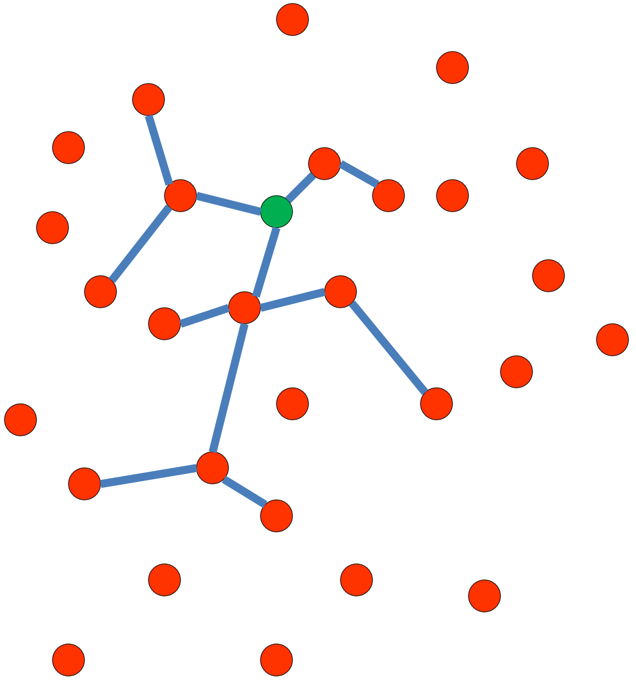
a root vertex r

a subset R of terminals

Output: a tree T connecting R to r
of minimum length/cost.

Fact: MST(R ∪ r) is a 2-approx.

Online: One terminal appears at @ each step,
must be immediately/irrevocably
connected to root.

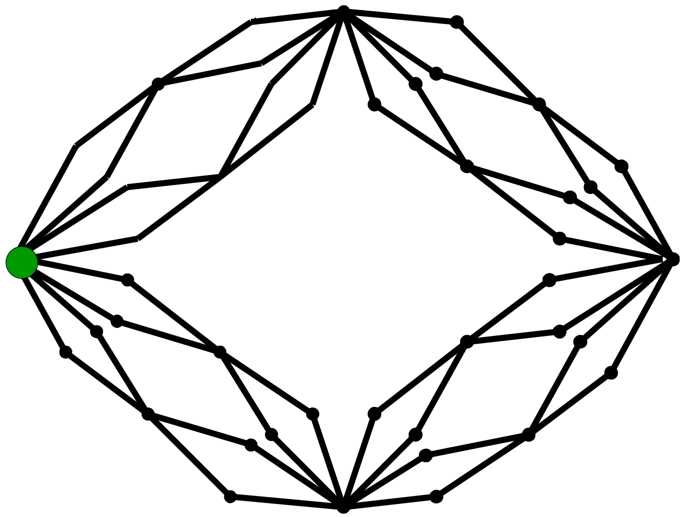


online greedy algorithm

[Imase Waxman '91]

the greedy algorithm is O(log k) competitive for sequences of length k.

and this is tight.



do better if input i.i.d?

Stochastic model can interpolate between offline and online.

Suppose the requested terminals are i.i.d. uniform vertices of the graph?

We want to get small (expected) competitive ratios.

$$\frac{\mathbf{E}_{\sigma,A}[\text{ cost of algorithm A on } \sigma \text{ }]}{\mathbf{E}_{\sigma}[\text{ OPT(set } \sigma) \text{ }]}$$

online Steiner tree : i.i.d.

Suppose demands are i.i.d. uniform nodes in V

Assume for this talk: know the length **k** of the sequence

Algorithm:

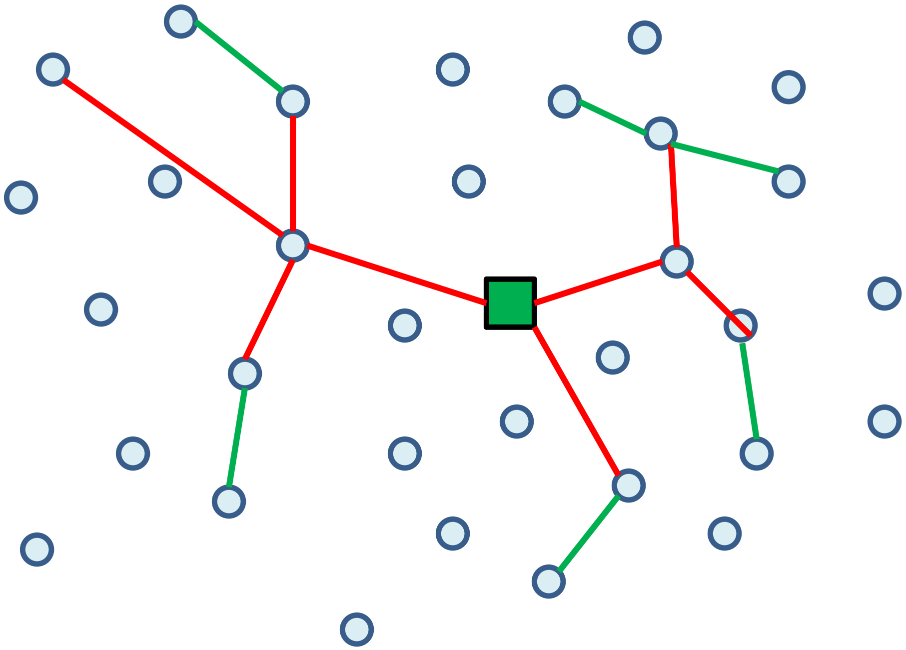
1. Sample **k** nodes from V, build a MST on sample + root.
2. When the **k** actual demands come, extend greedily.

Theorem:

$$\frac{\mathbf{E}_{\sigma,A}\text{[cost of algorithm A on } \sigma \text{]}}{\mathbf{E}_{\sigma}\text{[OPT(set } \sigma\text{)]}} \leq 4$$

[Garg G. Leonardi Sankowski 09]

augmented greedy



online Steiner tree : i.i.d.

Suppose demands are i.i.d. uniform nodes in V

Assume for this talk: know the length **k** of the sequence

sample ~ actual demands
so E[cost MST] ≤ 2E[OPT]

Algorithm:

1. Sample **k** nodes from V, build a MST on sample + root.
2. When the **k** actual demands come, extend greedily.

The

E[cost of single new demand]

= distance of random point from **k** random points

≤ distance of random point from **(k-1)** random points

≤ (1/k) * E[cost of MST on **k** random points]

take-aways

Stochastic arrivals soften the online (competitive ratio) model

Or is it a multi-stage (harder) version of the offline Steiner tree problem?

i.i.d. model used for network design problems, matchings, …

Are i.i.d. arrivals the “right” model? Probably not.

Alternatives:

E.g. arrivals i.i.d. with unknown distributions (how to “learn” distrib.?)

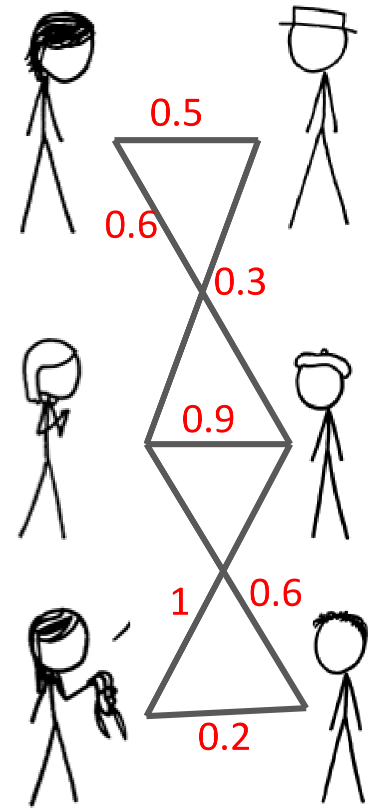
or from Markov chains of small complexity

or sequences with “enough entropy” in each request

vignettes I: matchings

Given a "template graph" with probabilities on edges

Actual graph: sample edges independently

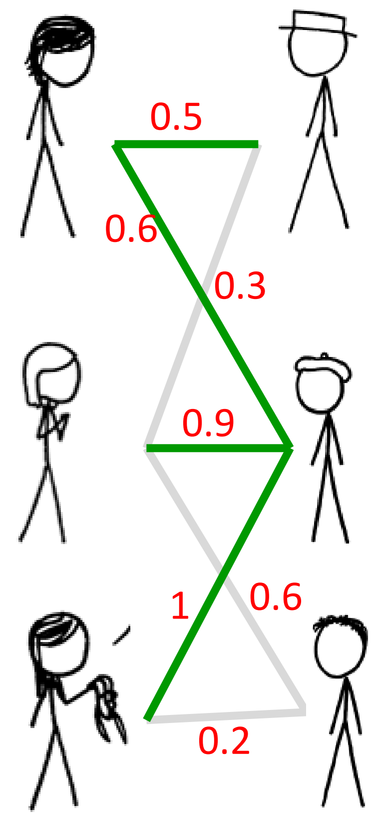


[Chen Immorlica Karlin Mahdian Rudra, Bansal+, Adamczyk…]

vignettes I: matchings

Given a “template graph” with probabilities on edges

Actual graph: sample edges independently



[Chen Immorlica Karlin Mahdian Rudra, Bansal+, Adamczyk…]

vignettes I: matchings

Given a "template graph" with probabilities on edges

Actual graph: sample edges independently

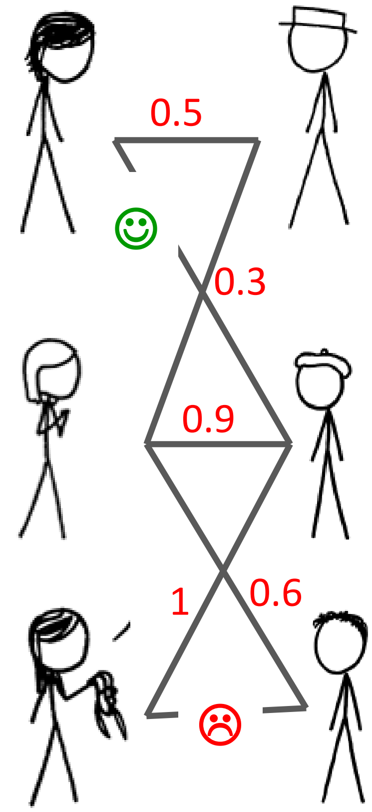
We don't see actual graph except by querying edges

Goal: Query few edges to find a large matching

Side-constraints:

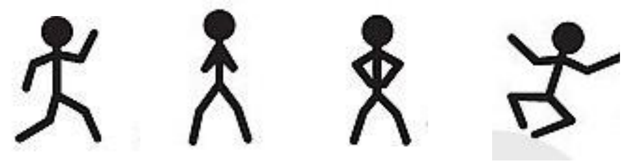
"query-commit", degree bounds, budgets

Arises in dating, kidney exchange, ad auctions



[Chen Immorlica Karlin Mahdian Rudra, Bansal+, Adamczyk…]

vignettes II: impatience



Each round: you pick one

others leave randomly…

Maximize expected value.

$12	$10	$15	$8
p = 1	p = 0	p = ¼	p = ½
X		Y	
	Y		X

departure probability

Surprisingly hard to beat (1-1/e) using simple heuristics; LPs give 70%.

How well can we solve this problem?

[Cygan Englert G. Mucha Sankowski]

vignettes III: random permutation model

An online model: the input **set S** is chosen by the adversary

but the actual **sequence σ** = **set S** in random order.

E.g., want to solve a packing LP where variables (& their coeff.s) revealed online

$$\max \quad \sum_i v_i x_i$$

$$\text{s.t.} \quad \sum_i x_i \leq 1, \qquad \text{and} \qquad x_i \geq 0$$

Aha! “secretary” problem if variables revealed in random order.

What if general packing linear problem?

Use “multiplicative weights” to combine multiple constraints into one.

Get optimal results this way. [Vöcking+][Molinaro, G.] [Agarwal Devanur]…

Q. Do we need random order? Cf. fast pagerank computation [Bahmani+]

to wrap up…

Stochastic problems arise in many different contexts
often interpolate between offline and online settings

Often get algorithms by relating to “right” deterministic variants
Often small adaptivity gaps

Connects to rich body of work in OR, control theory,
stochastic processes, Bayesian mechanism design…

Worst-case analysis viewpoint in stochastic optimization
leads to new problems/ideas.
lots of open directions here.