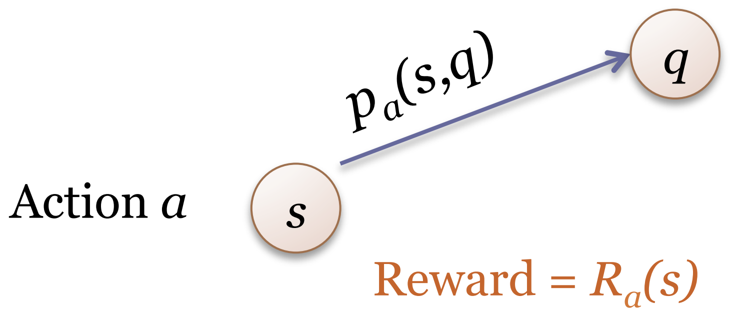
Approximation Algorithms for Stochastic Optimization

Kamesh Munagala
Duke University

Markov Decision Process

- Set S of states of the system
- Set A of actions

- If action a taken in state s:
  - Reward $R_a(s)$
  - System transitions to state q with probability $p_a(s,q)$



Markov Decision Process

- Set S of states of the system
- Set A of actions

- If action a taken in state s:
  - Reward $R_a(s)$ drawn from known distributions
  - System transitions to state q with probability $p_a(s,q)$

- **Input:**
  - Rewards and state transition matrices for each action
  - Start state s
  - Time horizon T

Policy for an MDP

- Maximize expected reward over T steps
  - Expectation over stochastic nature of rewards and state transitions
- **Policy:** Mapping from states S to actions A
  - Specifies optimal action for each observed state
- Dynamic Programming **[Bellman '54]**
  - Optimal policy computable in time $poly(|S|,|A|,T)$

This talk

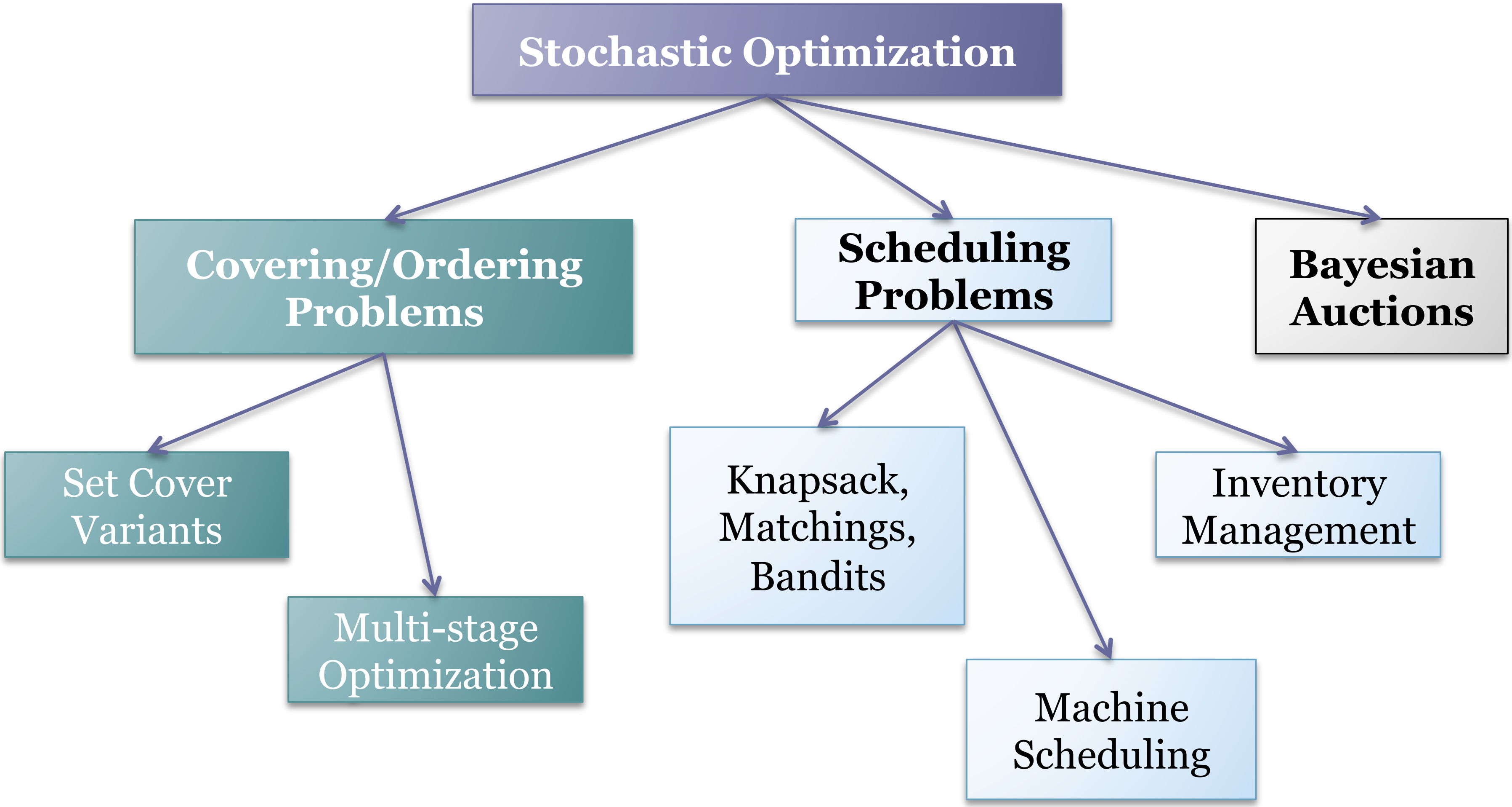
- For many problems:
  - $|S|$ is exponentially large in problem parameters

  ... or $|A|$ is exponentially large
  - Many examples to follow

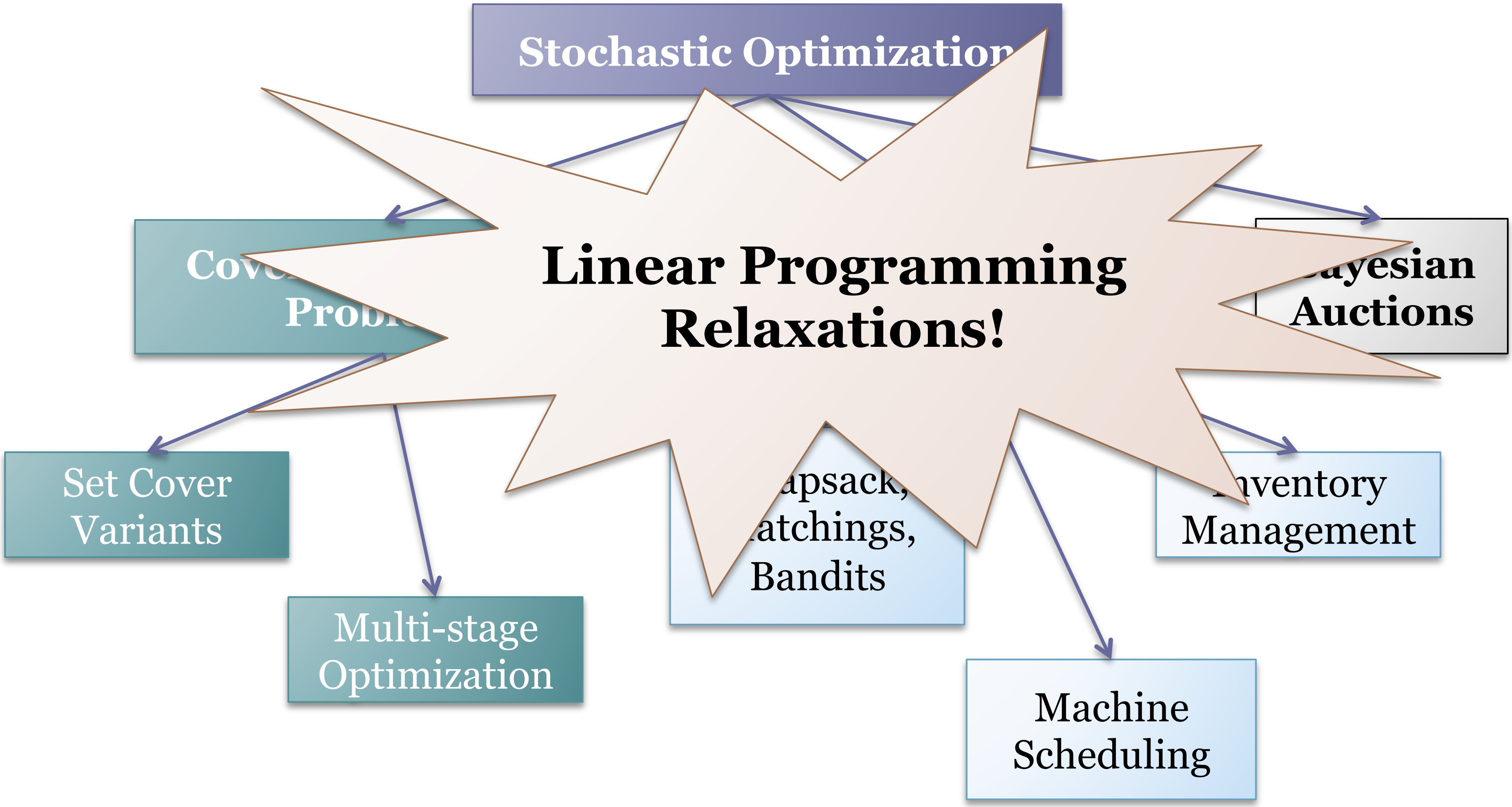
- Simpler decision policies?
  - **Approximately optimal in a provable sense**
  - Efficient to compute and execute

Talk Overview

Classes of Decision Problems



Classes of Decision Problems



Part 1. Maximum Value Problem

- Really simple decision problem
  - Illustrate basic concepts
  - Adaptive vs. Non-adaptive policies
- Non-adaptive policies
  - Submodularity and the Greedy algorithm
- Adaptive policies
  - LP Relaxation and "Weak Coupling"
  - Rounding using Markov's Inequality
- Duality
  - Simple structure of LP optimum
  - Gap between adaptive and non-adaptive policies

Part 2. Weakly Coupled LPs

- General technique via LP and Duality
  - LP relaxation has very few constraints
  - Dual yields infeasible policies with simple structure

- Examples
  - Stochastic knapsack
  - Stochastic matching
  - Bayesian multi-item pricing

Part 3. Sampling Scenarios

- Exponential sized LP over all possible "scenarios" of underlying distributions

- Solve LP or its Lagrangian by sampling the scenarios

- Examples:
  - 2-stage vertex cover
  - Stochastic Steiner trees
  - Bayesian auctions
  - Solving LPs online

Part 4. Stochastic Scheduling

- New aspect of timing the actions

- Two techniques:
  - Stronger LP relaxations than weak coupling
    - Stochastic scheduling on identical machines
    - Stochastic knapsack (not covered)
  - Greedy policies
    - Gittins index theorem

Important Disclaimer

By no means is this comprehensive!

Part 1.
The Maximum Value Problem

[Guha, Munagala '07, '09,
Dean, Goemans, Vondrak '04]

The Maximum Value Problem

- There is a gambler who is shown n boxes
  - Box j has reward drawn from distribution X_j
  - Gambler knows X_j but box is closed
  - All distributions are independent

The Maximum Value Problem

X_1 X_2 X_3 X_4 X_5

- Gambler knows all the distributions
- Distributions are independent

The Maximum Value Problem

Open some box, say Box 2

X_1	20	X_3	X_4	X_5

The Maximum Value Problem

Open another box based on observing $X_2 = 20$

X_1 20 X_3 X_4 X_5

Can open at most *k* boxes:

- Payoff = **Maximum reward** observed in these *k* boxes

Adaptivity:

- Gambler can choose next box to open based on observations so far

Example: Bernoulli Boxes

Gambler can open
k = 2 boxes

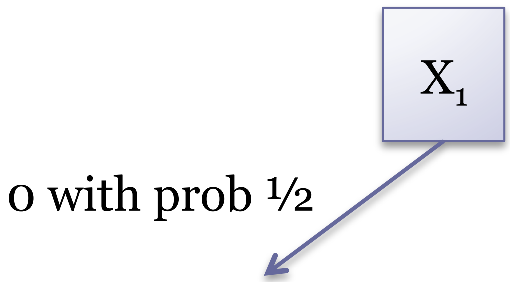
X_1	50 with probability ½
X_2	60 with probability 1/3
X_3	25 with probability 1

Optimal Decision Policy

$X_1 = B(50, 1/2)$

$X_2 = B(60, 1/3)$

$X_3 = B(25, 1)$



X_3 has expected payoff 25

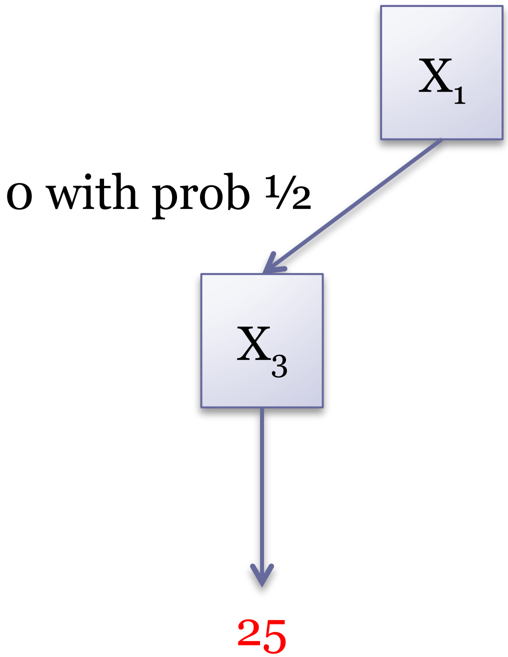
X_2 has expected payoff $60/3 = 20$

Optimal Decision Policy

$X_1 = B(50, 1/2)$

$X_2 = B(60, 1/3)$

$X_3 = B(25, 1)$

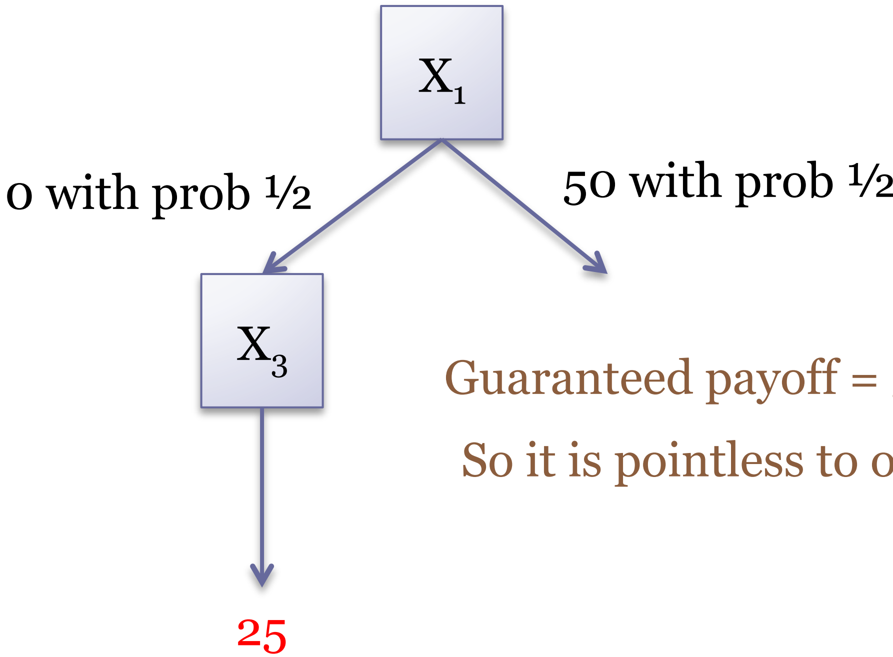


Optimal Decision Policy

$X_1 = B(50, 1/2)$

$X_2 = B(60, 1/3)$

$X_3 = B(25, 1)$



Guaranteed payoff = 50

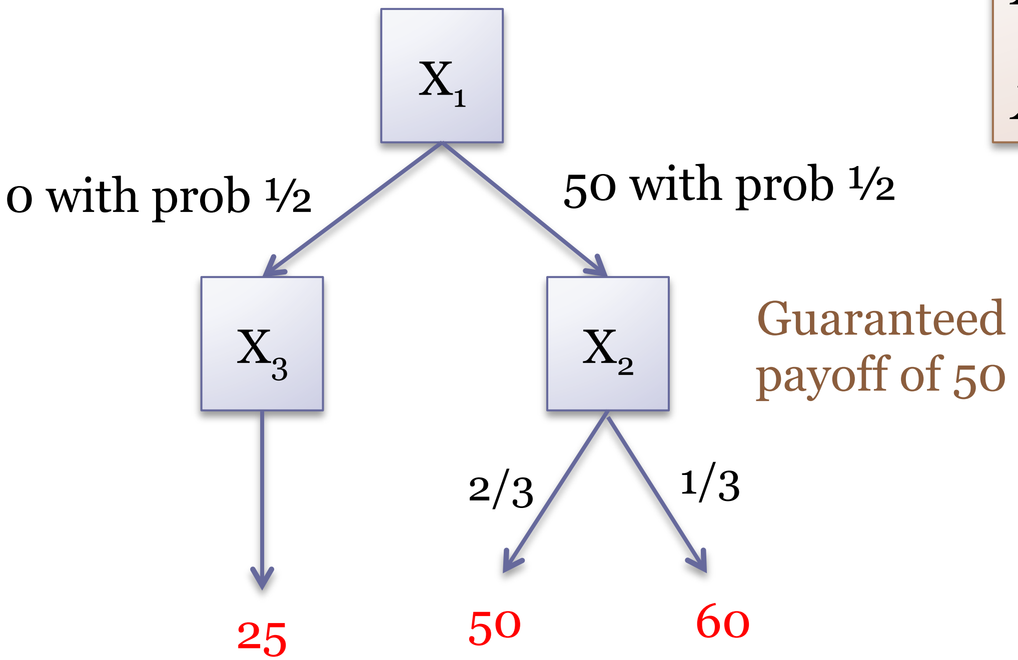
So it is pointless to open X_3

Optimal Decision Policy

$X_1 = B(50,1/2)$

$X_2 = B(60,1/3)$

$X_3 = B(25, 1)$

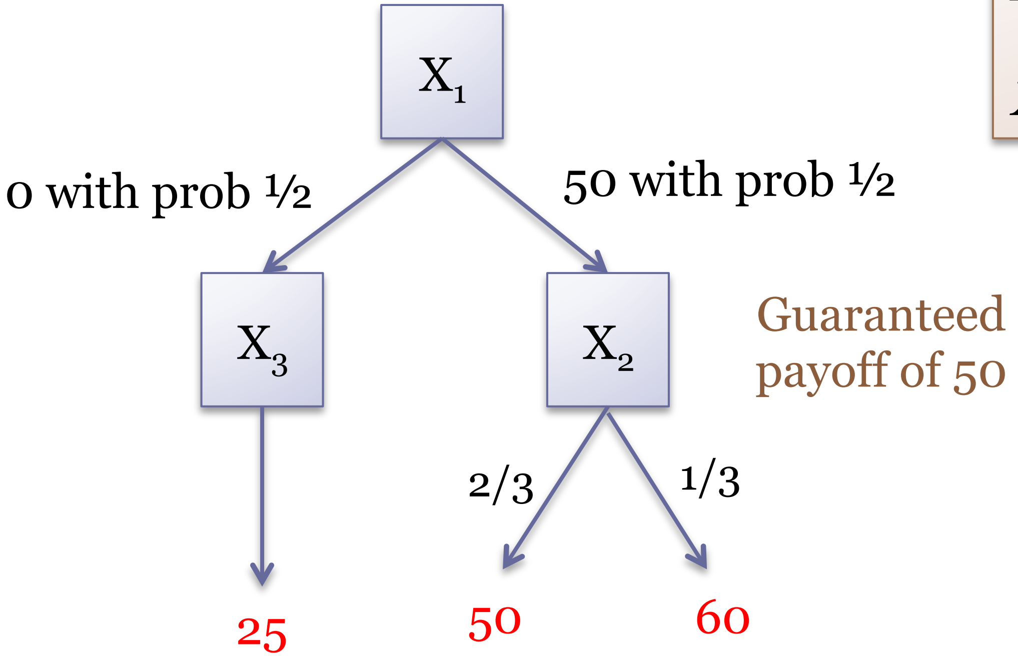


Optimal Decision Policy

$X_1 = B(50,1/2)$

$X_2 = B(60,1/3)$

$X_3 = B(25, 1)$



Expected Payoff = 25/2 + 50/3 + 60/6 = **39.167**

Can Gambler be Non-adaptive?

- Choose *k* boxes upfront before opening them
  - Open these boxes and obtain maximum value
- Best solution = Pick X_1 and X_3 upfront
  - Payoff = $\frac{1}{2} \times 50 + \frac{1}{2} \times 25 = 37.5 < 39.167$
  - Adaptively choosing next box after opening X_1 is better!

Can Gambler be Non-adaptive?

- Choose *k* boxes upfront before opening them
  - Open these boxes and obtain maximum value
- Best solution = Pick X_1 and X_3 upfront
  - Payoff = $\frac{1}{2} \times 50 + \frac{1}{2} \times 25 = 37.5 < 39.167$
  - Adaptively choosing next box after opening X_1 is better!
  - **Subtler point**: It's not that much better…

Benchmark

- Value of optimal decision policy (decision tree)
  - Call this value OPT
  - Optimal decision tree can have size exponential in k

- Can we design a:
  - *Polynomial time algorithm*
  - *… that produces poly-sized decision tree*
  - *… that approximates OPT?*

Outline for Part 1

- Approximation algorithms for Maximum Value
    - Non-adaptive policy
    - Linear programming relaxation
    - Duality and “adaptivity gap”

  - Please ignore the constant factors!

- Later on: “Weakly coupled” decision systems
    - Applications to matching, pricing, scheduling, …

Non-adaptive Algorithm

Submodularity

[Kempe, Kleinberg, Tardos ’03, …]

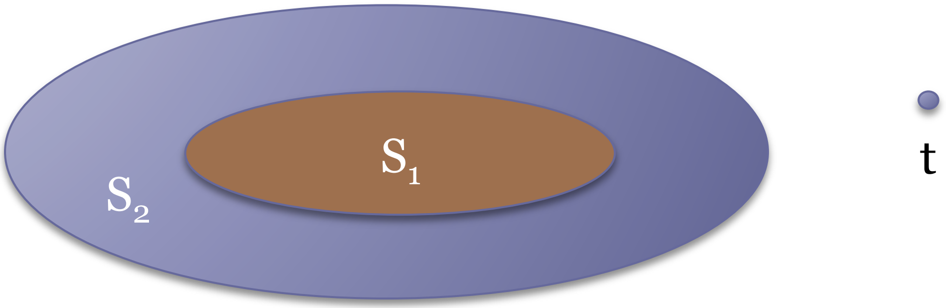
Non-adaptive Problem

- For any subset S of boxes, if gambler opens S non-adaptively, the payoff observed is

$$f(S) = \mathbf{E}\left[\max_{i \in S} X_i\right]$$

- Goal:
  - Find S such that $|S| \leq k$
  - Maximize $f(S)$

Submodularity of Set Functions



$$f\left(S_1 \cup \{t\}\right) - f\left(S_1\right) \geq f\left(S_2 \cup \{t\}\right) - f\left(S_2\right)$$

Also need **non-negativity** and **monotonicity**: $f(S_2) \geq f(S_1) \geq 0$

The Greedy Algorithm

$S \leftarrow \Phi$

While $|S| \leq k$:

$$t \leftarrow \mathrm{argmax}_{q \notin S} \left(f(S \cup \{q\}) - f(S) \right)$$

$$S \leftarrow S \cup \{t\}$$

Output S

Classical Result

[Nemhauser, Wolsey, Fisher '78]

- Greedy is a $1 - 1/e \approx 0.632$ approximation to the value of the optimal subset of size k
- Similar results hold even when:
  - Different elements have different costs and there is a budget on total cost of chosen set S
  - General matroid constraints on chosen set S

Maximum Value is Submodular

- Let D = Joint distribution of $X_1, X_2, \ldots, X_n$
- Consider any sample r drawn from D
  - Yields a sample of values $v_{1r}, v_{2r}, \ldots, v_{nr}$
  - Let $f(S, r) = \max_{i \in S} v_{ir}$
  - Easy to check this is submodular
- $f(S)$ is the expectation over samples r of $f(S,r)$
  - Submodularity preserved under taking expectation!
- **Note:** Do not need independence of variables!

More things that are Submodular

- Payoff from many opened boxes **[Guha, Munagala '07]**

$$f(S) = \mathbf{E}\left[\max_{\vec{x}\in[0,1]^n;\sum_{i\in S} s_i x_i \le B} \sum_{i\in T} X_i\right]$$

More things that are Submodular

- Payoff from many opened boxes

[Guha, Munagala '07]

$$f(S) = \mathbf{E}\left[\max_{\vec{x}\in[0,1]^n;\sum_{i\in S} s_i x_i \le B} \sum_{i\in T} X_i\right]$$

- Payoff = Minimizing the minimum value

[Goel, Guha, Munagala '06]

$$f(S) = -\log \mathbf{E}\left[\min_{i\in S} X_i\right]$$

More things that are Submodular

- Payoff from many opened boxes **[Guha, Munagala '07]**

$$f(S) = \mathbf{E}\left[\max_{\vec{x}\in[0,1]^n;\sum_{i\in S} s_i x_i \le B} \sum_{i\in T} X_i\right]$$

- Payoff = Minimizing the minimum value **[Goel, Guha, Munagala '06]**

$$f(S) = -\log \mathbf{E}\left[\min_{i\in S} X_i\right]$$

- *Spread of epidemic* with seed set S **[Kempe, Kleinberg, Tardos '03]**
- *Discrete entropy* of joint distribution of S **[Krause, Guestrin '05]**

Adaptive Algorithms

Linear Programming

[Dean, Goemans, Vondrak ’04; Guha, Munagala ‘07]

Linear Programming

Consider optimal decision policy

- Adaptively opens at most *k* boxes
- Obtains payoff from one opened box

$$y_j = \Pr[\text{Box } j \text{ is opened}]$$

$$z_{jv} = \Pr[\text{Policy's payoff is from box } j \wedge X_j = v]$$

Example from before…

$X_1 = B(50, 1/2)$

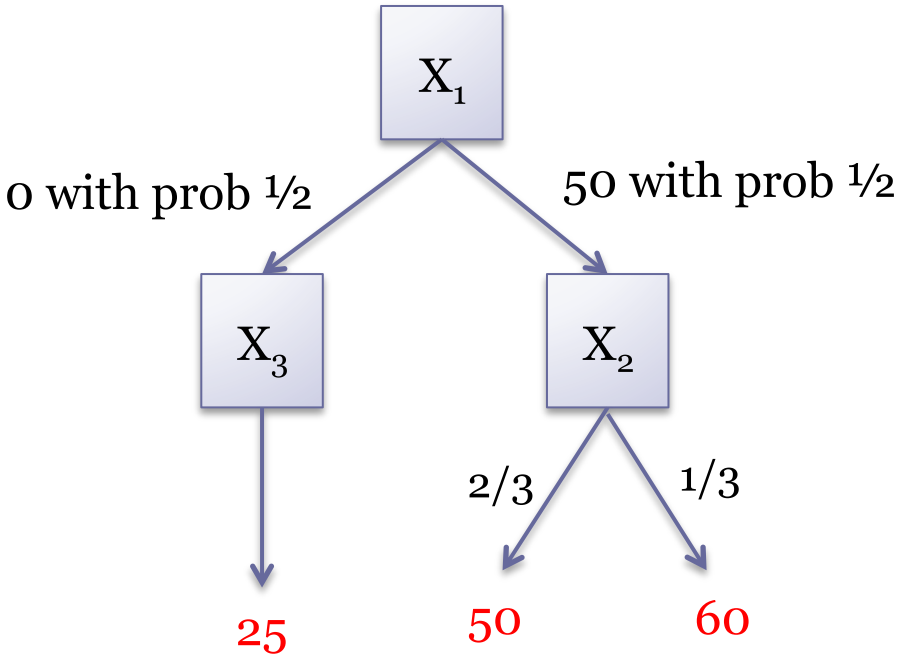
$X_2 = B(60, 1/3)$

$X_3 = B(25, 1)$

$y_1 = 1$

$y_2 = 1/2$

$y_3 = 1/2$



$z_{1,50} = 1/3$

$z_{2,60} = 1/6$

$z_{3,25} = 1/2$

Basic Idea

- LP captures behavior of policy
  - Use y_j and z_{jv} as the variables
- These variables are insufficient to capture entire structure of optimal policy
  - What we end up with will be a *relaxation*
- Steps:
  - Understand structure of relaxation
  - Convert solution to a feasible policy for gambler
  - Bound the adaptivity gap

Constraints

Let Z = Identity of box from which payoff is finally obtained

$$z_{jv} \quad = \quad \Pr[Z = j \wedge X_j = v]$$

Constraints

Let Z = Identity of box from which payoff is finally obtained

$$z_{jv} \quad = \quad \Pr[Z = j \wedge X_j = v]$$

For this event to happen, the following events must have happened:

- Box j was opened by the policy
- Box j has value $X_j = v$

Constraints

Let Z = Identity of box from which payoff is finally obtained

$$z_{jv} \quad = \quad \Pr[Z = j \wedge X_j = v]$$

For this event to happen, the following events must have happened:

- Box j was opened by the policy
- Box j has value $X_j = v$

These two events are independent since all the X's are independent!

Constraints

$$
\begin{aligned}
z_{jv} &= \Pr[Z = j \wedge X_j = v] \\
&\leq \Pr[\text{Box } j \text{ opened}] \times \Pr[X_j = v] \\
&= y_j \times f_j(v)
\end{aligned}
$$

Use independence here

Constraints

Can only get payoff from opened box: $z_{jv} \leq y_j \times f_j(v)$

Any policy obtains payoff from one box: $\sum_{j,v} z_{jv} \leq 1$

Expected number of boxes from which payoff is obtained

Relaxation: Only encode *expected number* of boxes from which payoff is obtained

Constraints

Can only get payoff from opened box: $z_{jv} \quad \leq \quad y_j \times f_j(v)$

Any policy obtains payoff from one box: $\sum_{j,v} z_{jv} \quad \leq \quad 1$

Any policy opens at most *k* boxes: $\underbrace{\sum_j y_j} \quad \leq \quad k$

Expected number of boxes opened

Relaxation: Only encode *expected number* of boxes opened and not for every decision path

Constraints

Can only get payoff from opened box: $z_{jv} \quad \leq \quad y_j \times f_j(v)$

Any policy obtains payoff from one box: $\sum_{j,v} z_{jv} \quad \leq \quad 1$

Any policy opens at most *k* boxes: $\sum_j y_j \quad \leq \quad k$

y_j is a probability value: $y_j \quad \in \quad [0, 1]$

LP Relaxation of Optimal Policy

Can only get payoff from opened box: $z_{jv} \quad \leq \quad y_j \times f_j(v)$

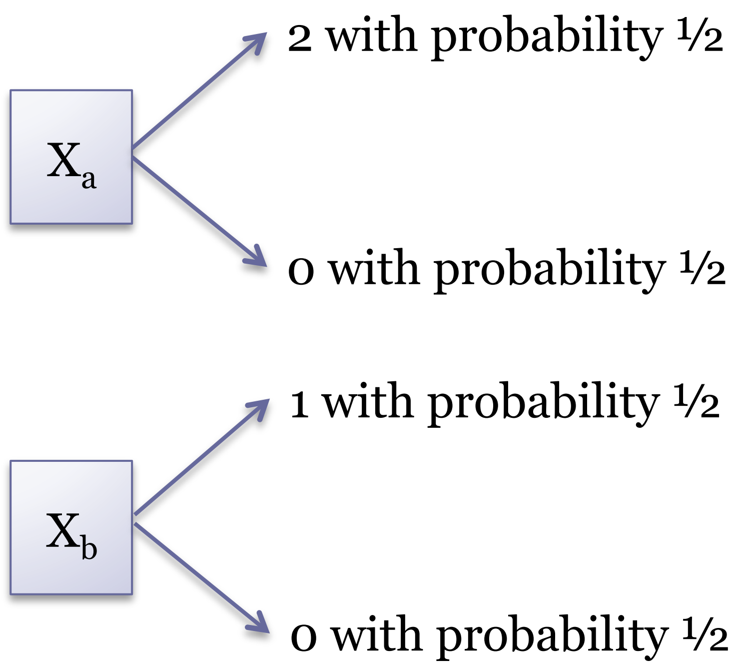
Any policy obtains payoff from one box: $\sum_{j,v} z_{jv} \quad \leq \quad 1$

Any policy opens at most *k* boxes: $\sum_j y_j \quad \leq \quad k$

y_j is a probability value: $y_j \quad \in \quad [0, 1]$

Maximize Payoff = $\sum_{j,v} v \times z_{jv}$

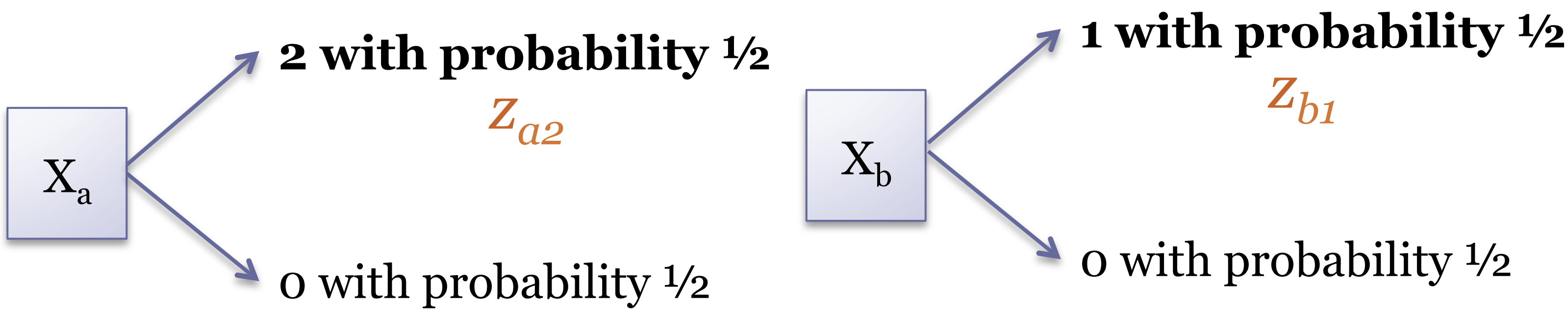
Simple Example: Open all boxes



$k = 2$

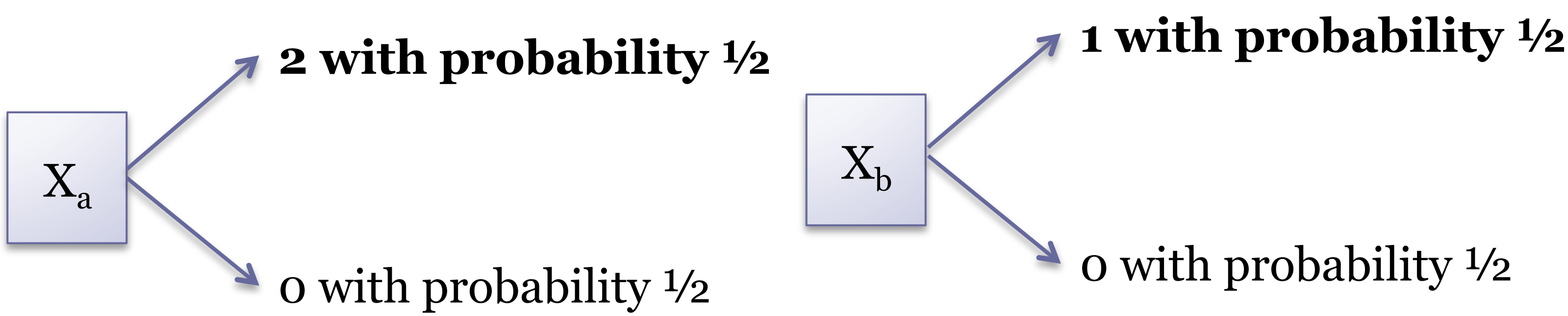
$y_a = y_b = 1$

LP Relaxation



$$\begin{array}{lrcl}
\text{Maximize} & & & 2 \times z_{a2} + 1 \times z_{b1} \\
 & z_{a2} + z_{b1} & \leq & 1 \\
 & z_{a2} & \in & [0, 1/2] \\
 & z_{b1} & \in & [0, 1/2]
\end{array}$$

LP Optimum



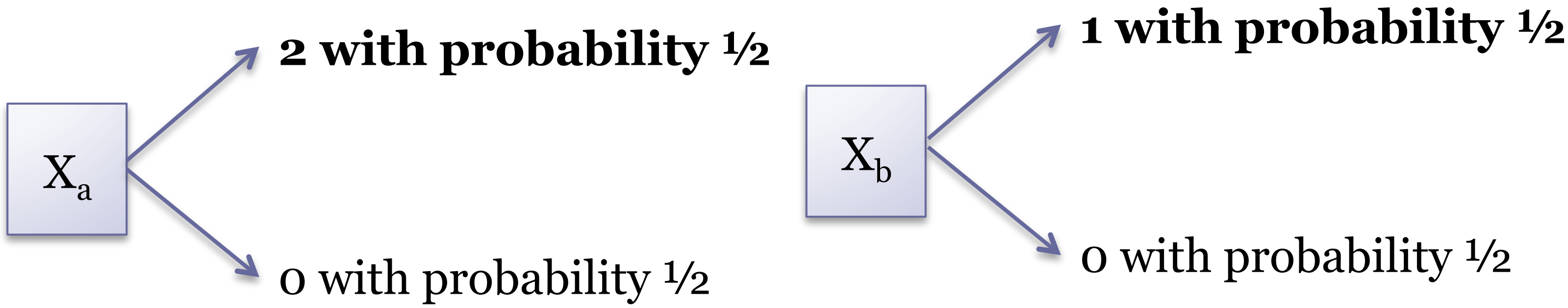
$$
\begin{aligned}
&\text{Maximize} \quad 2 \times z_{a2} + 1 \times z_{b1} \\
&z_{a2} + z_{b1} \leq 1 \\
&z_{a2} \in [0, 1/2] \\
&z_{b1} \in [0, 1/2]
\end{aligned}
$$

$z_{a2} = 1/2$

$z_{b1} = 1/2$

LP optimal payoff = 1.5

Optimal Decision Policy?



$$\text{Maximize} \quad 2 \times z_{a2} + 1 \times z_{b1}$$

$$\begin{aligned} z_{a2} + z_{b1} &\leq 1 \\ z_{a2} &\in [0, 1/2] \\ z_{b1} &\in [0, 1/2] \end{aligned}$$

$z_{a2} = 1/2$

$z_{b1} = 1/4$

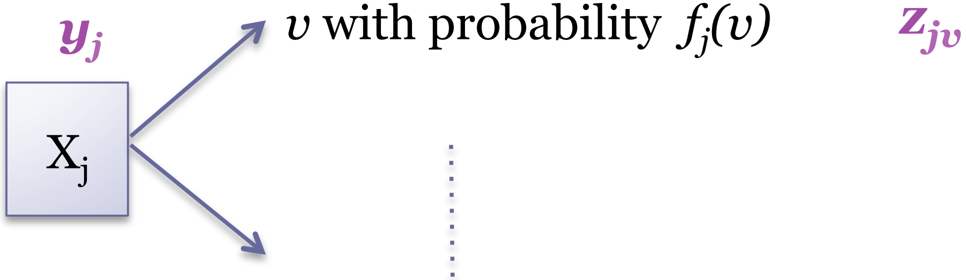
Optimal payoff

= 1.25

What do we do with LP solution?

- Will convert it into a feasible policy
- Bound the payoff in terms of LP optimum
  - LP Optimum upper bounds optimal payoff

LP Variables yield Single-box Policy P_j



Open j with probability y_j

If $X_j = v$ then

Take this payoff with probability $z_{jv}/(y_j\ f_j(v))$

Simpler Notation for Policy P_j

$$
\begin{aligned}
O(P_j) &= \Pr[j \text{ opened}] &= y_j \\
C(P_j) &= \Pr[\text{Payoff of } j \text{ chosen}] &= \textstyle\sum_v z_{jv} \\
R(P_j) &= \mathbf{E}[\text{Reward from } j] &= \textstyle\sum_v v \times z_{jv}
\end{aligned}
$$

LP Relaxation

$$
\begin{array}{llll}
\text{Maximize} & \sum_{j,v} v \cdot z_{jv} & & \\
\sum_v z_{jv} & \leq & 1 & \\
\sum_j y_j & \leq & k & \\
z_{jv} & \leq & y_j \cdot f_j(v) & \forall j, v \\
y_j & \in & [0,1] & \forall j
\end{array}
$$

$$
\begin{array}{lll}
\text{Maximize} & \sum_j R(P_j) & \\
\sum_j C(P_j) & \leq & 1 \\
\sum_j O(P_j) & \leq & k \\
\text{Each} & P_j & \text{feasible}
\end{array}
$$

LP yields collection of Single Box Policies!

What does LP give us?

- LP yields single box policies such that
  - $\Sigma_i R(P_i) \geq OPT$
  - $\Sigma_i C(P_i) \leq 1$
  - $\Sigma_i O(P_i) \leq k$
- To convert to a *feasible* policy:
  - Step 1: Order boxes arbitrarily as 1,2,3,…
  - Consider boxes in this order

Final Algorithm

- When box j encountered:
  - With probability ¾ skip this box
  - With probability ¼, execute policy P_j

Final Algorithm

- When box j encountered:
  - With probability ¾ skip this box
  - With probability ¼, execute policy P_j
- Policy P_j probabilistically decides to open j, and if opened, take its payoff

Final Algorithm

- When box j encountered:
  - With probability ¾ skip this box
  - With probability ¼, execute policy P_j

- Policy P_j probabilistically decides to open j, and if opened, take its payoff

- **If** policy decides to take payoff from j:
  - Take this payoff and **STOP**
- **Else** move to box $j+1$

Final Algorithm

- When box j encountered:
  - With probability ¾ skip this box
  - With probability ¼, execute policy P_j
- Policy P_j probabilistically decides to open j, and if opened, take its payoff
- **If** policy decides to take payoff from j:
  - Take this payoff and **STOP**
- **Else** move to box $j+1$
- If k boxes already opened, then **STOP**

Box-by-box Accounting

- Let $O_j = 1$ if policy P_j opens j
- Let $C_j = 1$ if policy P_j chooses payoff from j
- Policy reaches box i iff:

$$\left.\begin{array}{lll} \sum_{j<i} C_j & < & 1 \\ \sum_{j<i} O_j & < & k \end{array}\right\}$$

Let's lower bound this probability

Markov's Inequality

$$\Pr\left[\sum_{j<i} C_j < 1\right] \quad \geq \quad 1 - \sum_{j<i} \mathbf{E}[C_j]$$

$$\Pr\left[\sum_{j<i} O_j < k\right] \quad \geq \quad 1 - \frac{\sum_{j<i} \mathbf{E}[O_j]}{k}$$

Union Bounds

Policy reaches box i

$$\overbrace{\Pr\left[\sum_{j<i} C_j < 1 \text{ and } \sum_{j<i} O_j < k\right]}$$

$$\geq \quad 1 - \left(\sum_{j<i} \mathbf{E}[C_j] + \frac{\sum_{j<i} \mathbf{E}[O_j]}{k}\right)$$

Use Independence of Boxes

$$
\begin{aligned}
\mathbf{E}[C_j] &\leq \mathbf{E}\left[C_j \mid \text{Box } j \text{ not skipped}\right] \times \Pr\left[\text{ Box } j \text{ not skipped}\right] \\
&\leq C(P_j) \times \tfrac{1}{4}
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{E}[O_j] &\leq \mathbf{E}\left[O_j \mid \text{Box } j \text{ not skipped}\right] \times \Pr\left[\text{ Box } j \text{ not skipped}\right] \\
&\leq O(P_j) \times \tfrac{1}{4}
\end{aligned}
$$

Putting it together

Policy reaches box i

$$\Pr\left[\sum_{j<i} C_j < 1 \text{ and } \sum_{j<i} O_j < k\right]$$

$$\geq \quad 1 - \left(\sum_{j<i} \mathbf{E}[C_j] + \frac{\sum_{j<i} \mathbf{E}[O_j]}{k}\right)$$

$$\geq \quad 1 - \frac{1}{4}\left(\sum_{j<i} C(P_j) + \frac{\sum_{j<i} O(P_j)}{k}\right)$$

$$\geq \quad 1 - \frac{1}{4} \times (1+1) = \frac{1}{2}$$

8-approximation

Expected contribution to reward from P_i

$$\geq \quad \Pr[\text{ Box } i \text{ is reached}] \times \mathbf{E}[\text{Reward from } i]$$

$$\geq \quad \frac{1}{2} \times \Pr[\text{ Box } i \text{ is not skipped }] \times R(P_i)$$

$$\geq \quad \frac{R(P_i)}{8}$$

Adaptivity Gap

Duality

[Guha, Munagala '09]

Recall LP Relaxation

Maximize Payoff	$\longrightarrow$	Maximize $\sum_j R(P_j)$
Policy obtains payoff from one box	$\longrightarrow$	$\sum_j C(P_j) \leq 1$
Any policy opens at most k boxes	$\longrightarrow$	$\sum_j O(P_j) \leq k$
Single-box policy is feasible	$\longrightarrow$	Each P_j feasible

Relaxed LP

$$\text{Maximize} \quad \sum_j R(P_j)$$

$$\sum_j \left(C(P_j) + \frac{O(P_j)}{k} \right) \leq 2$$

$$\text{Each} \quad P_j \quad \text{feasible}$$

Scale down variables by factor 2

Maximize $\sum_j R(P_j)$

$$\sum_j \left(C(P_j) + \frac{O(P_j)}{k} \right) \leq 1$$

Each P_j feasible

Lagrangian

Maximize $\sum_j R(P_j)$

$\sum_j \left(C(P_j) + \frac{O(P_j)}{k} \right) \leq 1$ ← Dual variable = w

Each P_j feasible

Max. $w + \sum_j \left(R(P_j) - w \times C(P_j) - \frac{w}{k} O(P_j) \right)$

Each P_j feasible

Interpretation of Lagrangian

$$\text{Max. } w + \sum_j \left(R(P_j) - w \times C(P_j) - \frac{w}{k} O(P_j)\right)$$

$$\text{Each } P_j \text{ feasible}$$

- Decouples into a separate optimization per box!
- Can open and choose payoff from many boxes

Optimization Problem for Box *j*

$$\text{Max. } R(P_j) - w \times C(P_j) - \frac{w}{k} O(P_j)$$

$$P_j \text{ feasible}$$

- Net value from choosing *j*:
  - If *j* opened, then pay cost = w/k
  - If we choose payoff of *j*, then pay cost = w
  - If we choose payoff of *j*, obtain that reward
- Net value = Reward minus cost paid

Optimal Solution to Lagrangian

- For box j, choose solution with better value
- **Solution 1:** Don't open box
    - Net value = 0
- **Solution 2:** Open box
    - Pay cost = w/k
    - If Reward > w, then choose this reward, pay cost w
    - Net value = $\mathbf{E}$[Reward – Cost]
- Decision to open any box is deterministic!

Strong Duality (roughly speaking)

$$\text{Lag}(w) = \sum_j R_j + w \times \left(1 - \sum_j \left(C_j + \frac{O_j}{k}\right)\right)$$

Choose Lagrange multiplier w such that

$$\begin{aligned} \textstyle\sum_j \left(C_j + \frac{O_j}{k}\right) &= 1 \\ \Rightarrow \qquad \textstyle\sum_j R_j &\geq \frac{OPT}{2} \end{aligned}$$

Non-adaptive Policy

- Since O_j is either 0 or 1
  - LP optimum opens at most k boxes deterministically!
  - Suppose we open all these boxes
- The expected maximum payoff of these boxes is at least the value of rounding the LP
  - But rounding has value at least OPT/16
- Therefore, the adaptivity gap is at most 16!
  - Better choice of w improves this to factor 3

[Guha, Munagala, Shi '09]

Takeaways…

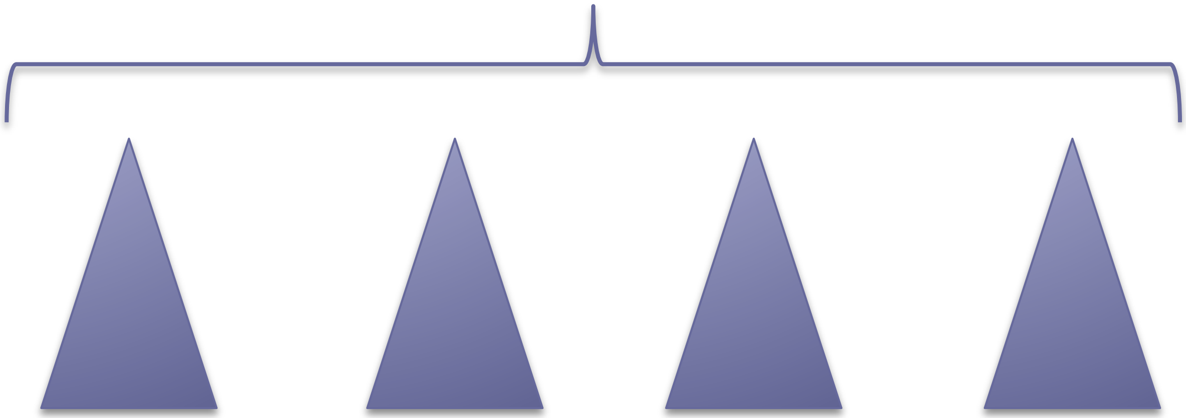
- LP-based proof oblivious to non-linear closed form for max
- Automatically yields policies with right “form”
  - Adaptivity gap follows from duality
- Needs independence of random variables
  - Weakly coupled linear program and rounding
  - More on weak and strong relaxations in next half!

Part 2.
Weakly Coupled Relaxations

Weakly Coupled Decision Systems

Independent decision spaces

Few constraints coupling decisions across spaces



[Singh & Cohn ’97; Meuleau *et al*. ‘98]

General Recipe

- Write LP with constraints on expected values
    - **Important:** Constant number of such constraints
    - Stronger relaxations are sometimes needed
- Solve LP and use Markov's inequality to round
- Dual typically yields more structured solution
    - For instance, threshold policies and adaptivity gaps

Maximum Value Setting

- Each box defines its own decision space
  - Payoffs of boxes are independent
- Coupling constraints (write in expectation):
  - At most k boxes opened
  - At most one box's payoff finally chosen
- LP yields a threshold policy:
  - Choose payoff if value > dual multiplier w

Stochastic Knapsack

[Dean, Goemans, Vondrak '04; Bhalgat, Goel, Khanna '11]

- Size of item i drawn from distribution X_i
  - Learn actual size only after placing i in knapsack
  - Sizes of items independent
  - Any size at most knapsack capacity B
- Adaptive policy for placing items in knapsack
  - If knapsack capacity violated, then STOP
- Maximize expected reward

Weakly Coupled Relaxation

$$\text{Maximize} \qquad \underbrace{\textstyle\sum_j R_j y_j}_{\text{Expected reward}}$$

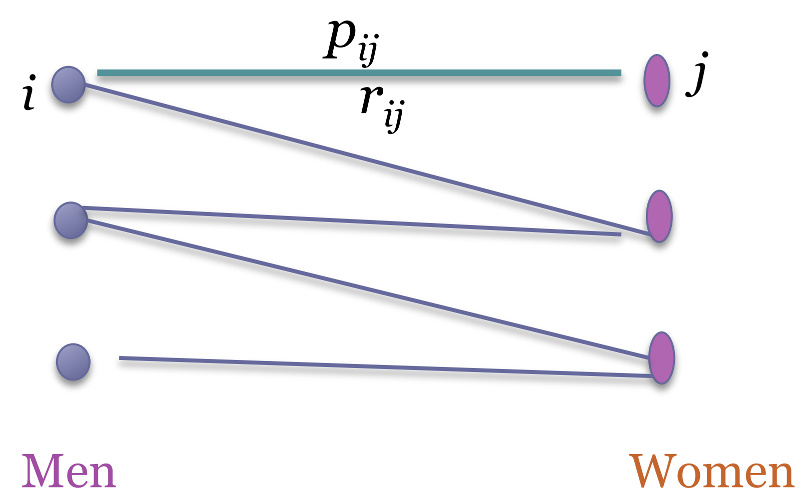
$$\textstyle\sum_j y_j \cdot \mathbf{E}[X_j] \quad \leq \quad 2B$$

$$y_j \quad \in \quad [0, 1]$$

↑

Pr[*j* placed in knapsack]

Stochastic Matching



- Can send some man *i* and some woman *j* on date
- Date *succeeds* with probability p_{ij} and yields reward r_{ij}
  - Successful match removes *i* and *j* from graph
  - Failed match deletes edge *(i,j)*

Stochastic Matching [Chen et al. '09; Bansal et al. '10]

- **Input:** Matrix of p_{ij} and r_{ij}
- **Decision policy:**
  - Adaptive order of setting up dates
- **Goal:**
  - Maximize expected reward of successful matches

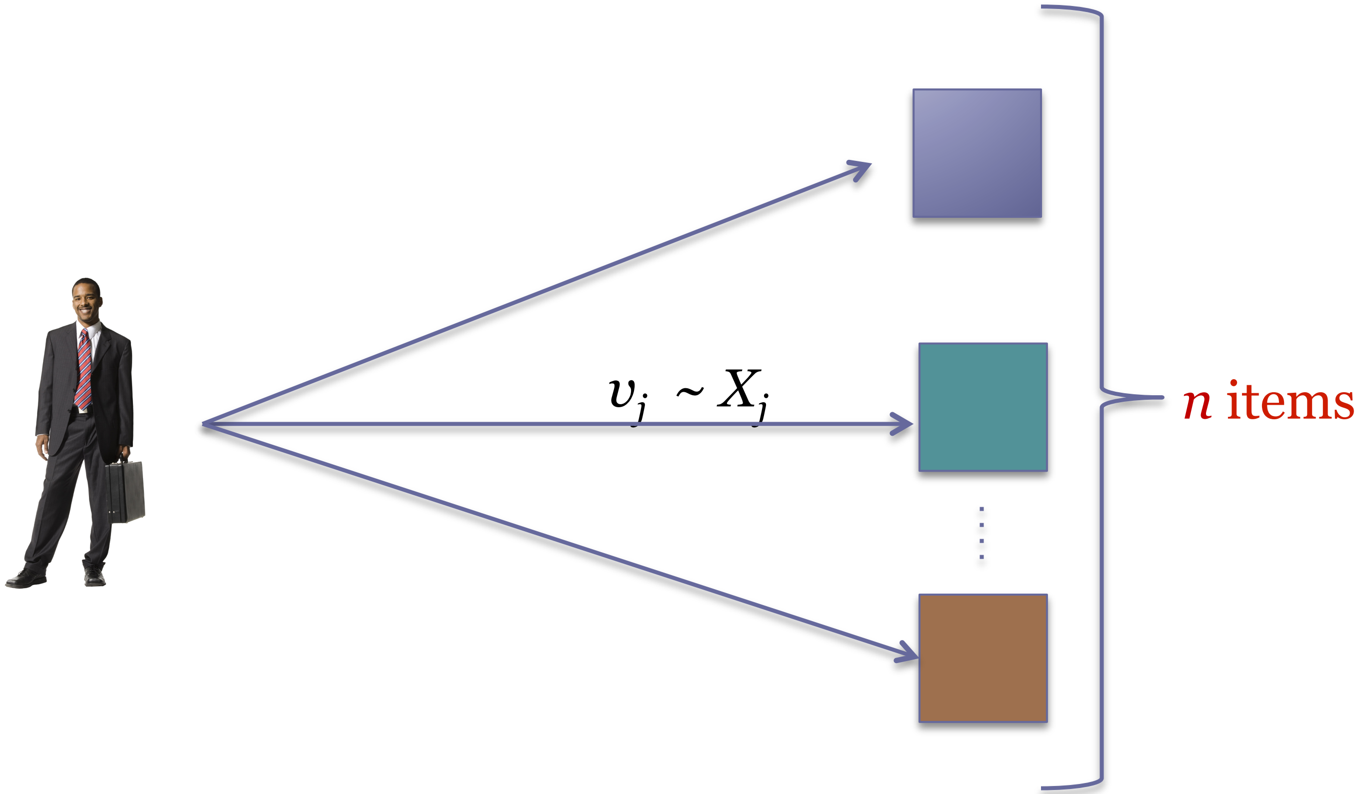
LP Relaxation

$$
\begin{array}{rcll}
\text{Maximize} & & \sum_{i,j} r_{ij} p_{ij} x_{ij} & \\
\sum_j p_{ij} x_{ij} & \leq & 1 & \forall i \\
\sum_i p_{ij} x_{ij} & \leq & 1 & \forall j \\
x_{ij} & \in & [0,1] & \forall i, j
\end{array}
$$

Expected number of successful matches per man and woman at most 1

Pr[*i* goes on a date with *j*]

Bayesian Pricing

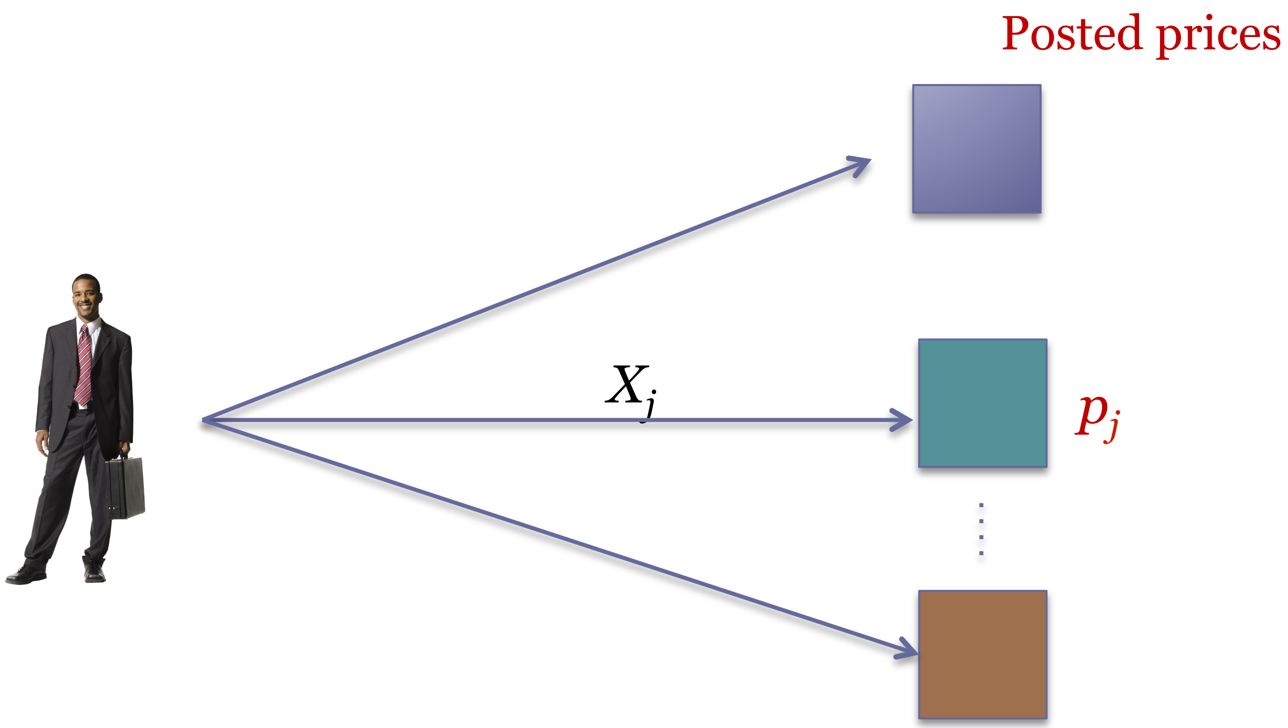


Unit Demand Setting

[Chawla, Hartline, Kleinberg ’07; Chawla et al. ‘10; Bhattacharya et al. ‘10]

- One agent and n items
  - Agent wants only one item
- Value v_j follows independent distribution X_j
  - Exact value known only to agent
  - Seller only knows distribution

Item Pricing Scheme



Buyer chooses item that maximizes $v_j - p_j$

Revenue Maximization

- Bayesian Pricing:
  - Post prices p_j for each item j based on knowing X_j
  - Agent chooses that item that maximizes $v_j - p_j$
  - Seller earns the price p_j

- Seller's goal:
  - Maximize Revenue = Expected price earned

LP Variables

$$x_{jp} = \Pr[\text{Price of } j = p]$$

$$y_{jp}(v) = \Pr[\text{Price of } j = p \wedge X_j = v \wedge j \text{ is bought}]$$

LP Constraints:

- Every item has exactly one price
- Agent buys at most one item
- *Agent only buys item if value is larger than price*

LP Relaxation

$$\text{Maximize} \qquad \sum_{j,p,v} p \cdot y_{jp}(v)$$

$$\sum_{j,p,v} y_{jp}(v) \leq 1$$

E[Items bought] is at most 1

$$\sum_p x_{jp} \leq 1 \qquad \forall j$$

One price for each *j*

$$y_{jp}(v) \leq x_{jp} f_j(v) \qquad \forall j, p, v \geq p$$

Pr*[X_j = v]*

Lagrangian decouples across items!

$$\text{Maximize} \quad \sum_{j,p,v} (p - \lambda) \cdot y_{jp}(v)$$

$$\sum_p x_{jp} \leq 1 \qquad \forall j$$

$$y_{jp}(v) \leq x_{jp} f_j(v) \qquad \forall j, p, v$$

↑ Integral variable

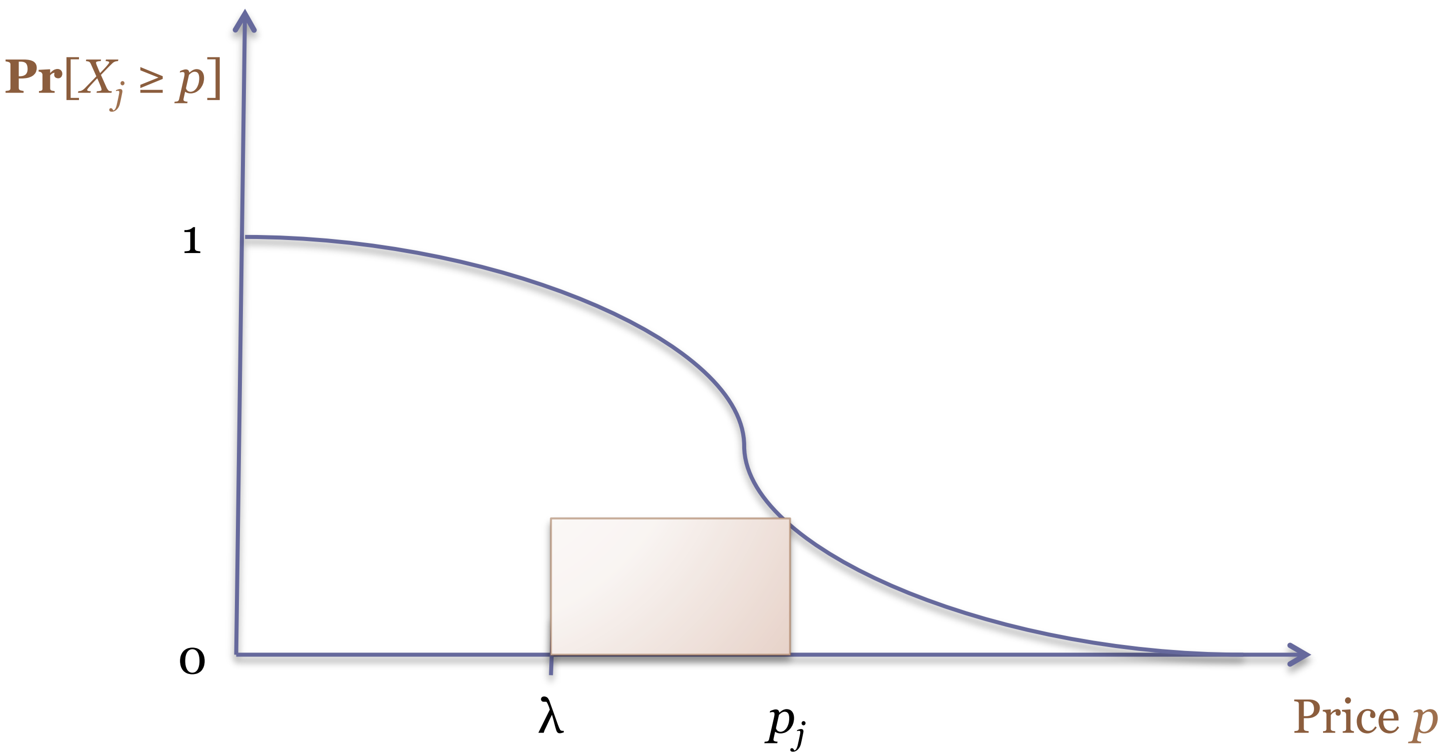
For each j, Lagrangian chooses one price p_j

Lagrangian optimum is simple

$$p_j^*(\lambda) = \mathrm{argmax}_{p \geq \lambda} \left((p - \lambda) \cdot \Pr\left[X_j \geq p\right]\right)$$

LP optimum chooses λ so that expected number of items bought is exactly 1

Lagrangian Optimum for Item j



Some Complexity Results

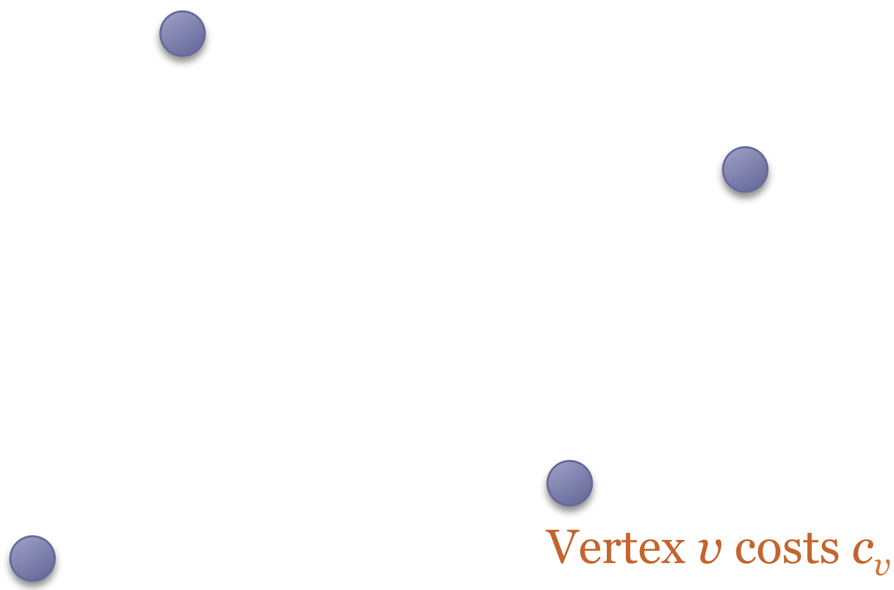
- Bayesian Pricing
  - (Q)PTAS for "reasonable" distributions [Cai Daskalakis '11]
  - NP-complete in general [Chen et al. '13]
  - Correlated distributions
    - Hard to approximate beyond logarithmic factors [Briest '11]
- Stochastic Knapsack
  - PTAS [Bhalgat, Goel, Khanna '11]

Part 3.
Sampling-based Approaches

Overview

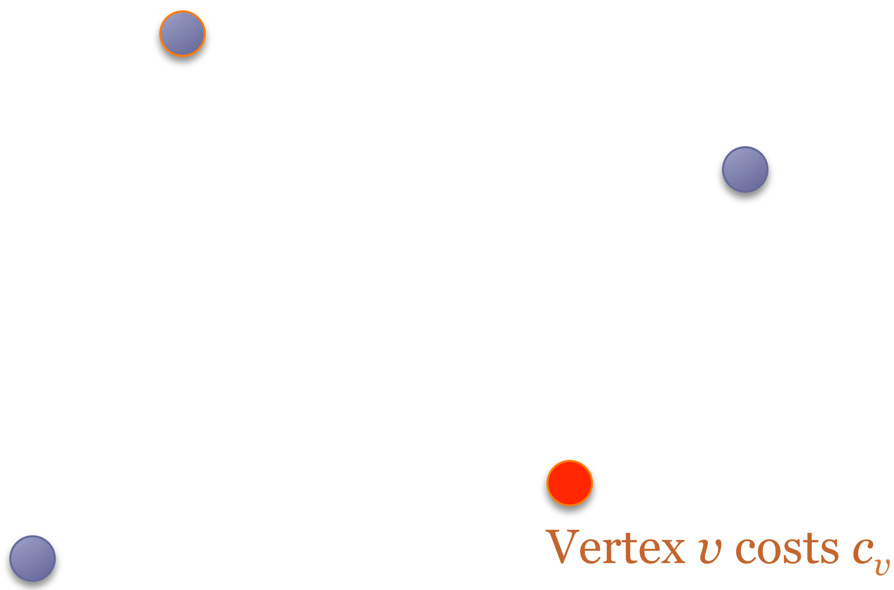
- MDPs with small number of “stages”
- Exponential sized LP over all possible “scenarios” of underlying distribution
  - Solve LP or its Lagrangian by sampling the scenarios
- Examples:
  - 2-stage vertex cover
  - Stochastic Steiner trees (combinatorial algorithm)
  - Bayesian auctions
  - Solving LPs online

Multi-stage Vertex Cover



Distribution D over possible edge sets that can be realized

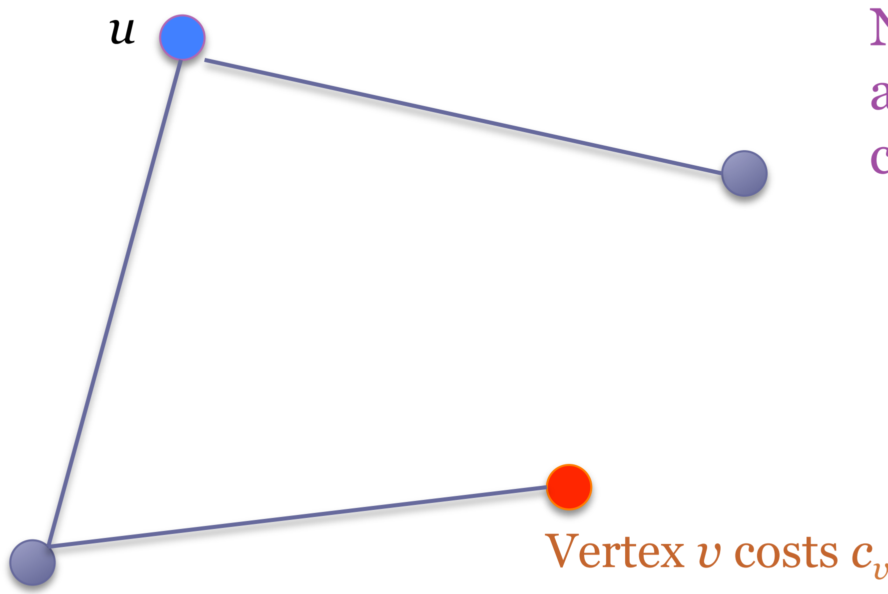
Stage 1: Buy some vertices cheaply



Pay cost c_v

Buy some vertices only knowing D

Stage 2: Edge set realized



Need to buy vertices at scaled up price to cover realized edges

Total cost = $c_v + \lambda\, c_u$

Multi-stage Covering Problems

[Kleywegt, Shapiro, Homem-de-Mello '01; Shmoys, Swamy '04; Charikar, Chekuri, Pal '05]

- Decision Policy:
  - What vertices should we buy in Stage 1?
  - Knowing only *D*, costs, and scaling factor $\lambda > 1$
- Minimize total expected cost of vertices
  - Expectation over realization of edges from *D*

LP when |D| is small

$$\text{Maximize } \sum_v x_v + \lambda \cdot \mathbf{E}_{\sigma \in D}\left[\sum_v y_v(\sigma)\right]$$

$$x_u + x_v + y_u(\sigma) + y_v(\sigma) \;\geq\; 1 \;\forall \sigma, e \in E(\sigma)$$

Rounding similar to vertex cover

Randomized rounding yields tight 2 approximation

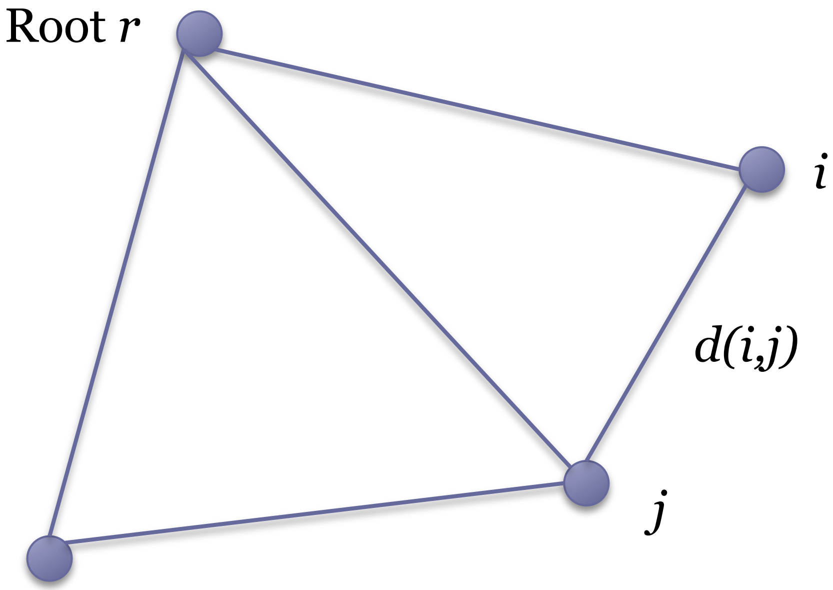
Generalizes to multi-stage vertex cover

Black Box Access to *D*

- Sample Average Approximation
  - Draw poly many samples; solve LP on these samples
  - Approximation results carry over with small loss

- Combinatorial "boosted sampling" **[Gupta et al.'04]**
  - Draw a set of samples from *D* in Stage 1
  - Solve covering problem on union of these samples
  - Augment this solution with the realization in stage 2

Stochastic Steiner Tree



Distribution D over vertices V

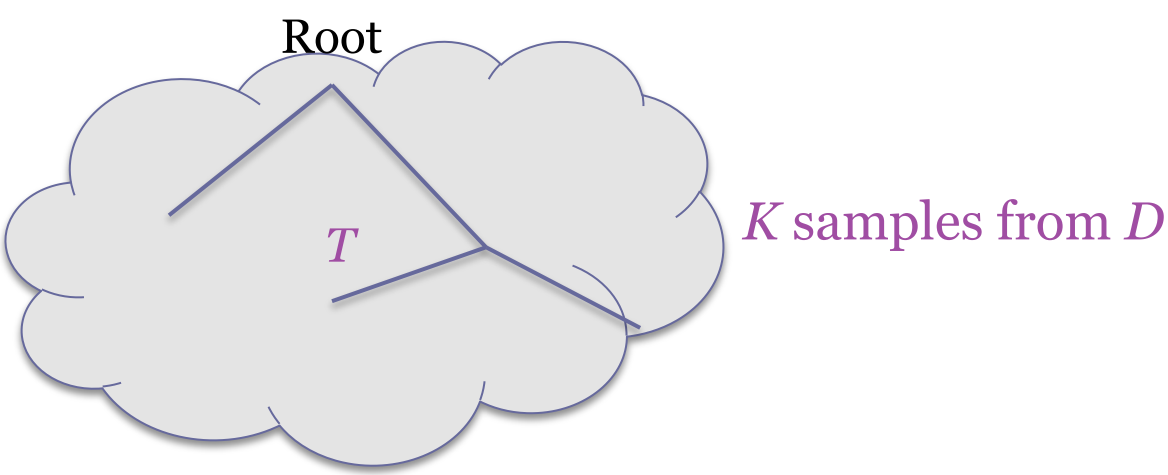
Stochastic Steiner Tree

[Garg et al. '08]

- K vertices arrive one at a time
  - Drawn *i.i.d.* from distribution D
- **Goal:**
  - Construct ***online*** Steiner tree connecting arriving vertices to r
- **Technique:** Sampling from D

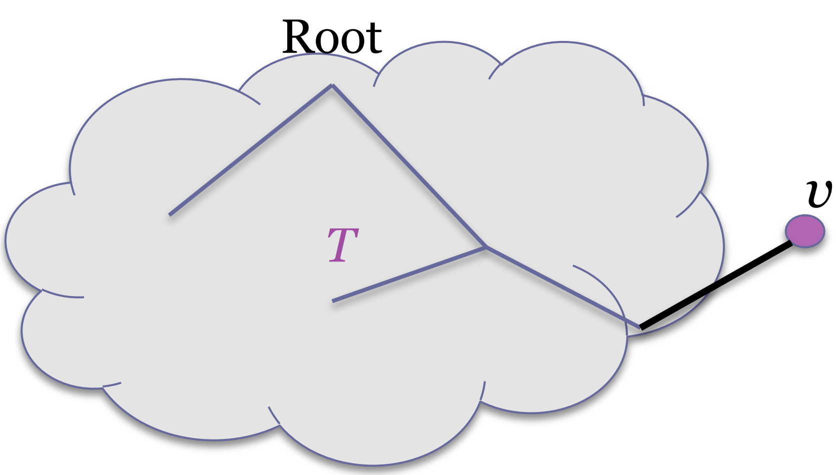
Algorithm: Offline Stage 1

- Draw K samples from D
- Construct 2-approximate Steiner tree T on samples
- Expected cost at most $2OPT$
  - Samples statistically identical to online input



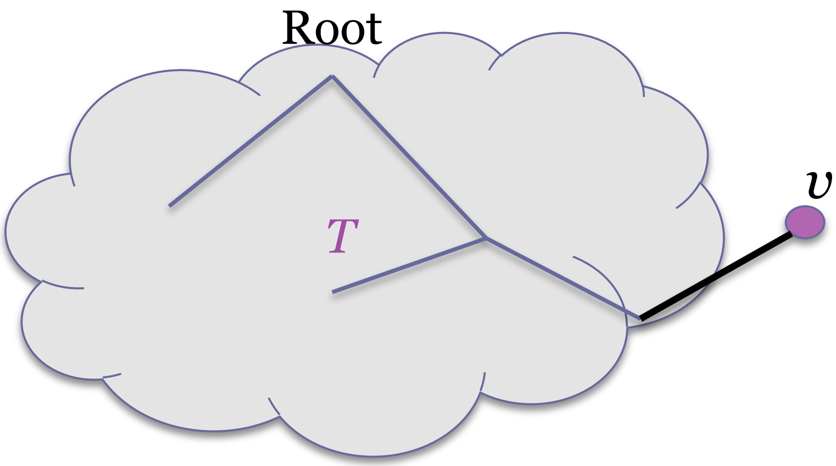
Algorithm: Online Stage 2

- When input vertex v arrives online
  - Connect v by shortest path to T

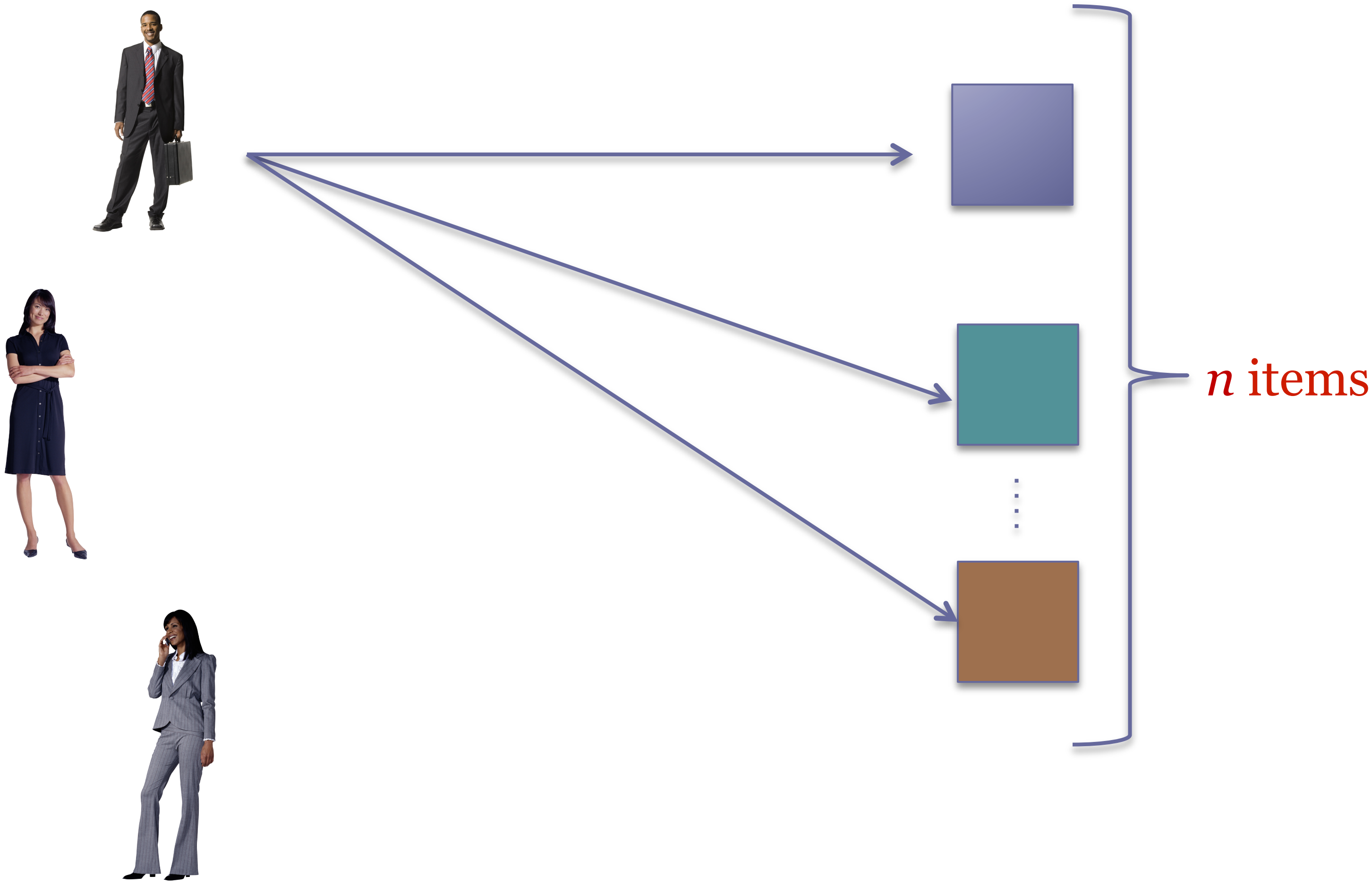


Sampling Analysis

- K points in Stage 1 and v together are a random sample of size $K+1$ from D.
  - Therefore, expected cost of connecting v most $2OPT/K$
- Overall cost at most $4\ OPT$!



Bayesian Multi-item Auctions



Bayesian Setting

[Cai, Daskalakis Weinberg, '12-'15, Bhalgat, Gollapudi, Munagala '13]

- Many bidders and items
  - Constraints on possible allocations
- Bidder *j's* valuation vector follows distribution σ_j
  - Exact value known only to bidder
  - Distributions for different bidders independent
  - Auctioneer only knows distribution
- **Assume:** Single bidder's distribution σ_j is poly-size

Auction Design

- Design auction maximizing expected revenue (or total price charged)
  - Auction = (Allocations, Prices) given revealed bids

Auction Design

- Design auction maximizing expected revenue (or total price charged)

- Bayesian Incentive Compatibility:
    - Revealing true value maximizes *expected utility* of bidder
    - Expectation is over distribution of other agents

Auction Design

- Design auction maximizing expected revenue (or total price charged)
- Bayesian Incentive Compatibility:
  - Revealing true value maximizes *expected utility* of bidder
  - Expectation is over distribution of other agents
- Individual Rationality:
  - Charge prices so that utility of any agent is non-negative
  - Constraint could be per scenario and not in expectation

Why is this easier than Pricing?

- We allow “lotteries”
  - Randomized menu of allocations and prices
  - Incentive compatibility in expectation
  - Lotteries can be encoded by an LP

- Deterministic menus are hard to approximate!

[Briest ‘11]

Two types of LP variables

Expected value (marginal) variables

$$X_j(\vec{v_j}) \quad = \quad \mathbf{E}[\text{ Allocation to } j | \sigma_j = \vec{v_j}]$$

$$P_j(\vec{v_j}) \quad = \quad \mathbf{E}[\text{ Price for } j | \sigma_j = \vec{v_j}]$$

Expectation over valuations of other agents

Per-scenario variables

$$\vec{x}(\eta) \quad = \quad \text{Allocations} \mid \text{Valuations} \;= \eta$$

$$\vec{p}(\eta) \quad = \quad \text{Prices} \mid \text{Valuations} \;= \eta$$

Exponentially many scenarios!

LP Constraints

- Expected value constraints for every agent j and valuation vector v_j:
  - Bayesian incentive compatibility
  - Maximize expected revenue

LP Constraints

- Expected value constraints for every agent j and valuation vector v_j:
  - Bayesian incentive compatibility
  - Maximize expected revenue
- Per-scenario constraints (exponentially many):
  - Allocations and prices are feasible for every scenario η
  - Individual rationality

LP Constraints

- Expected value constraints for every agent *j* and valuation vector v_j:
  - Bayesian incentive compatibility
  - Maximize expected revenue
- Per-scenario constraints (exponentially many):
  - Allocations and prices are feasible for every scenario η
  - Individual rationality
- Coupling constraints:

$$X_j(\vec{v_j}) = \sum_{\eta \mid \eta_j = \vec{v_j}} \Pr[\eta] \cdot x_j(\eta)$$

$$P_j(\vec{v_j}) = \sum_{\eta \mid \eta_j = \vec{v_j}} \Pr[\eta] \cdot p_j(\eta)$$

Exponentially large summation!

Key Idea: Sample Scenarios

- Take Lagrangian of coupling constraints
  - One Lagrange multiplier for each agent and its value
  - Poly-many multipliers or "virtual welfares"

$$
\begin{aligned}
X_j(\vec{v_j}) &= \textstyle\sum_{\eta|\eta_j=\vec{v_j}} \Pr[\eta] \cdot x_j(\eta) \\
P_j(\vec{v_j}) &= \textstyle\sum_{\eta|\eta_j=\vec{v_j}} \Pr[\eta] \cdot p_j(\eta)
\end{aligned}
$$

Key Idea: Sample Scenarios

- Take Lagrangian of coupling constraints
  - One Lagrange multiplier for each agent and its value
  - Poly-many multipliers or "virtual welfares"

- Lagrangian decouples into two separate problems:
  - LP over expected value variables
  - Separate maximization problem for each scenario η and take expectation over scenarios
    - Estimate this expectation by sampling the scenarios!

Key Idea: Sample Scenarios

- Take Lagrangian of coupling constraints
  - One Lagrange multiplier for each agent and its value
  - Poly-many multipliers or "virtual welfares"

- Lagrangian decouples into two separate problems:
  - LP over expected value variables
  - Maximization problem for each scenario η and take expectation over scenarios
    - Estimate this expectation by sampling scenarios!

- Given efficient oracle for solving Lagrangian
  - Solve LP using no-regret learning, Ellipsoid, …

“Online” Algorithms

[Agarwal, Devanur ‘14]

- Suppose scenarios arrive *i.i.d.* from unknown distribution

- Need to solve some LP over expected allocations
  - But with feasibility constraints per scenario
  - **Motivation:** Budgeted allocations, envy-freeness, …

- Arriving scenarios can be treated as samples!
  - Implies overall LP can be solved online via Lagrangian
  - Need not even know distribution upfront!

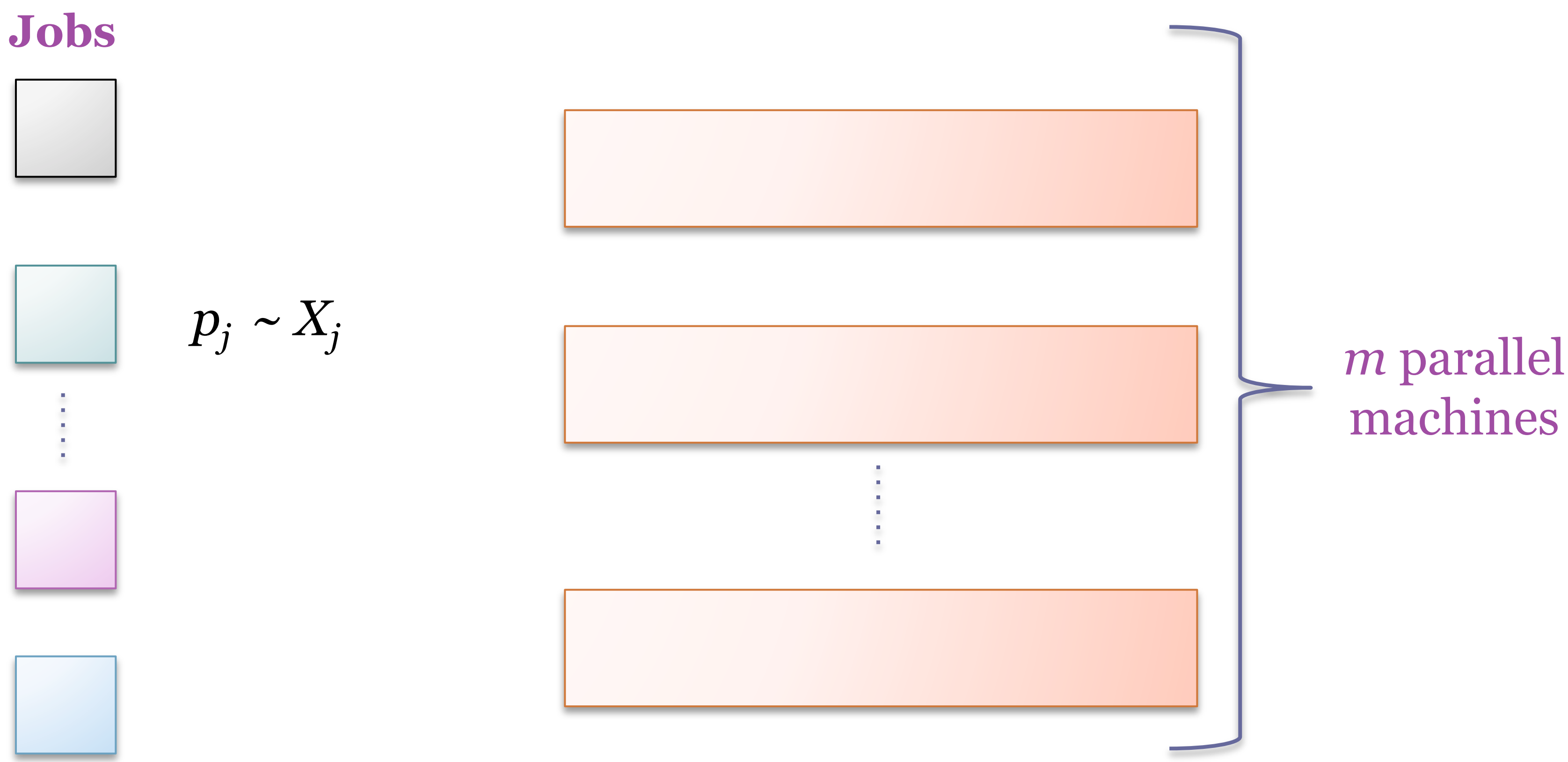
Part 4.
Scheduling Problems

Overview

- New aspect of timing the actions
    - So far, we have ignored timing completely!

- Two techniques:
    - Stronger LP relaxations than weak coupling
        - Stochastic scheduling on identical machines
        - Stochastic knapsack (not covered)
    - Greedy policies
        - Gittins index theorem

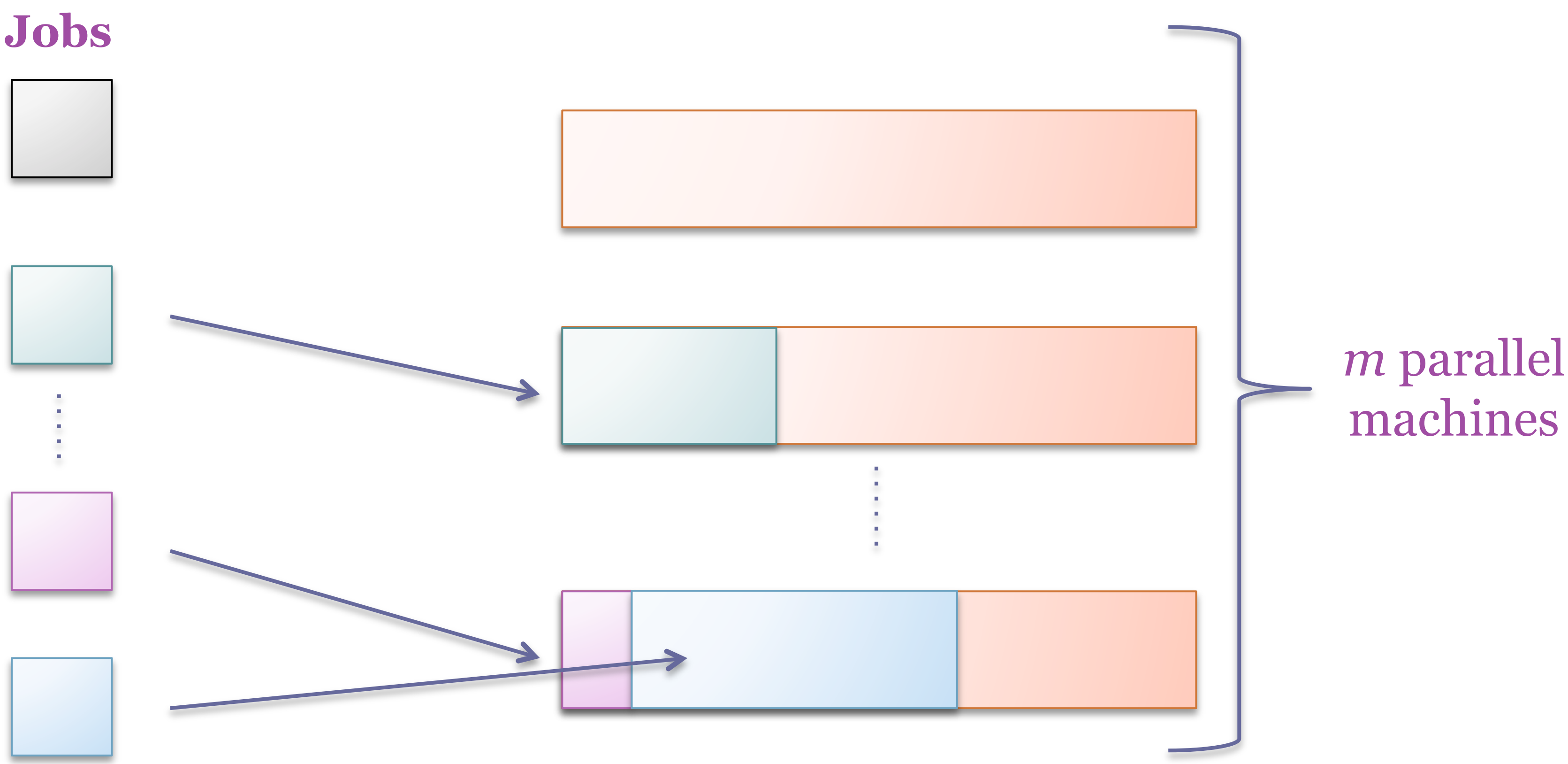
Stochastic Scheduling



Stochastic Scheduling [Mohring, Schulz, Uetz '96]

- Realize exact length only after job is scheduled
  - No preemption or release dates
- Adaptive policy:
  - Adaptive ordering of jobs and machines to assign them to
- Goal:
  - Minimize expected sum of completion times

Adaptive Policy



LP-based Reduction to Determinism

- Write LP assuming job lengths are deterministic
- Variables are start times S_j of jobs

$$\text{Minimize} \qquad \sum_j (p_j + S_j)$$

$$\sum_{j \in A} p_j S_j \quad \geq \quad \frac{1}{2m} \sum_{i \neq j \in A} p_i p_j - \frac{m-1}{2m} \sum_{j \in A} p_j^2$$

$$\forall \text{ subsets } A \text{ of jobs}$$

LP for Stochastic Case

- Take expectations over job lengths
  - Note job length independent of start time
- Rounding: Schedule jobs in increasing order of LP objective

Minimize $\sum_j (\mathbf{E}[S_j] + \mu_j)$

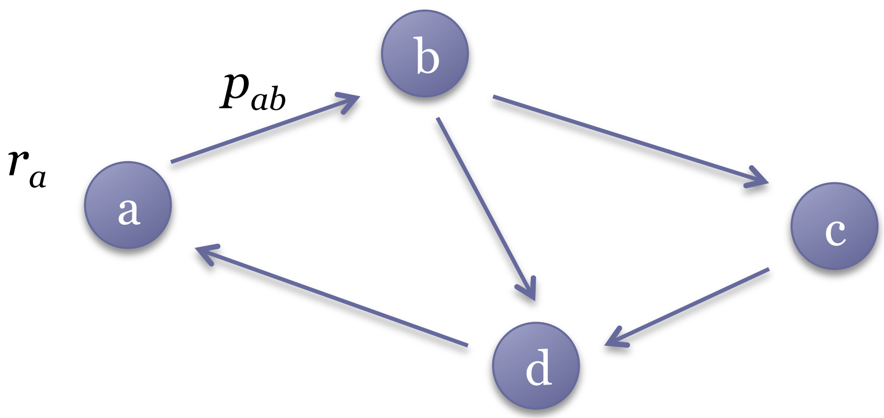
$$\sum_{j \in A} \mu_j \mathbf{E}[S_j] \geq \frac{1}{2m} \left(\sum_{j \in A} \mu_j \right)^2 - \frac{1}{2} \sum_{j \in A} \mu_j^2 - \frac{m-1}{2m} \sum_{j \in A} \sigma_j^2$$

$\forall$ subsets A of jobs

Multi-armed Bandits

[Gittins and Jones '74, Tsitsiklis '80]

- n **independent** bandit arms
  - Each arm defines its own Markov decision space
  - Only two actions per arm: "PLAY" or "STOP"

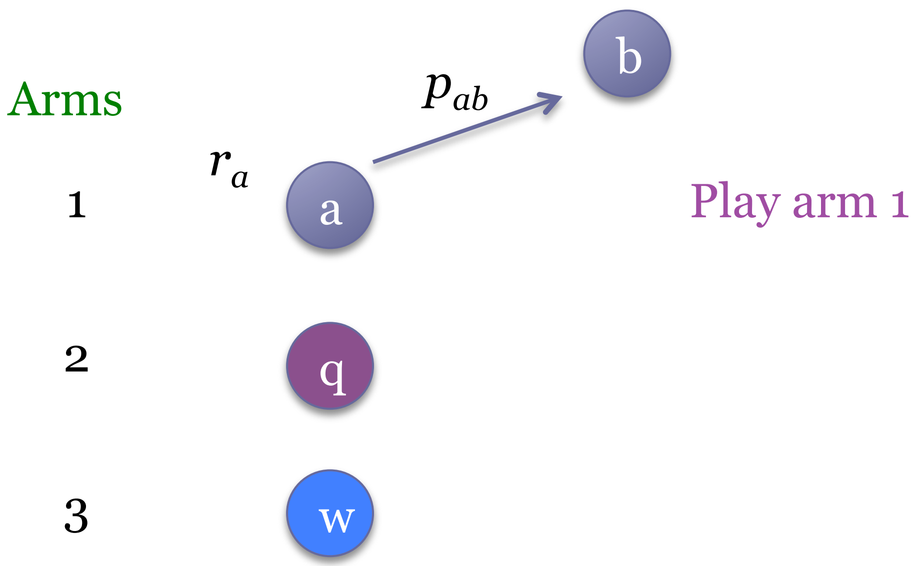


State space of an arm

Multi-armed Bandits

[Gittins and Jones '74, Tsitsiklis '80]

- n **independent** bandit arms
  - Each arm defines its own Markov decision space
  - Only two actions per arm: "PLAY" or "STOP"
- At each step, can play at most one arm



Arm's state changes only when played

Multi-armed Bandits

[Gittins and Jones ’74, Tsitsiklis ‘80]

- R_t = Reward at time t
- Υ = Discount factor < 1
- Find policy that maximizes discounted reward:

$$\mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t\right]$$

What is a policy?

- Given current state of each arm
  - Which arm to play next?
- “State space” is exponential in number of arms
- Surprising but non-trivial result:
  - A greedy policy is optimal!
  - Polynomial time computable and executable!

Why is this non-trivial?

- Playing arm whose current state has highest reward may be sub-optimal
  - Arm can have low reward right now, but playing it yields state with high reward
  - But this can happen two states down the road, …
- This means policy needs to take entire future behavior of arm into account!

Single Arm Problem via Duality

- Fix penalty (or dual cost) λ
- Focus on some state s of some arm i
  - Suppose this is the start state
- Suppose arm i was only arm in system
  - At each step, can play arm i by paying penalty λ
  - Or can STOP and exit
- $V_i(s, \lambda)$ = Optimal discounted payoff
  - Easy to compute by dynamic programming

The Gittins Index

- For state *s* of arm *i*, Gittins index:

  Largest penalty λ such that $V_i(s, \lambda) = 0$

- Same as:
  - Expected discounted per-step reward if we keep playing *i* as long as state is "at least as good as" *s*

- "At least as good as" = Larger Gittins index!

Intuition

- A state has large Gittins index if either:
  - State *itself* has high reward
    - So play in this state and then STOP
  - State *leads to* states with large reward
    - So long-term per-step reward is large

- In either case, this state is a “good” state to play

Gittins index policy

- At each step, play the arm whose current state has largest Gittins index
  - Optimal!

- Proof of optimality
  - Exchange argument similar to greedy analyses

Other Problems and Approaches

- Stochastic makespan, Bin packing

  [Kleinberg, Rabani, Tardos '97]

- Inventory management **[Levi, Pal, Roundy, Shmoys '04]**

- Stochastic set cover and probing problems

  [Etzioni et al., '96; Munagala, Srivastava, Widom '06; Liu et al., '08; Gupta-Nagarajan '15 ...]

- Techniques:
  - Analysis of greedy policies
  - Discretizing distributions and dynamic programming

Open Questions

- How far can we push LP based techniques?
  - Can we encode adaptive policies more generally?
  - For instance, bandits with matroid constraints?

- Several problem classes poorly understood
  - Stochastic machine scheduling
  - Auctions with budget constraints

- What if we don't have full independence?
  - Some success in auction design
  - In general, need tractable models of correlation

Thanks!