Nearly-Optimal Solutions to Dynamic and Stochastic Resource Capacity Allocation Problems

Peter Jacko*

INFORMS APS
July 8, 2011

*Basque Center for Applied Mathematics, Spain

Neoclassical Economics

- Standard problem of neoclassical economics:
  - ▷ maximize aggregate utility w.r.t. budget constraint
- Standard assumptions (among others):
  - ▷ goods/services are continuously-divisible
  - ▷ budget (money) is continuously-divisible
  - ▷ goods/services do not change over time
- Standard solution:
  - ▷ marginal utility per unit of money spent must be equal for each good

Motivation

- Resource allocation when the assumptions do not hold:
  - ▷ communications (routing, scheduling)
  - ▷ robotics (tracking, ranking)
  - ▷ marketing (assortment, inventory control)
  - ▷ labor economics (job search)
  - ▷ clinical trials (treatment selection)
- Can we still apply marginalism ideas?
- What policies are optimal?
- What policies are simple to implement?

Talk Outline

- Resource allocation problem
- Framework
- Approach and adaptive greedy rules
- Known results
- Challenges

Resource Allocation Problem (RAP)

- Stochastic and dynamic
- There is a number of independent competitors
- Constraint: resource capacity at every moment
- Objective: maximize expected "reward"
- Captures the exploitation vs. exploration trade-off
  - ▷ always exploiting (being myopic) is not optimal
  - ▷ always exploring (being utopic) is not optimal
- This is a model of learning by doing!

Questions to Answer

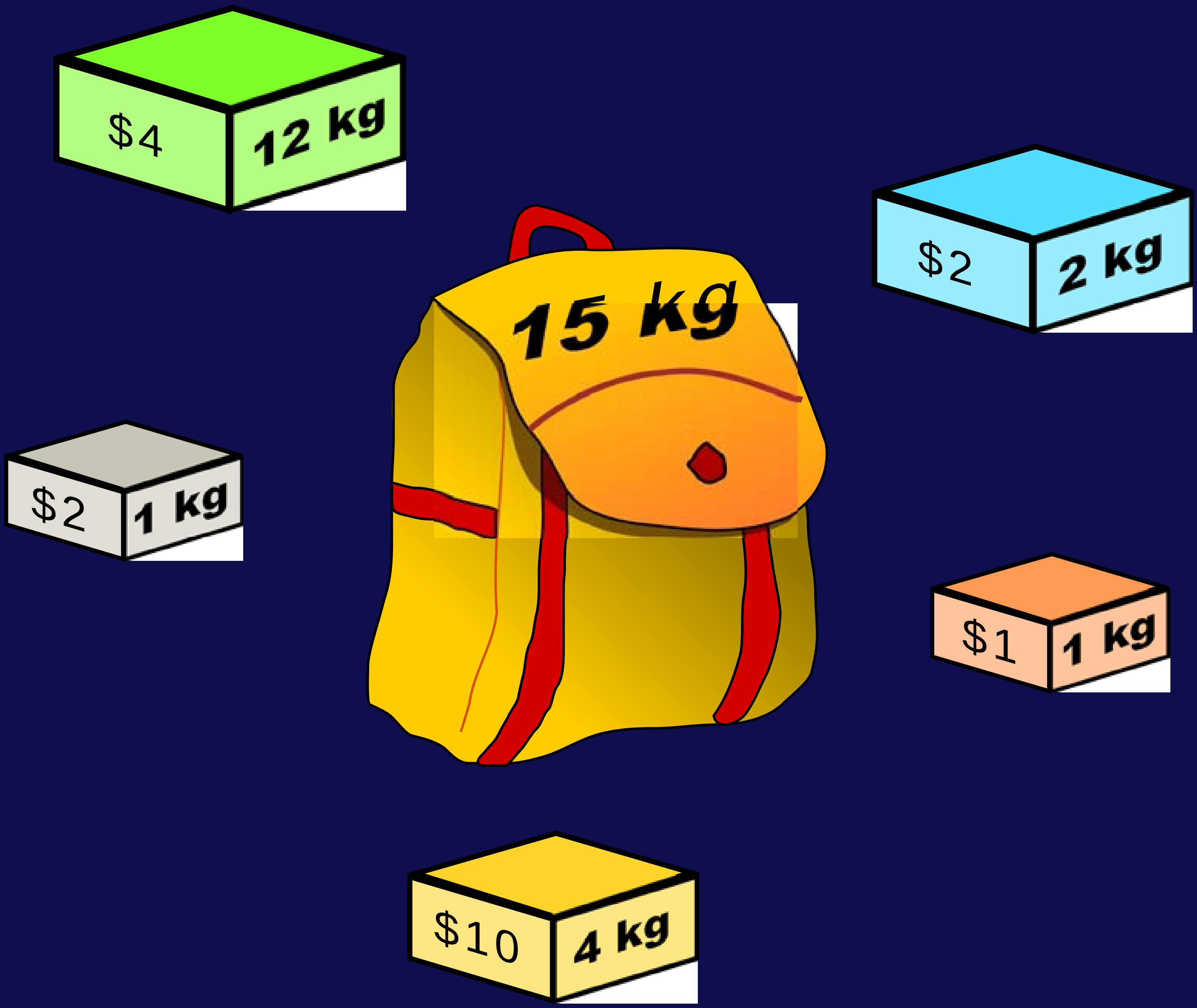
- [Economic] For a given joint goal, is it possible to define dynamic quantities for each competitor that can be interpreted as prices? And if yes,

- [Algorithmic] How to calculate such prices quickly?

- [Mathematical] Under what conditions is there a greedy rule that achieves optimal resource capacity allocation?

- [Experimental] If greedy rules are not optimal, how close to optimality do they come? And how do they compare to alternative rules?

Static RAP: Knapsack Problem



MDP Framework

- Markov Decision Processes (Stoch. dyn. programming)
- Discrete time model $(t = 0, 1, 2, \dots)$
- Competitor $k \in \mathcal{K}$ is defined by
  - ▷ state space $\mathcal{N}_k$, action space $\mathcal{A}$
  - ▷ expected one-period capacity consumption $\boldsymbol{W}_k^a$
  - ▷ expected one-period reward $\boldsymbol{R}_k^a$
  - ▷ one-period transition probability matrix $\boldsymbol{P}_k^a$
- State process $X_k(t) \in \mathcal{N}_k$
- Action process $a_k(t) \in \mathcal{A}$ – to be decided

Example: Job Sequencing Problem

- Find a serving sequence minimizing the total cost of waiting of jobs $k \in \mathcal{K}$
  - ▷ c_k = cost of waiting for jobs k
  - ▷ μ_k = service rate for jobs k
- $\mathcal{N}_k := \{\text{'completed'}, \text{'waiting'}\}$, $\mathcal{A}_k := \{\text{'serve'}, \text{'wait'}\}$
- expected one-period capacity consumption

$$W_{k,\text{'completed'}}^{\text{'serve'}} := 1, \qquad W_{k,\text{'waiting'}}^{\text{'serve'}} := 1,$$
$$W_{k,\text{'completed'}}^{\text{'wait'}} := 0, \qquad W_{k,\text{'waiting'}}^{\text{'wait'}} := 0;$$

Example: Job Sequencing Problem

- expected one-period reward

$$R^{\text{'serve'}}_{k,\text{'completed'}} := 0, \qquad R^{\text{'serve'}}_{k,\text{'waiting'}} := -c_k(1-\mu_k),$$
$$R^{\text{'wait'}}_{k,\text{'completed'}} := 0, \qquad R^{\text{'wait'}}_{k,\text{'waiting'}} := -c_k;$$

- one-period transition probability matrices

$$\boldsymbol{P}^{\text{'serve'}}_k := \begin{array}{r} \\ \text{'completed'} \\ \text{'waiting'} \end{array} \begin{array}{c} \begin{array}{cc} \text{'completed'} & \text{'waiting'} \end{array} \\ \begin{pmatrix} 1 & 0 \\ \mu_k & 1-\mu_k \end{pmatrix} \end{array},$$

$$\boldsymbol{P}^{\text{'wait'}}_k := \begin{array}{r} \\ \text{'completed'} \\ \text{'waiting'} \end{array} \begin{array}{c} \begin{array}{cc} \text{'completed'} & \text{'waiting'} \end{array} \\ \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \end{array}.$$

Resource Allocation Problem

- Formulation under the β-discounted criterion:

$$\max_{\pi\in\Pi} \sum_{k\in\mathcal{K}} \mathbb{E}^{\pi}\left[\sum_{t=0}^{\infty} \beta^t R_{k,X_k(t)}^{a_k(t)}\right]$$

$$\text{subject to} \quad \sum_{k\in\mathcal{K}} W_{k,X_k(t)}^{a_k(t)} \le W, \qquad \text{for all } t = 0, 1, 2, \ldots$$

- This problem is PSPACE-hard
  - ▷ intractable to solve exactly by Dynamic Programming
  - ▷ instead, we relax and decompose the problem

Whittle's Relaxation

- Fill the capacity in expectation
  - ▷ infinite number of constraints is replaced by one
  - ▷ sort of perfect market assumption

$$\max_{\pi\in\Pi} \sum_{k\in\mathcal{K}} \mathbb{E}^{\pi}\left[\sum_{t=0}^{\infty} \beta^t R_{k,X_k(t)}^{a_k(t)}\right]$$

$$\text{subject to} \quad \sum_{k\in\mathcal{K}} \mathbb{E}^{\pi}\left[\sum_{t=0}^{\infty} \beta^t W_{k,X_k(t)}^{a_k(t)}\right] \leq \sum_{t=0}^{\infty} \beta^t W$$

- Provides an upper bound for RAP

Lagrangian Relaxation

- Pay cost λ for using the capacity
  - the constraint is moved into the objective

$$\max_{\pi \in \Pi} \sum_{k \in \mathcal{K}} \mathbb{E}^{\pi} \left[\sum_{t=0}^{\infty} \beta^t R_{k, X_k(t)}^{a_k(t)} \right] - \lambda \sum_{k \in \mathcal{K}} \mathbb{E}^{\pi} \left[\sum_{t=0}^{\infty} \beta^t W_{k, X_k(t)}^{a_k(t)} \right]$$

- Also provides an upper bound for RAP
- Decomposes due to competitor's independence into single-competitor parametric subproblems
  - solved by identifying the efficiency frontier

Dynamic Prices

- We will assign each competitor a dynamic price
- They arise in the solution of the parametric subproblem
  - ▷ optimal policy: use capacity iff price lower than λ
- Prices are values of λ when optimal solution changes
- They define indifference curves
- However, such prices may not exist!
- Price computation:
  - ▷ in general, by parametric simplex method
  - ▷ after math, sometimes obtained in a closed form

Knapsack Rule

$$
\begin{aligned}
\max_{\boldsymbol{z}} \quad & \sum_{k\in\mathcal{K}, a\in\mathcal{A}_{k,n_k}} v^a_{k,n_k} a z_{k,a} & \\
\text{s. t.} \quad & \sum_{k\in\mathcal{K}, a\in\mathcal{A}_{k,n_k}} a z_{k,a} = W & \text{(GKP)} \\
& \sum_{a\in\mathcal{A}_{k,n_k}} z_{k,a} \le 1 \qquad \text{for all } k \in \mathcal{K}, & \\
& z_{k,a} \in \{0,1\} \quad \text{for all } k \in \mathcal{K}, a \in \mathcal{A}_{k,n_k} &
\end{aligned}
$$

where $z_{k,a}$ denotes whether competitor k is allocated capacity a

Adaptive Greedy Rules

- We are concerned with the following rule
  - ▷ at each moment be greedy:
    prefer competitors with higher current prices
  - ▷ this is the greedy solution to (GKP)
- It is adaptive because the prices are dynamic
- Experiments and simulations suggest that it gives a nearly-optimal solution to RAP
- In some simpler problems, it is optimal

Application: Perishable Items

- Decision moments: $s = T, T-1, \ldots, 1$
  - ▷ occupies space W, yields profit R
  - ▷ if promoted, it remains unsold with probability p
  - ▷ if not promoted, it remains unsold with probability $q > p$
  - ▷ once sold, it never resurrects
- Deadline: $s = 0$
  - ▷ yields salvage value αR, $\alpha < 1$ if not sold
- Aim: Fill in the promotion space so that the expected aggregate total β-discounted revenue is maximized

Price for Perishable Items

- Under a regularity condition

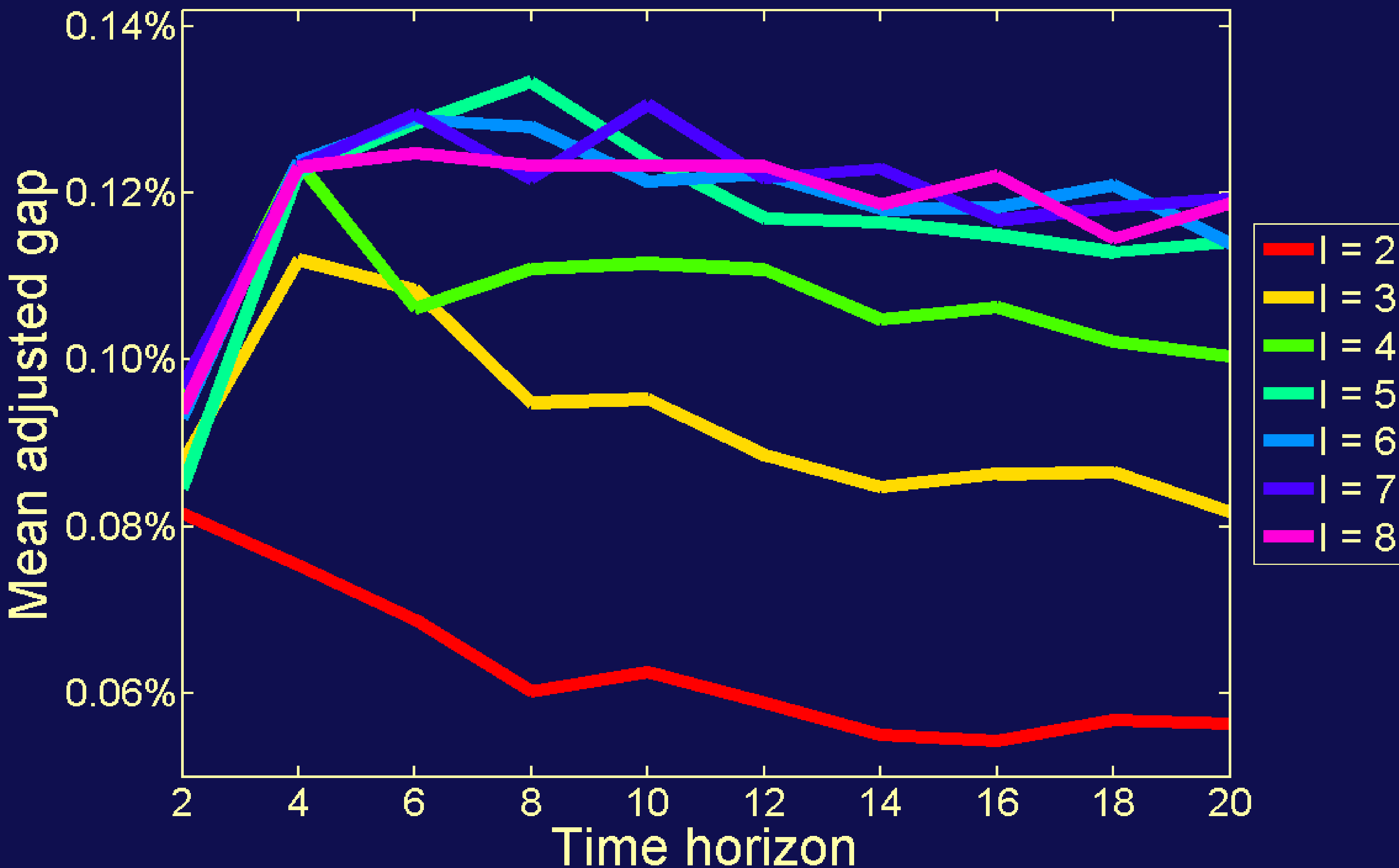
$$(1-q)-\alpha(1-\beta q)\geq 0$$

- Closed-form formula of the price of one item:

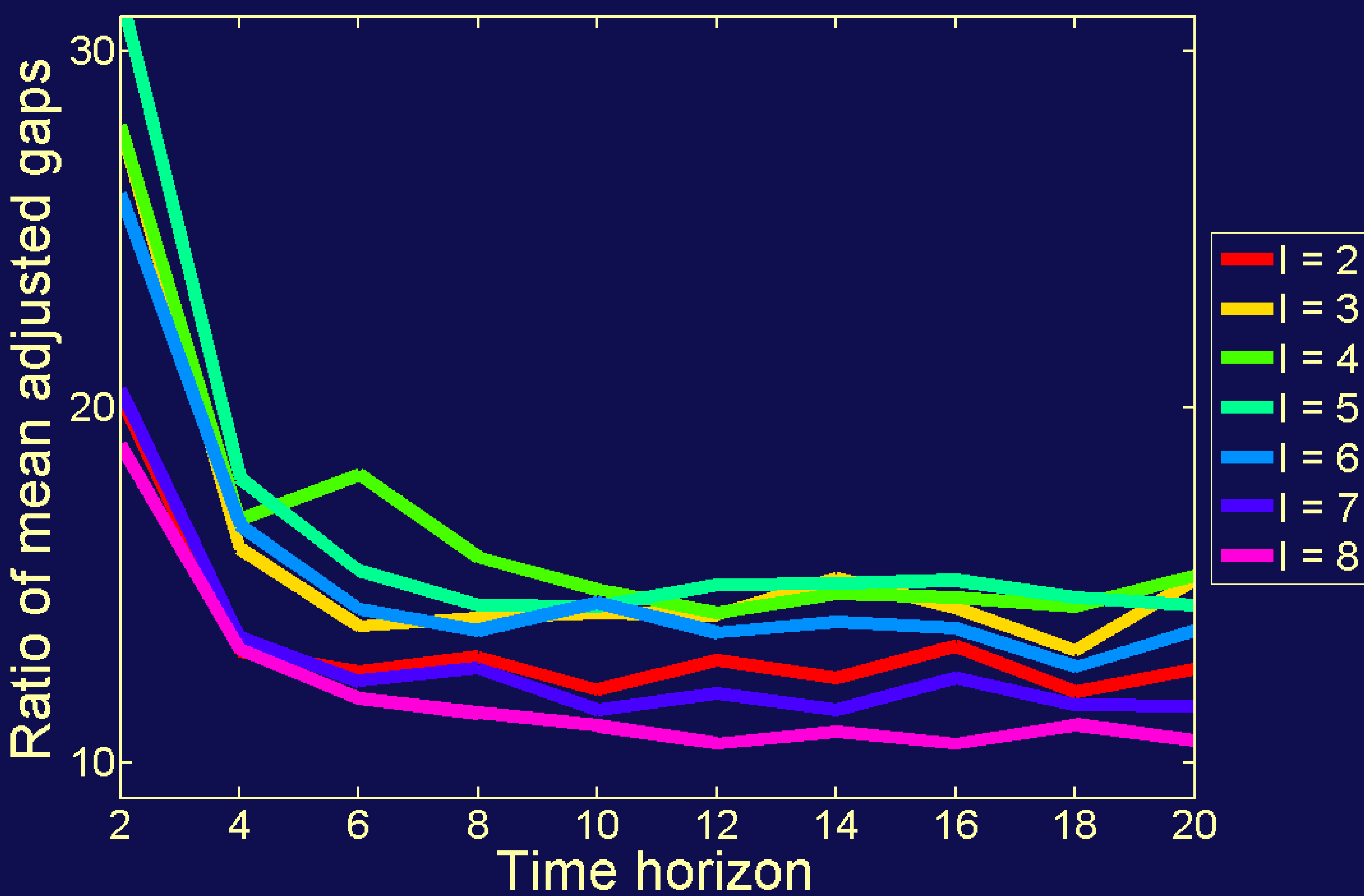
$$\nu_t = \frac{R}{W}\left\{[(1-p)-\alpha(1-\beta p)] - \frac{[(1-q)-\alpha(1-\beta q)](1-\beta p)}{(1-\beta q)+(\beta q-\beta p)(\beta p)^{t-1}}\right\}$$

- J. & Niño-Mora (2007), J. (2009, submitted 2011)
- For inventory of K perishable items, the prices can be computed in $\mathcal{O}(KT)$
- Graczová & J. (in preparation 2011)

Performance of Knapsack Rule



Performance of Greedy vs Knapsack Rule



Challenges

- Modeling
  - ...if modeling were as easy as mathematics...
- Proving (near-)optimality of greedy rules
  - asymptotic optimality proved for symmetric case, as number of competitors and resource capacity grow
- Incorporation of risk aversion, etc.

Thank you for your attention