Online Resource Allocation Problems, with Applications to Revenue Management

David Simchi-Levi (MIT)

Papers

1. Tight Weight-dependent Competitive Ratios for Online Matching Problems
   - joint work with Will Ma
2. Dynamic Recommendation at Checkout under Inventory Constraint
   - joint work with Xi Chen, Will Ma, and Linwei Xin

Papers

1. **Tight Weight-dependent Competitive Ratios for Online Matching Problems**
   - **joint work with Will Ma**
2. Dynamic Recommendation at Checkout under Inventory Constraint
   - joint work with Xi Chen, Will Ma, and Linwei Xin

A General Online Resource Allocation Problem

A General Online Resource Allocation Problem

Known at start:

A General Online Resource Allocation Problem

Known at start:

- number of items (resources), n

A General Online Resource Allocation Problem

Known at start:

- number of items (resources), n
- unreplenishable starting inventory of each item i, k_i

A General Online Resource Allocation Problem

Known at start:

- number of items (resources), n
- unreplenishable starting inventory of each item i, k_i
- m_i potential prices of each item i, satisfying $r_i^{(1)} < \ldots < r_i^{(m_i)}$

A General Online Resource Allocation Problem

Known at start:

- number of items (resources), n
- unreplenishable starting inventory of each item i, k_i
- m_i potential prices of each item i, satisfying $r_i^{(1)} < \ldots < r_i^{(m_i)}$

For $t = 1, 2, \ldots$

A General Online Resource Allocation Problem

Known at start:

- number of items (resources), n
- unreplenishable starting inventory of each item i, k_i
- m_i potential prices of each item i, satisfying $r_i^{(1)} < \ldots < r_i^{(m_i)}$

For $t = 1, 2, \ldots$

1. **Customer t Arrives**:
   - observe $p_{t,i}^{(j)}$, the probability of customer t buying item i at price j, for all i and j (can be based on customer features or classes)

A General Online Resource Allocation Problem

Known at start:

- number of items (resources), n
- unreplenishable starting inventory of each item i, k_i
- m_i potential prices of each item i, satisfying $r_i^{(1)} < \ldots < r_i^{(m_i)}$

For $t = 1, 2, \ldots$

1. **Customer t Arrives**:
   - observe $p_{t,i}^{(j)}$, the probability of customer t buying item i at price j, for all i and j (can be based on customer features or classes)
2. **Decision**:
   - offer an item i_t which has not stocked out, at a price j_t, to customer t

A General Online Resource Allocation Problem

Known at start:

- number of items (resources), n
- unreplenishable starting inventory of each item i, k_i
- m_i potential prices of each item i, satisfying $r_i^{(1)} < \ldots < r_i^{(m_i)}$

For $t = 1, 2, \ldots$

1. **Customer t Arrives**:
   - observe $p_{t,i}^{(j)}$, the probability of customer t buying item i at price j, for all i and j (can be based on customer features or classes)
2. **Decision**:
   - offer an item i_t which has not stocked out, at a price j_t, to customer t
3. **Realization**:
   - customer makes purchase with probability $p_{t,i_t}^{(j_t)}$
   - if she purchases, then earn revenue $r_{i_t}^{(j_t)}$ and decrement inventory of i_t

A General Online Resource Allocation Problem

Known at start:

- number of items (resources), n
- unreplenishable starting inventory of each item i, k_i
- m_i potential prices of each item i, satisfying $r_i^{(1)} < \ldots < r_i^{(m_i)}$

For $t = 1, 2, \ldots$

1. **Customer t Arrives**:
   - observe $p_{t,i}^{(j)}$, the probability of customer t buying item i at price j, for all i and j (can be based on customer features or classes)
2. **Decision**:
   - offer an item i_t which has not stocked out, at a price j_t, to customer t
3. **Realization**:
   - customer makes purchase with probability $p_{t,i_t}^{(j_t)}$
   - if she purchases, then earn revenue $r_{i_t}^{(j_t)}$ and decrement inventory of i_t

Extensions:

- can allow an *assortment* of items/prices to be offered to each customer
- can allow an item to have a *continuum* of potential prices
- can allow for *fractional* inventory consumption

Competitive Ratio

Competitive Ratio

- compare algorithm's performance vs. optimum which knows all arrival information at the start

Competitive Ratio

- compare algorithm's performance vs. optimum which knows all arrival information at the start
- develop algorithms whose *competitive ratio*

$$\inf_{\text{instance } \mathcal{I} \text{ (incl. arrivals)}} \frac{\mathbb{E}[\text{ALG}(\mathcal{I})]}{\text{OPT}(\mathcal{I})}$$

is bounded by a constant *independent of the number of items/customers*

Competitive Ratio

- compare algorithm's performance vs. optimum which knows all arrival information at the start
- develop algorithms whose *competitive ratio*

$$\inf_{\text{instance } \mathcal{I} \text{ (incl. arrivals)}} \frac{\mathbb{E}[\text{ALG}(\mathcal{I})]}{\text{OPT}(\mathcal{I})}$$

  is bounded by a constant *independent of the number of items/customers*
- algorithm makes no assumption on arrival sequence

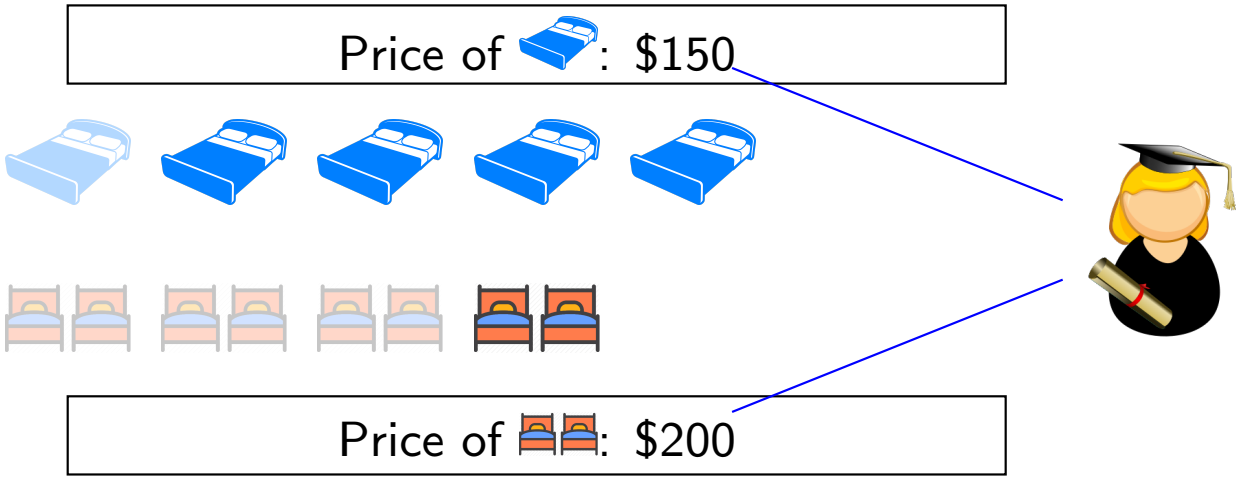
Key Challenges

Key Challenges

1. How do we prioritize between different items to be sold?

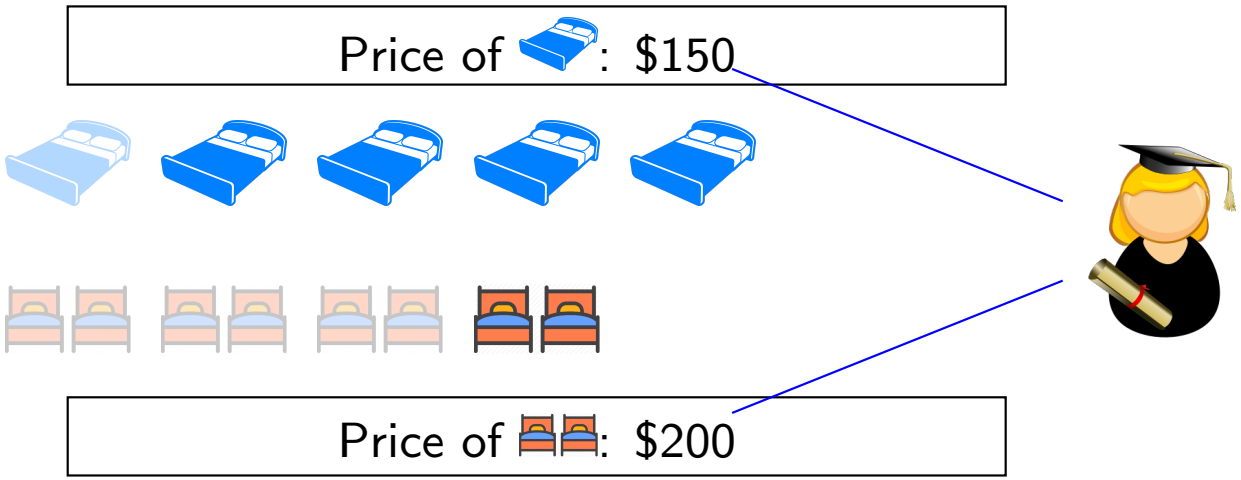
Key Challenges

1. How do we prioritize between different items to be sold?



Key Challenges

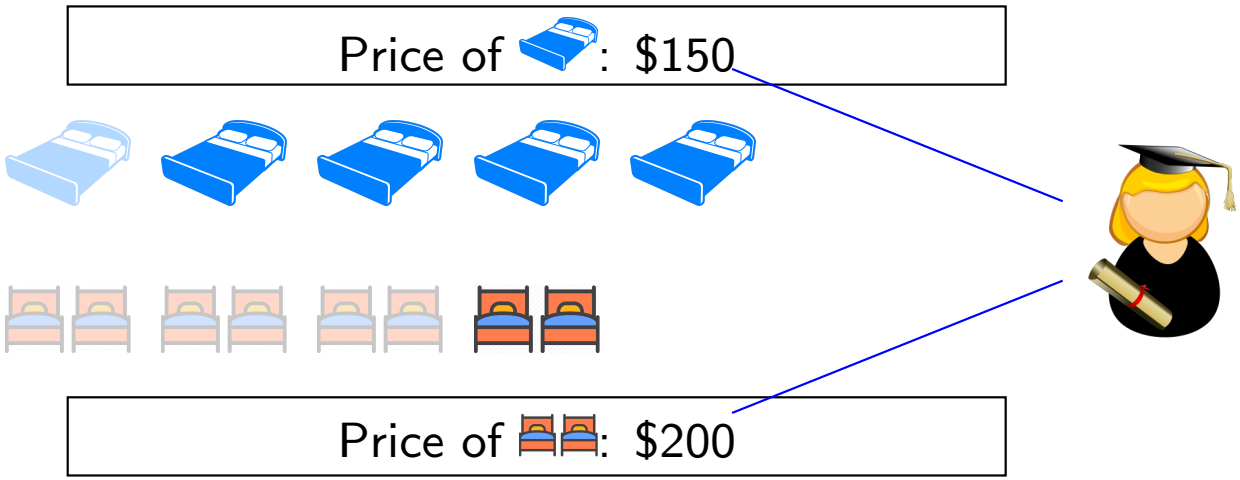
1. How do we prioritize between different items to be sold?



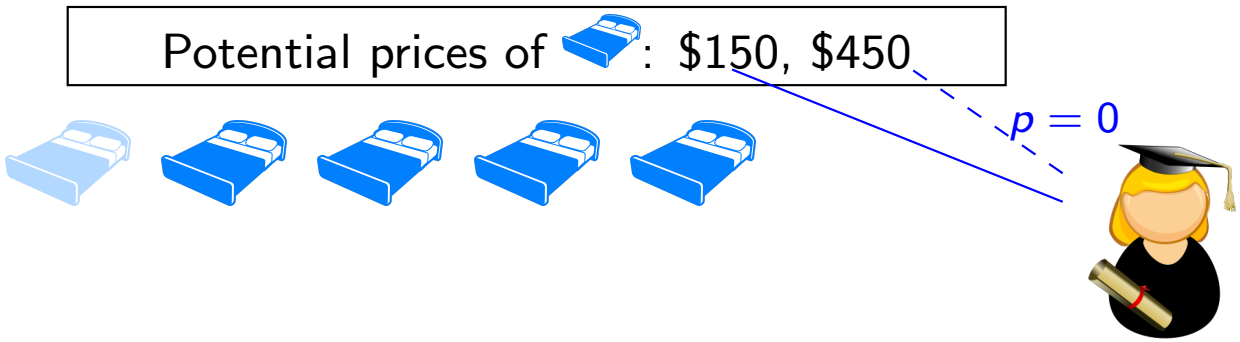
2. When do we reserve the inventory of an item for customers willing to pay higher prices?

Key Challenges

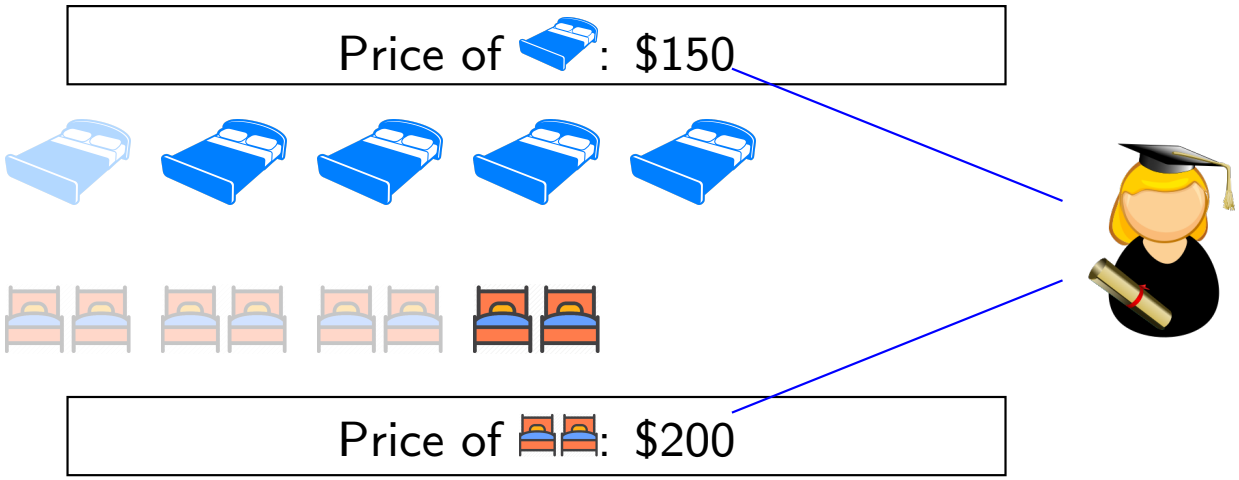
1. How do we prioritize between different items to be sold?



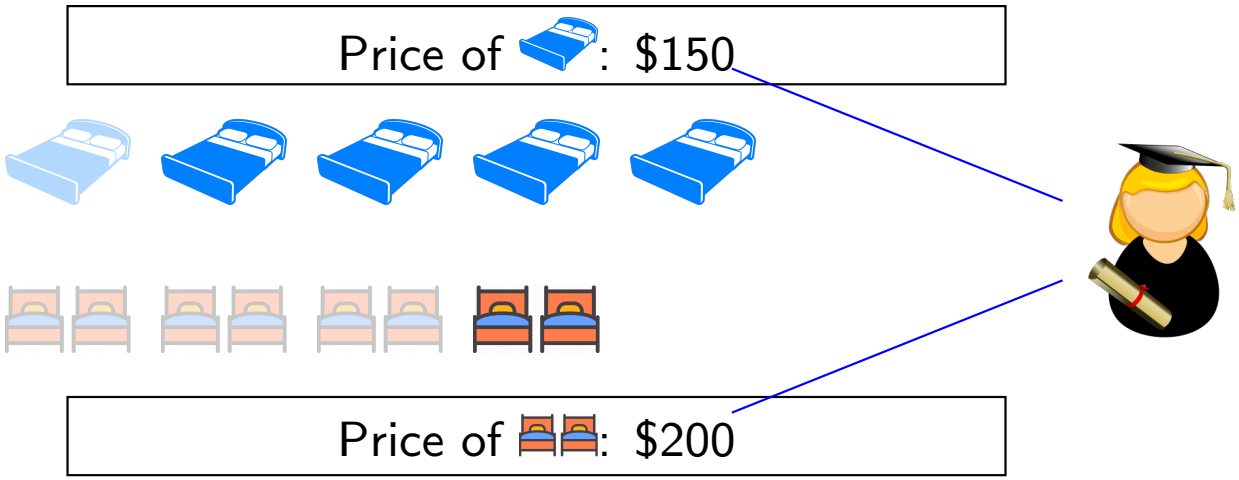
2. When do we reserve the inventory of an item for customers willing to pay higher prices?



Challenge 1: Multiple Items

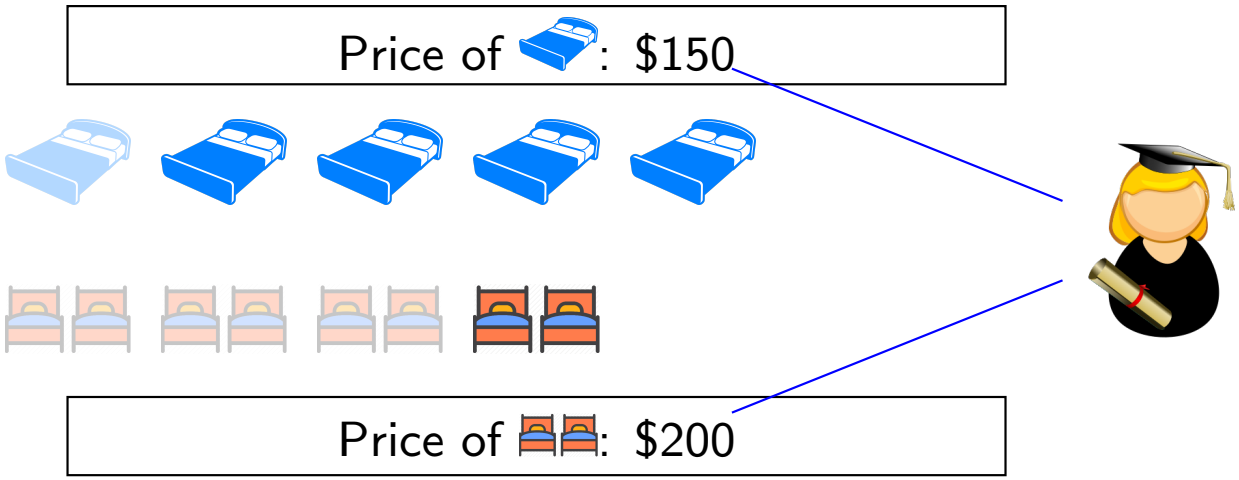


Challenge 1: Multiple Items



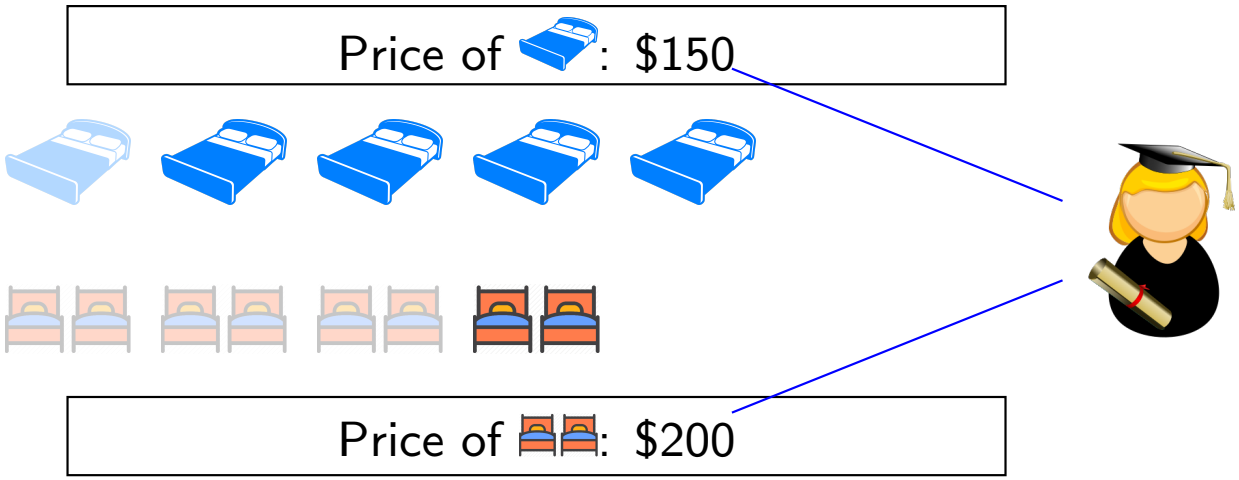
- **Solution**: "Inventory Balancing"

Challenge 1: Multiple Items



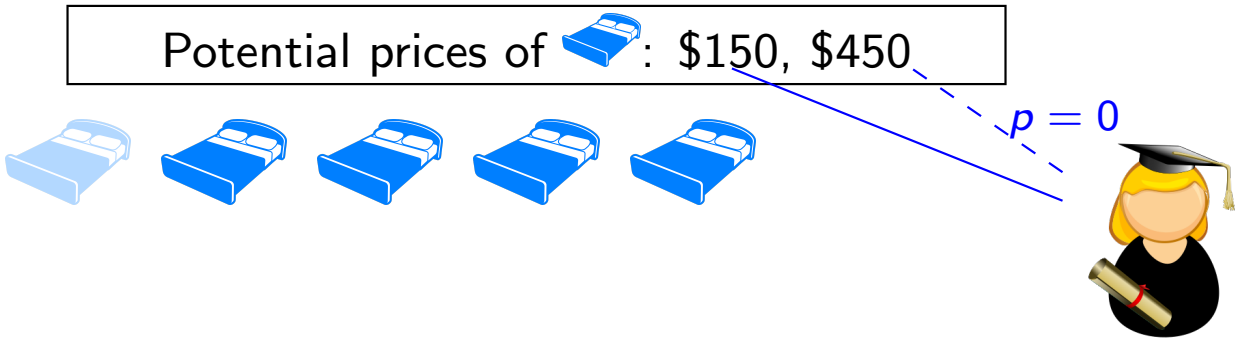
- **Solution**: "Inventory Balancing"
- *balance* the depletion of different items, relative to their prices

Challenge 1: Multiple Items

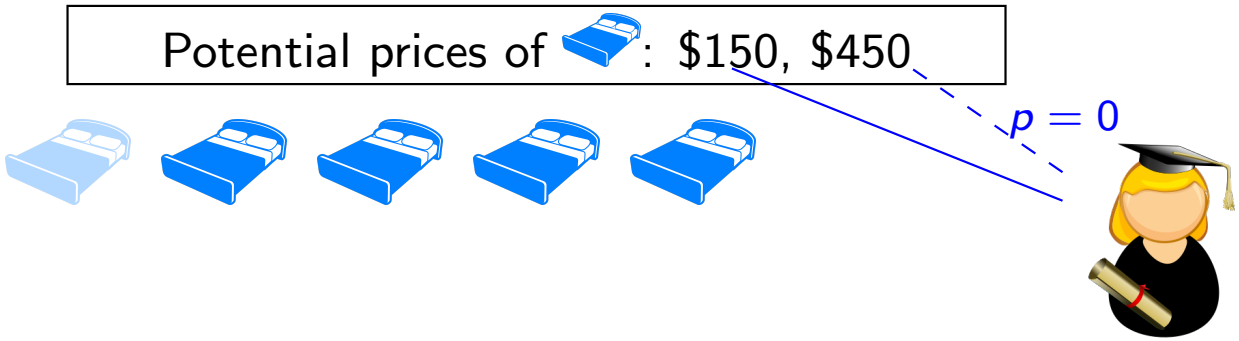


- **Solution**: "Inventory Balancing"
- *balance* the depletion of different items, relative to their prices
- idea has been used in the:
  - online *b*-matching problem (Kalyanasundaram/Pruhs '00)
  - Adwords problem (Mehta et al. '05, Buchbinder/Jain/Naor '07)
  - personalized assortment problem (Golrezaei/Nazerzadeh/Rusmevichientong '14)

Challenge 2: Multiple Prices

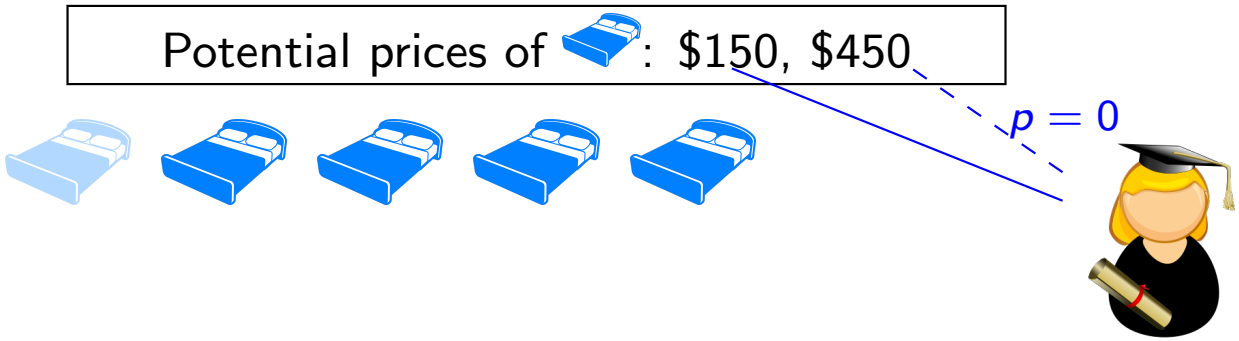


Challenge 2: Multiple Prices



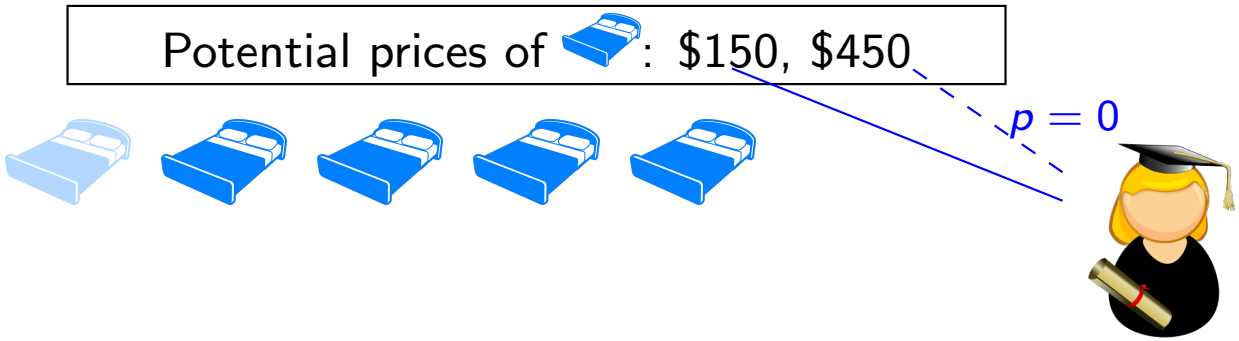
- **Solution**: "Booking Limits"

Challenge 2: Multiple Prices



- **Solution**: "Booking Limits"
- stop offering the item at the lower price after some *fraction* has been sold

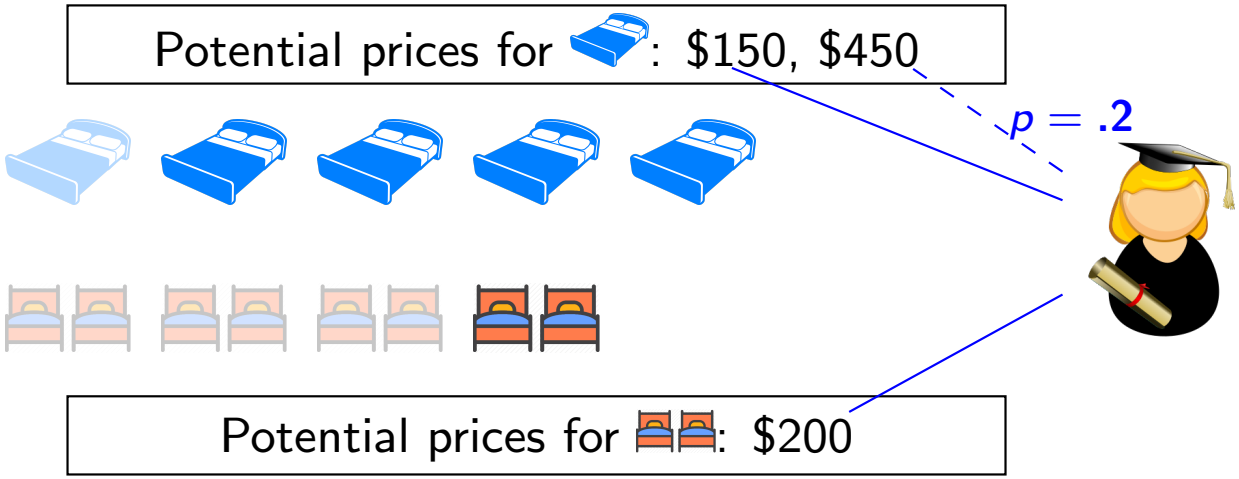
Challenge 2: Multiple Prices



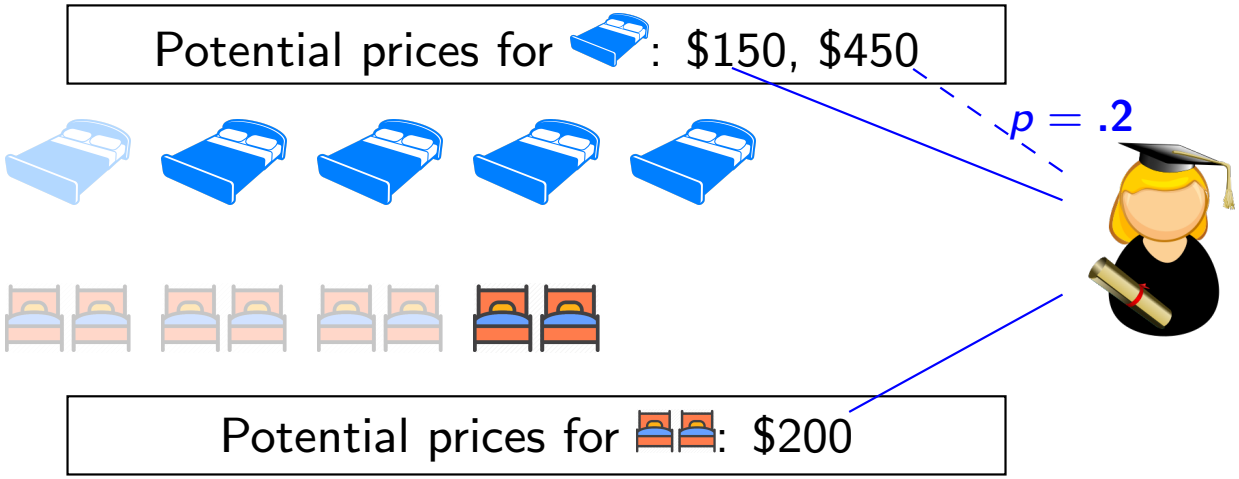
- **Solution**: "Booking Limits"
- stop offering the item at the lower price after some *fraction* has been sold
- idea has been used in the:
  - one-way-trading problem (Lavi/Nisan '00, El-Yaniv et al. '01)
  - single-leg booking problem (Ball/Queyranne '09)
  - online dynamic pricing problem (Ma/S.-L./Teo '17)

Combining the Challenges

Combining the Challenges



Combining the Challenges

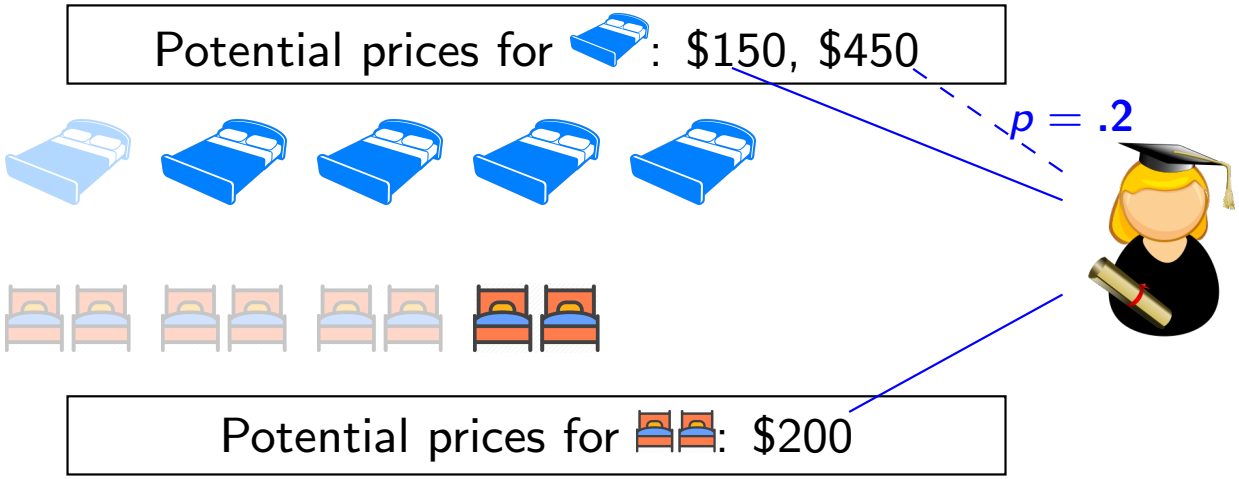


- **Bid-price Control Policy**: for each customer t, offer the item i and price j maximizing the expected "profit"

$$p_{t,i}^{(j)}(r_i^{(j)} - \lambda_i),$$

where λ_i is the "cost", or bid price, for one unit of inventory of i

Combining the Challenges



- **Bid-price Control Policy**: for each customer t, offer the item i and price j maximizing the expected "profit"

$$p_{t,i}^{(j)}(r_i^{(j)} - \lambda_i),$$

where λ_i is the "cost", or bid price, for one unit of inventory of i

- **Key Question**: what is the value of λ_i?

The Value of Inventory

Bid-price Control Policy:

$$\max_{i,j} p_{t,i}^{(j)}(r_i^{(j)} - \lambda_i)$$

The Value of Inventory

Bid-price Control Policy:

$$\max_{i,j} p_{t,i}^{(j)}(r_i^{(j)} - \lambda_i)$$

Typical definition of λ_i:

The Value of Inventory

Bid-price Control Policy:

$$\max_{i,j} p_{t,i}^{(j)}(r_i^{(j)} - \lambda_i)$$

Typical definition of λ_i:

- solve a deterministic LP based on the forecasted demand over the remaining time horizon

The Value of Inventory

Bid-price Control Policy:

$$\max_{i,j} p_{t,i}^{(j)}(r_i^{(j)} - \lambda_i)$$

Typical definition of λ_i:

- solve a deterministic LP based on the forecasted demand over the remaining time horizon
- λ_i is the "reduced cost" of the inventory constraint on item i

The Value of Inventory

Bid-price Control Policy:

$$\max_{i,j} p_{t,i}^{(j)}(r_i^{(j)} - \lambda_i)$$

Typical definition of λ_i:

- solve a deterministic LP based on the forecasted demand over the remaining time horizon
- λ_i is the "reduced cost" of the inventory constraint on item i

Our definition of λ_i:

The Value of Inventory

Bid-price Control Policy:

$$\max_{i,j} p_{t,i}^{(j)}(r_i^{(j)} - \lambda_i)$$

Typical definition of λ_i:

- solve a deterministic LP based on the forecasted demand over the remaining time horizon
- λ_i is the "reduced cost" of the inventory constraint on item i

Our definition of λ_i:

- let w_i be the fraction of item i's starting inventory already sold

The Value of Inventory

Bid-price Control Policy:

$$\max_{i,j} p_{t,i}^{(j)}(r_i^{(j)} - \lambda_i)$$

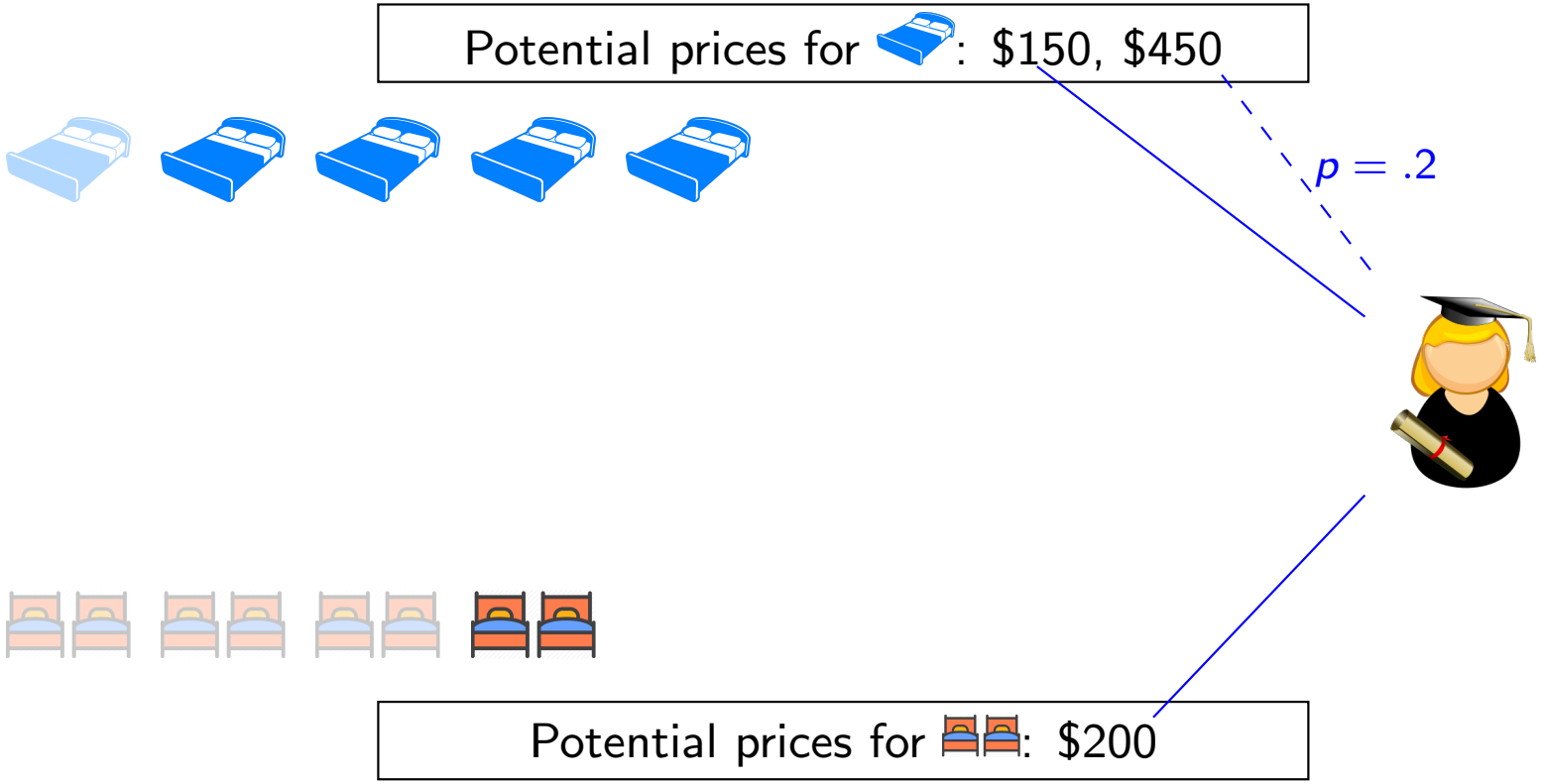
Typical definition of λ_i:

- solve a deterministic LP based on the forecasted demand over the remaining time horizon
- λ_i is the "reduced cost" of the inventory constraint on item i

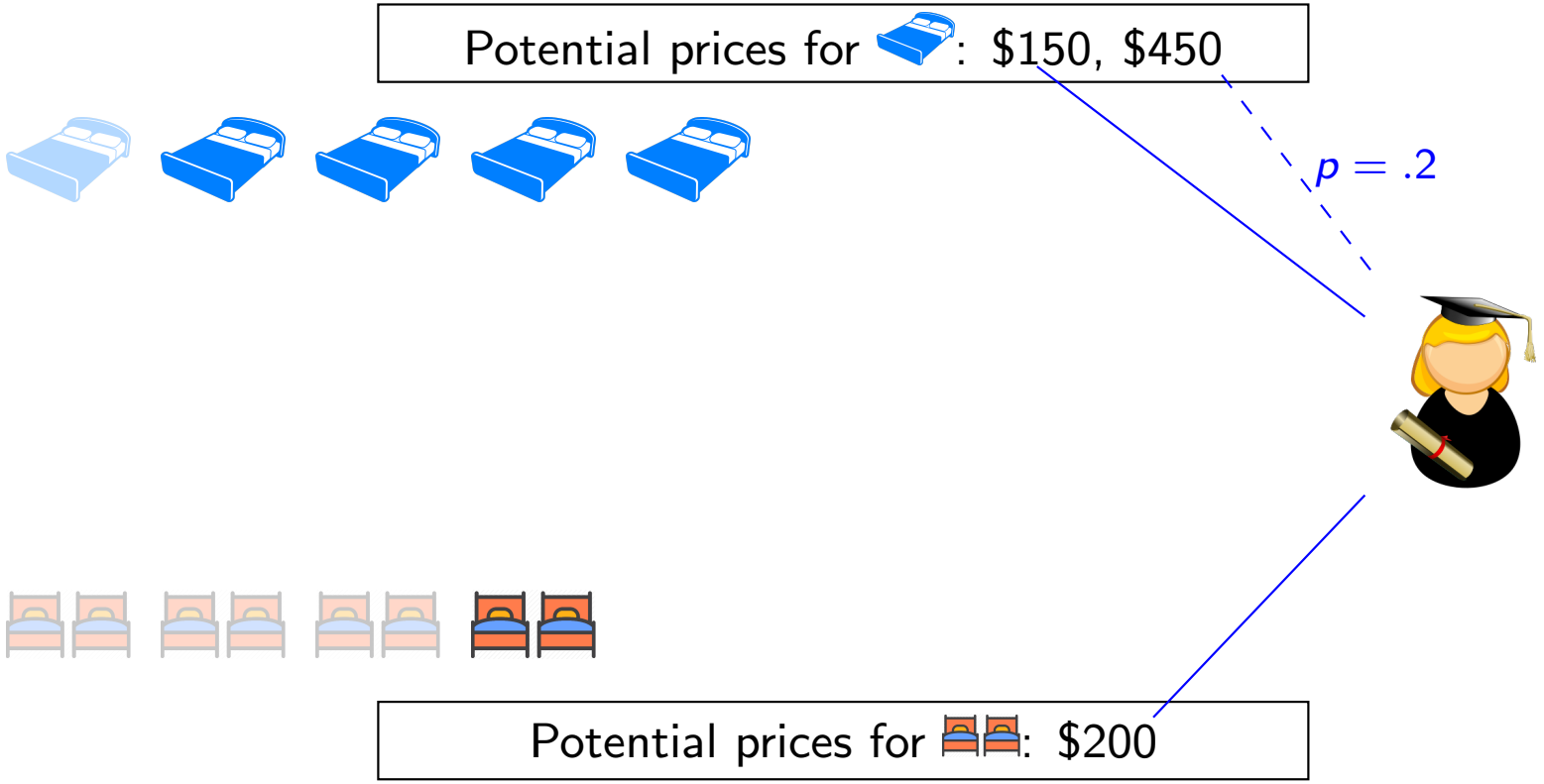
Our definition of λ_i:

- let w_i be the fraction of item i's starting inventory already sold
- $\lambda_i = \Phi_i(w_i)$, where Φ_i is an increasing function

Returning to the Example

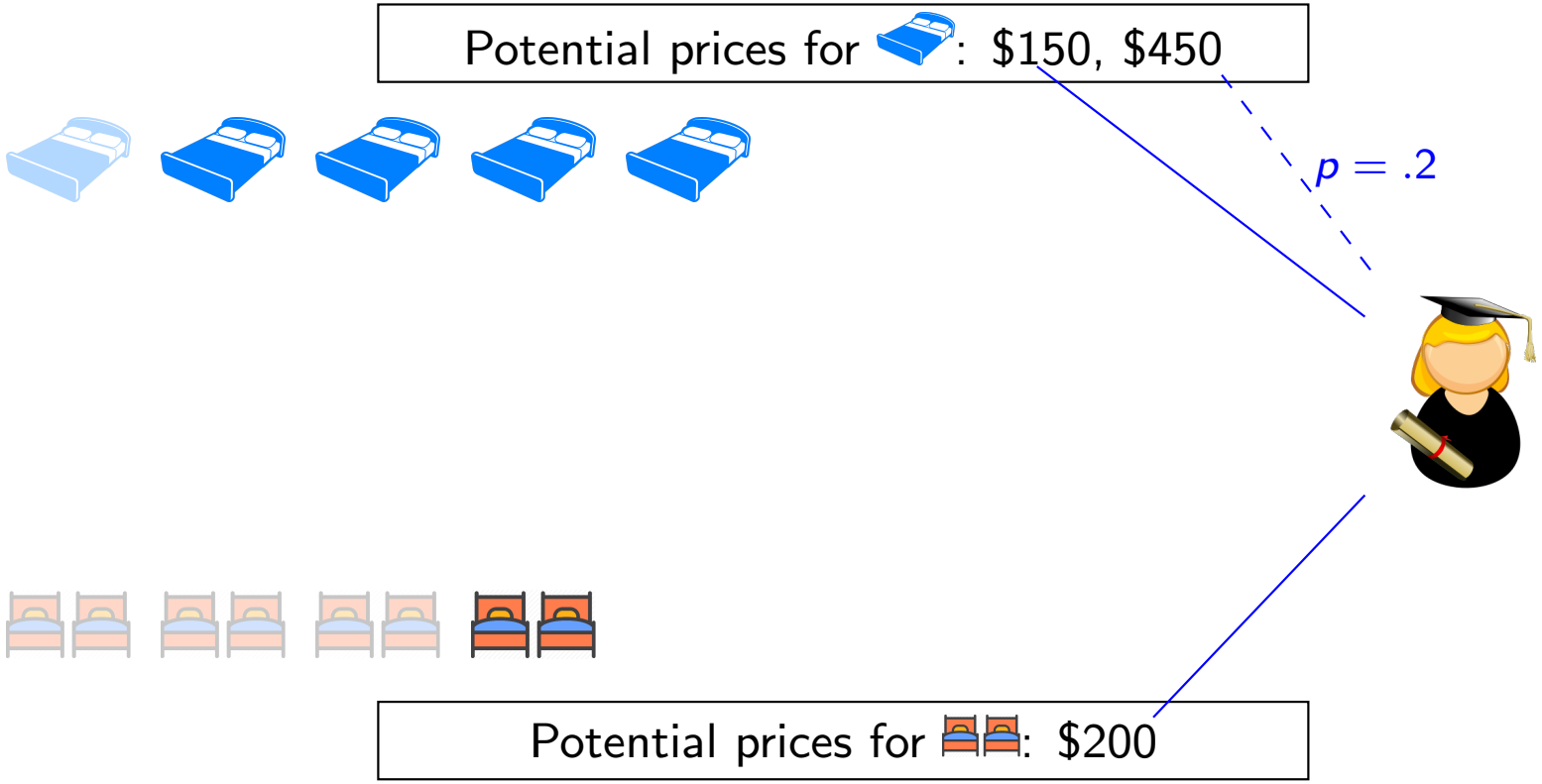


Returning to the Example



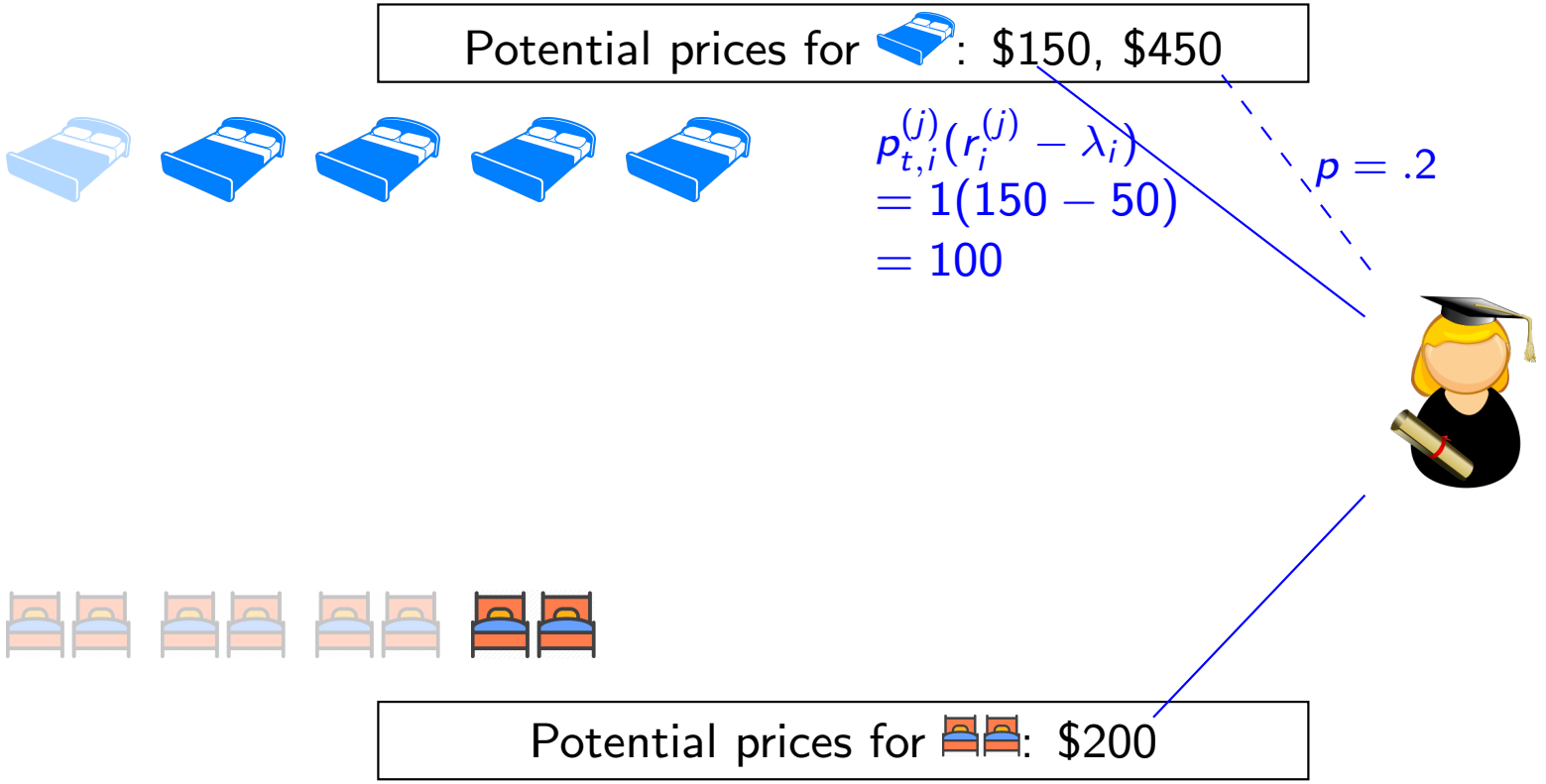
- value of is low: $\lambda_i = \Phi_i(\frac{1}{5}) \approx 50$ (most units still remaining)

Returning to the Example



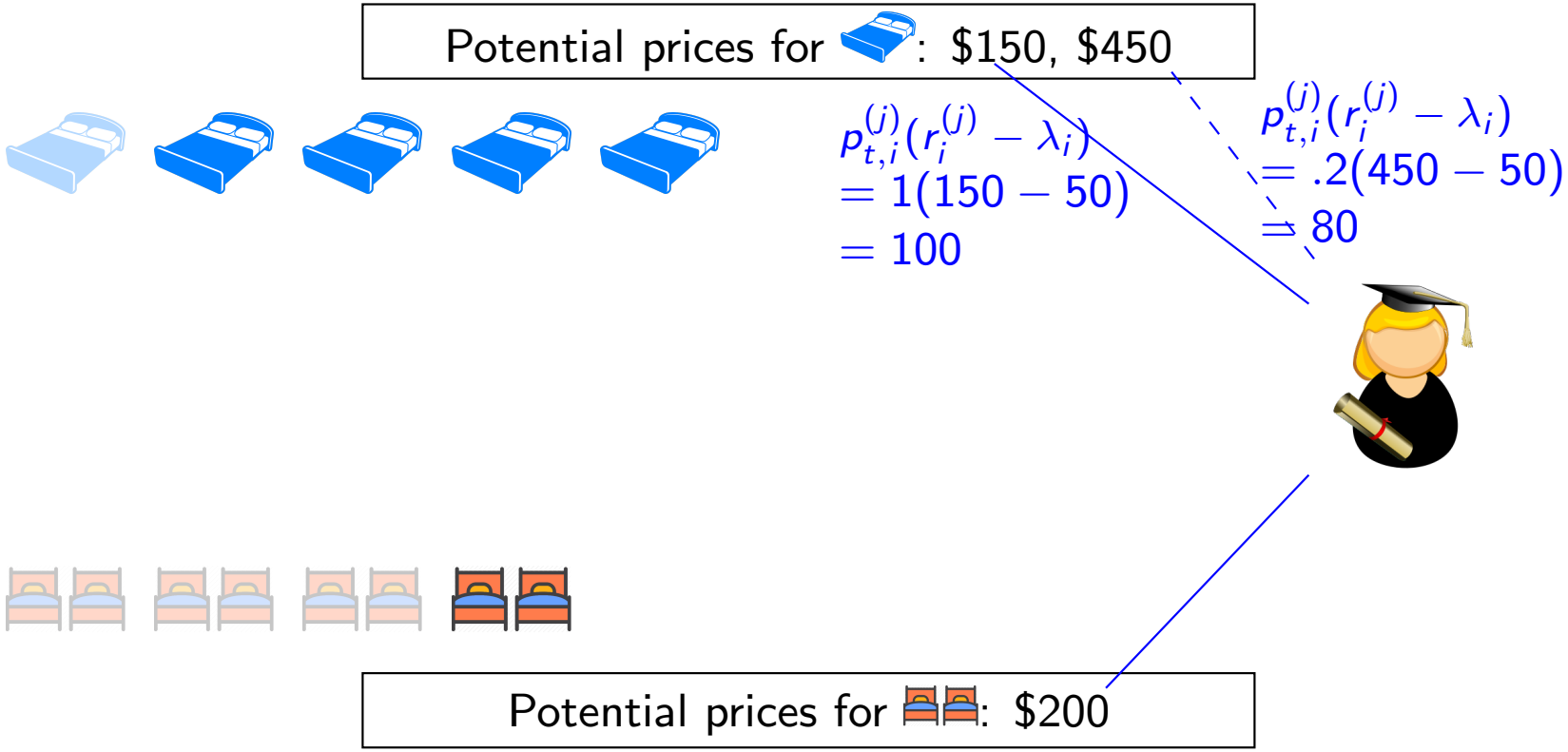
- value of is low: $\lambda_i = \Phi_i(\frac{1}{5}) \approx 50$ (most units still remaining)
- value of is high: $\lambda_i = \Phi_i(\frac{3}{4}) \approx 150$ (most units already sold)

Returning to the Example



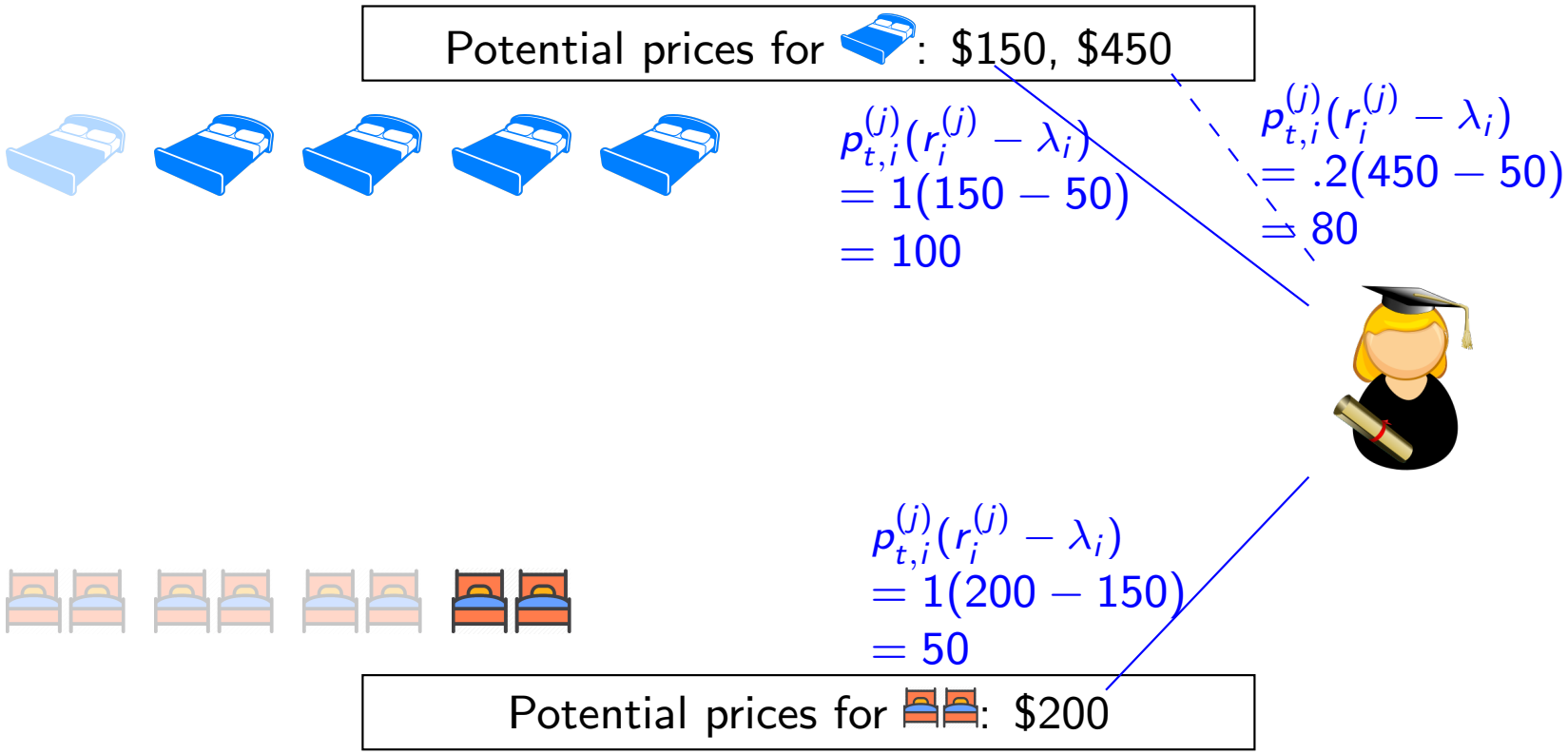
- value of is low: $\lambda_i = \Phi_i(\frac{1}{5}) \approx 50$ (most units still remaining)
- value of is high: $\lambda_i = \Phi_i(\frac{3}{4}) \approx 150$ (most units already sold)

Returning to the Example



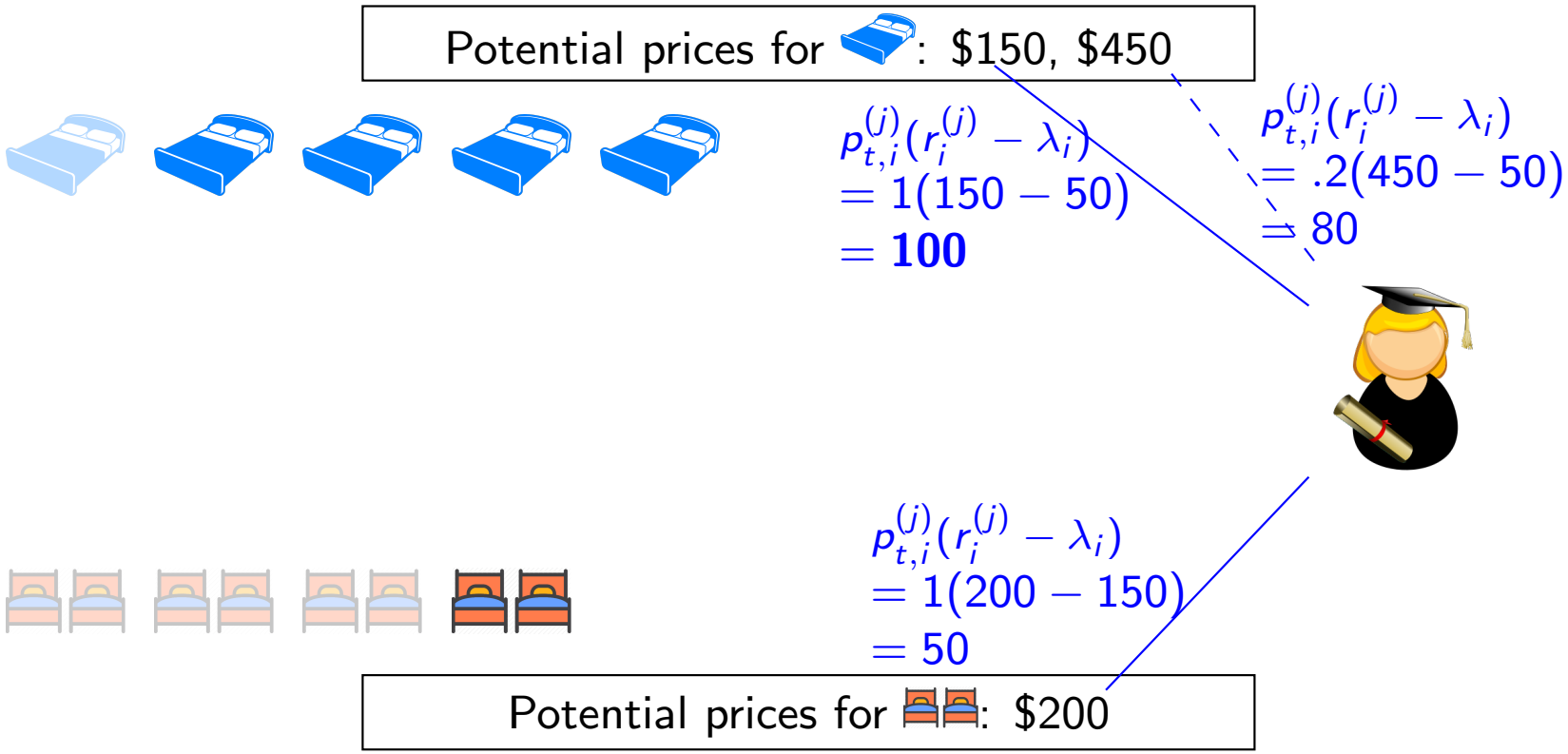
- value of is low: $\lambda_i = \Phi_i(\frac{1}{5}) \approx 50$ (most units still remaining)
- value of is high: $\lambda_i = \Phi_i(\frac{3}{4}) \approx 150$ (most units already sold)

Returning to the Example



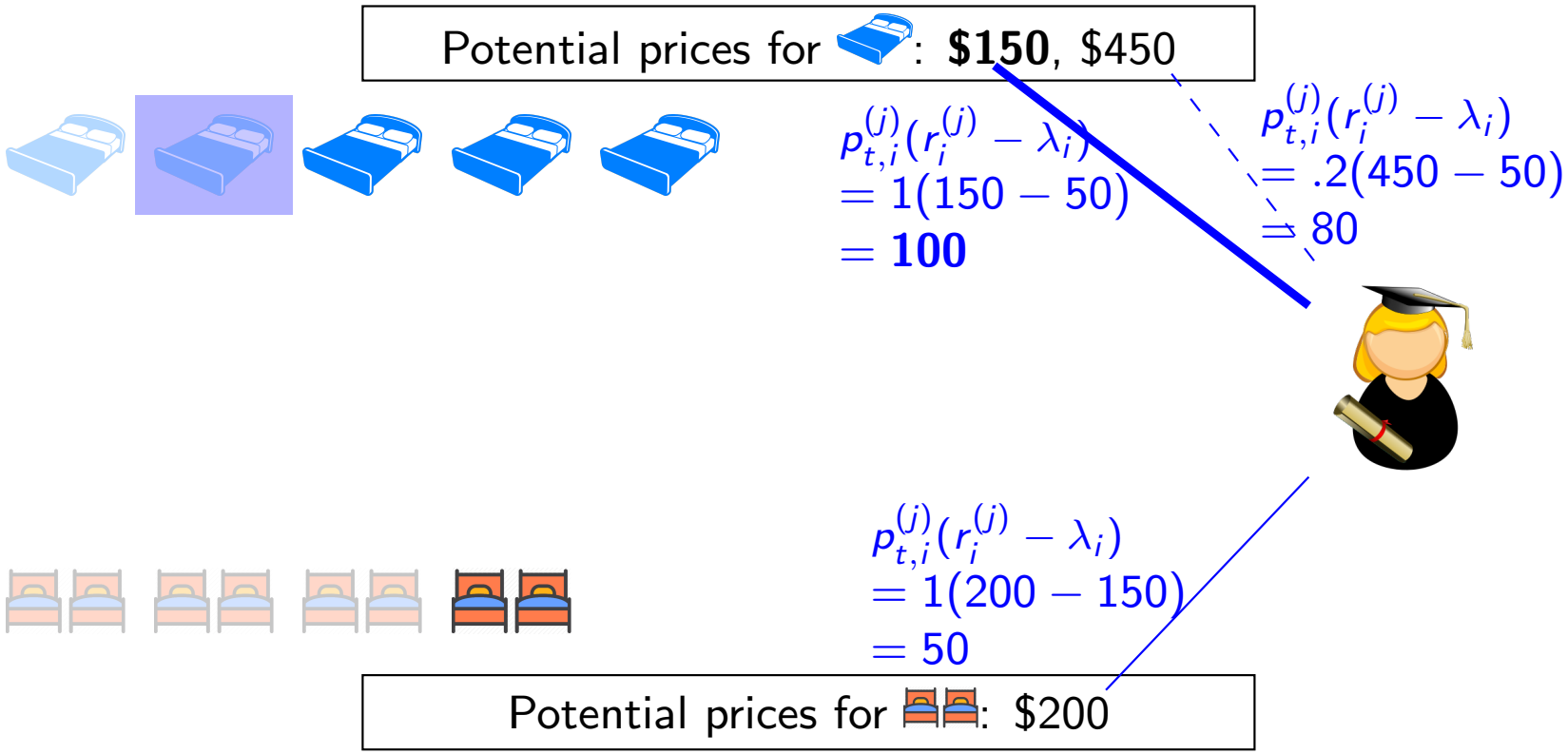
- value of is low: $\lambda_i = \Phi_i(\frac{1}{5}) \approx 50$ (most units still remaining)
- value of is high: $\lambda_i = \Phi_i(\frac{3}{4}) \approx 150$ (most units already sold)

Returning to the Example



- value of is low: $\lambda_i = \Phi_i(\frac{1}{5}) \approx 50$ (most units still remaining)
- value of is high: $\lambda_i = \Phi_i(\frac{3}{4}) \approx 150$ (most units already sold)

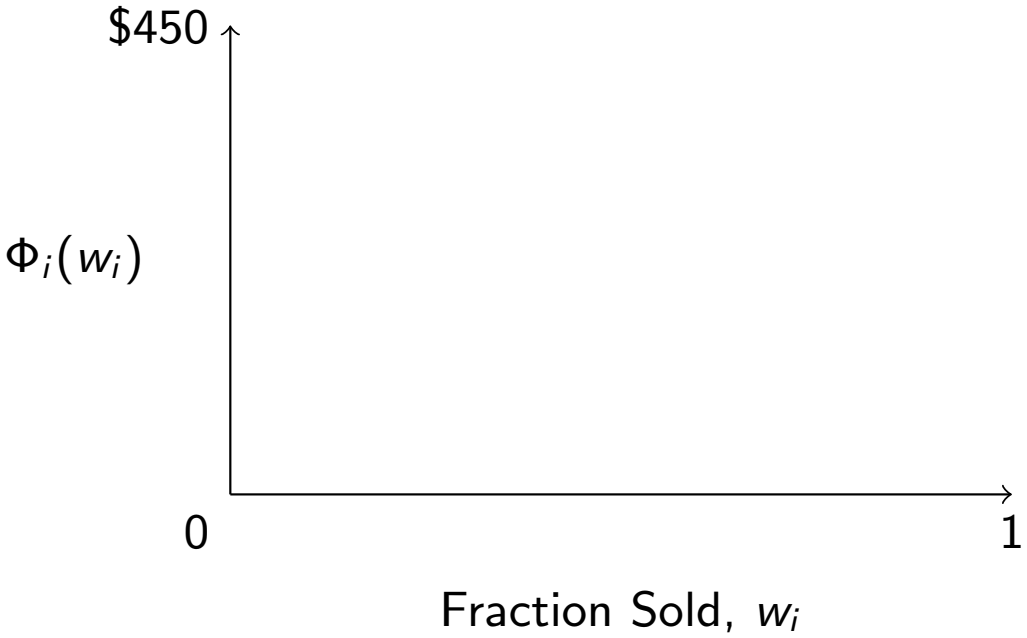
Returning to the Example



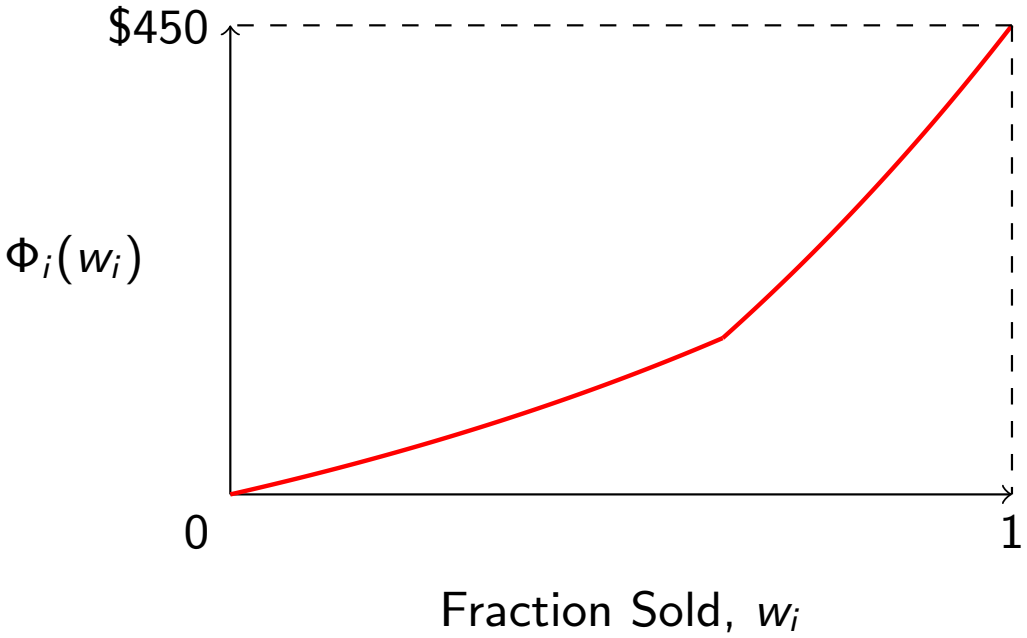
- value of is low: $\lambda_i = \Phi_i(\frac{1}{5}) \approx 50$ (most units still remaining)
- value of is high: $\lambda_i = \Phi_i(\frac{3}{4}) \approx 150$ (most units already sold)

Exact Value Function Φ_i for $i =$ 

Exact Value Function Φ_i for $i =$ 

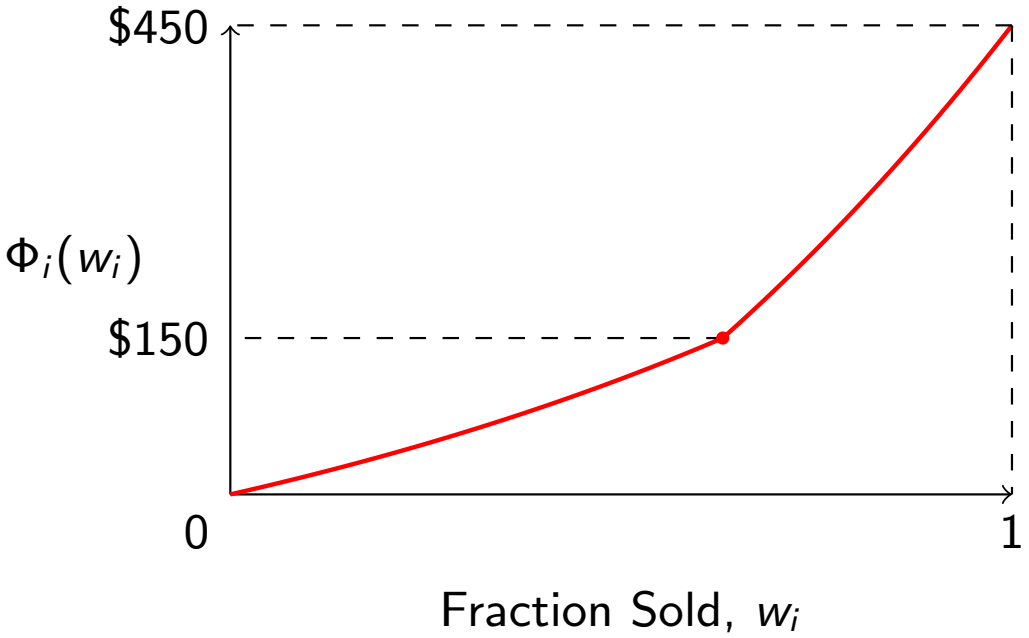


Exact Value Function Φ_i for $i =$ 



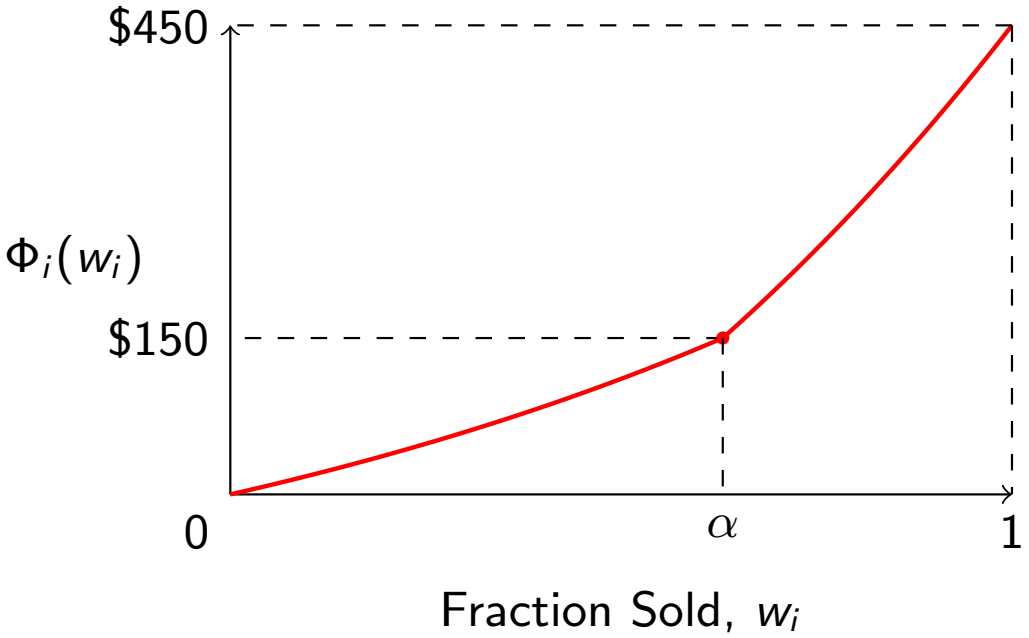
- Φ_i increases from 0 to the maximum price of \$450 over $[0, 1]$

Exact Value Function Φ_i for $i =$ 



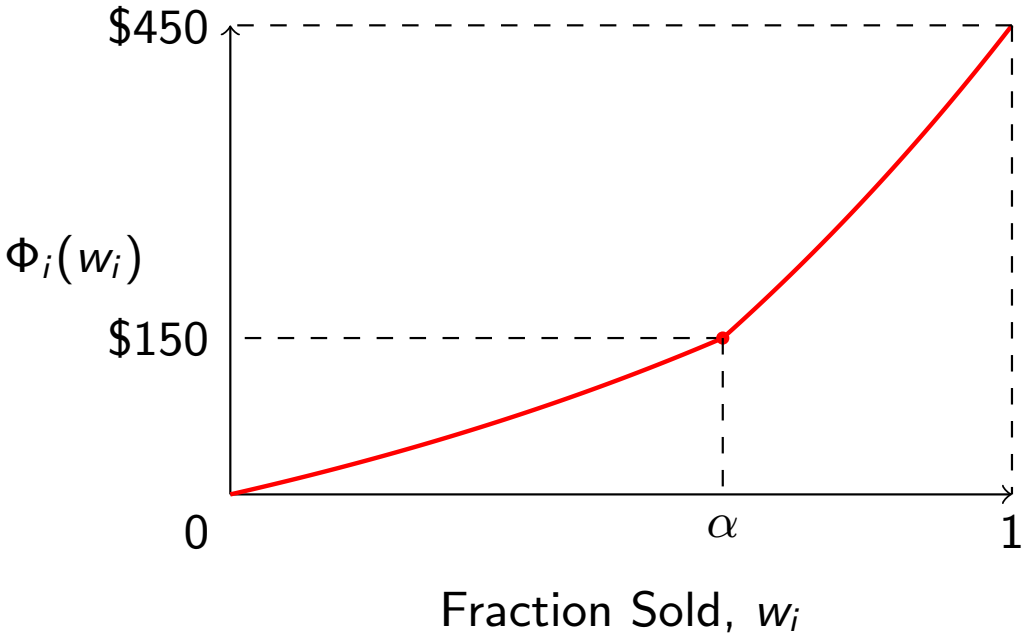
- Φ_i increases from 0 to the maximum price of \$450 over $[0, 1]$
- Φ_i is piecewise-convex

Exact Value Function Φ_i for $i =$ 



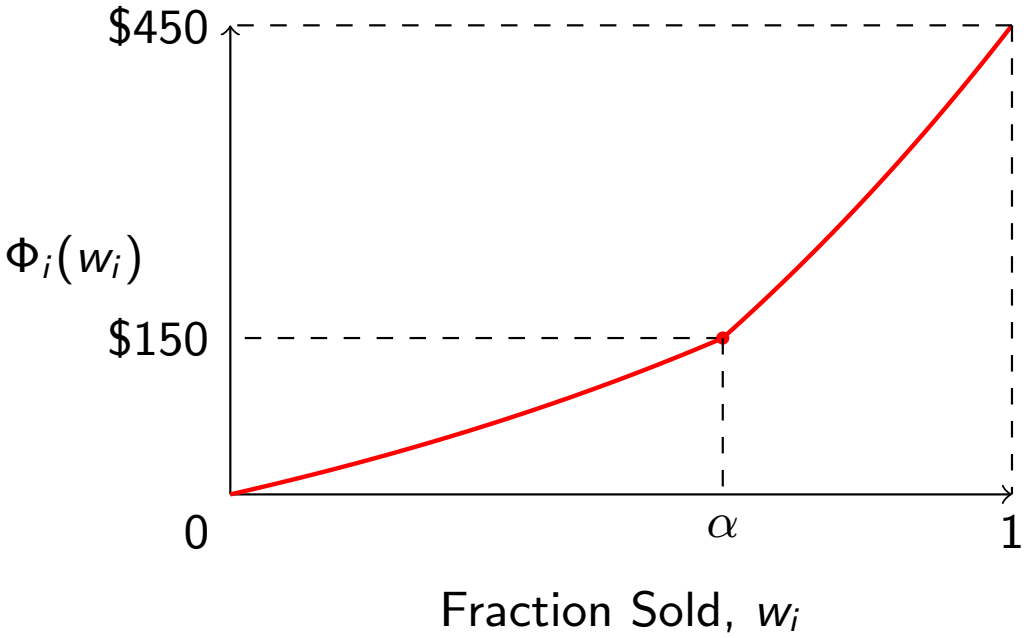
- Φ_i increases from 0 to the maximum price of \$450 over $[0, 1]$
- Φ_i is piecewise-convex
- the value α at which $\Phi_i(\alpha) = \$150$ is the "booking limit" for the lower price

Exact Value Function Φ_i for $i =$ 



- Φ_i increases from 0 to the maximum price of \$450 over $[0, 1]$
- Φ_i is piecewise-convex
- the value α at which $\Phi_i(\alpha) = \$150$ is the "booking limit" for the lower price
- the algorithm is highly disincentivized to offer the item at \$150 as w_i approaches α, and stops completely once $w_i \geq \alpha$

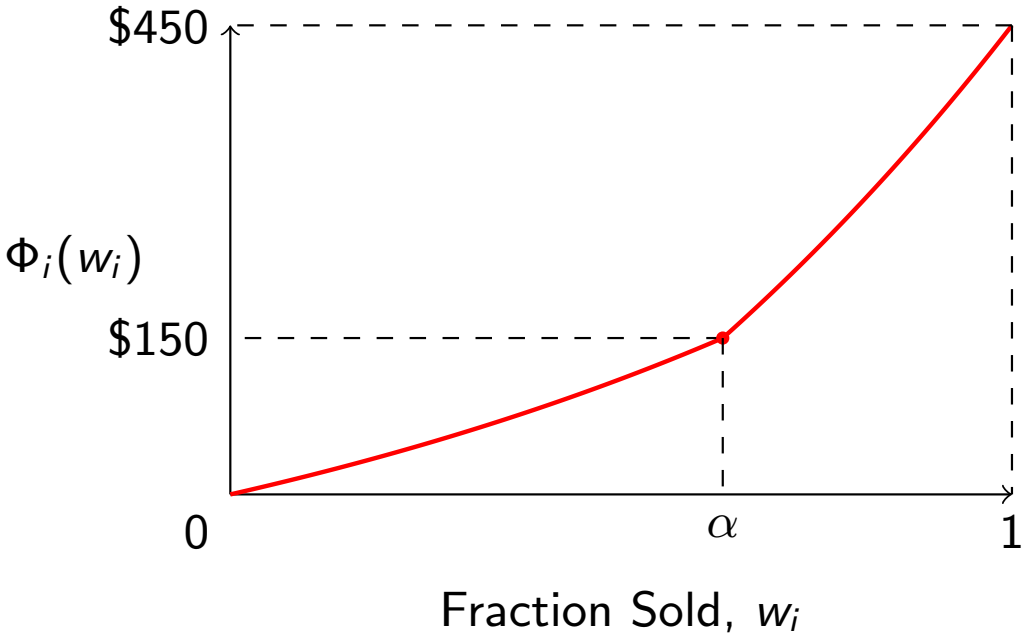
Exact Value Function Φ_i for $i =$ 



- $\alpha = \ln \frac{2(r-1)}{\sqrt{1+4r(r-1)/e}-1}$, where r is the ratio of high price to low price

- Φ_i increases from 0 to the maximum price of \$450 over $[0, 1]$
- Φ_i is piecewise-convex
- the value α at which $\Phi_i(\alpha) = \$150$ is the "booking limit" for the lower price
- the algorithm is highly disincentivized to offer the item at \$150 as w_i approaches α, and stops completely once $w_i \geq \alpha$

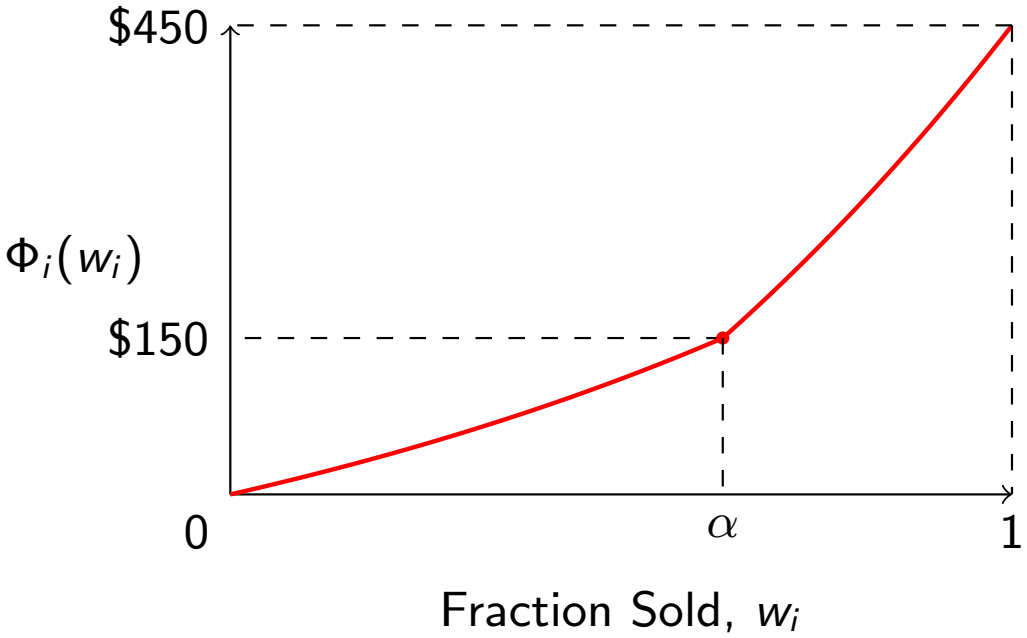
Exact Value Function Φ_i for $i =$ 



- $\alpha = \ln \frac{2(r-1)}{\sqrt{1+4r(r-1)/e}-1}$, where r is the ratio of high price to low price
- different than the optimal *single-item* booking limit $\sigma = \frac{r}{2r-1}$ of Ball/Queyranne

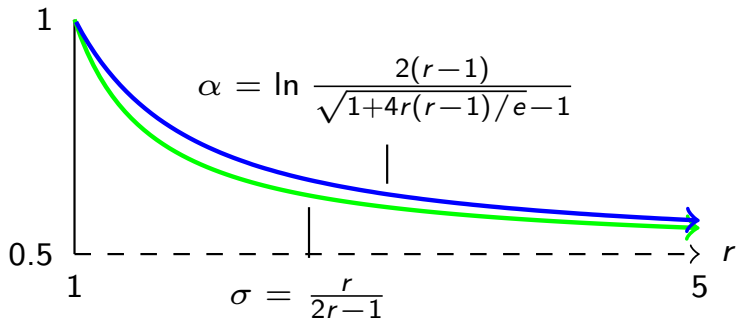
- Φ_i increases from 0 to the maximum price of \$450 over $[0, 1]$
- Φ_i is piecewise-convex
- the value α at which $\Phi_i(\alpha) = \$150$ is the "booking limit" for the lower price
- the algorithm is highly disincentivized to offer the item at \$150 as w_i approaches α, and stops completely once $w_i \geq \alpha$

Exact Value Function Φ_i for $i =$ 



- Φ_i increases from 0 to the maximum price of \$450 over $[0, 1]$
- Φ_i is piecewise-convex
- the value α at which $\Phi_i(\alpha) = \$150$ is the "booking limit" for the lower price
- the algorithm is highly disincentivized to offer the item at \$150 as w_i approaches α, and stops completely once $w_i \geq \alpha$

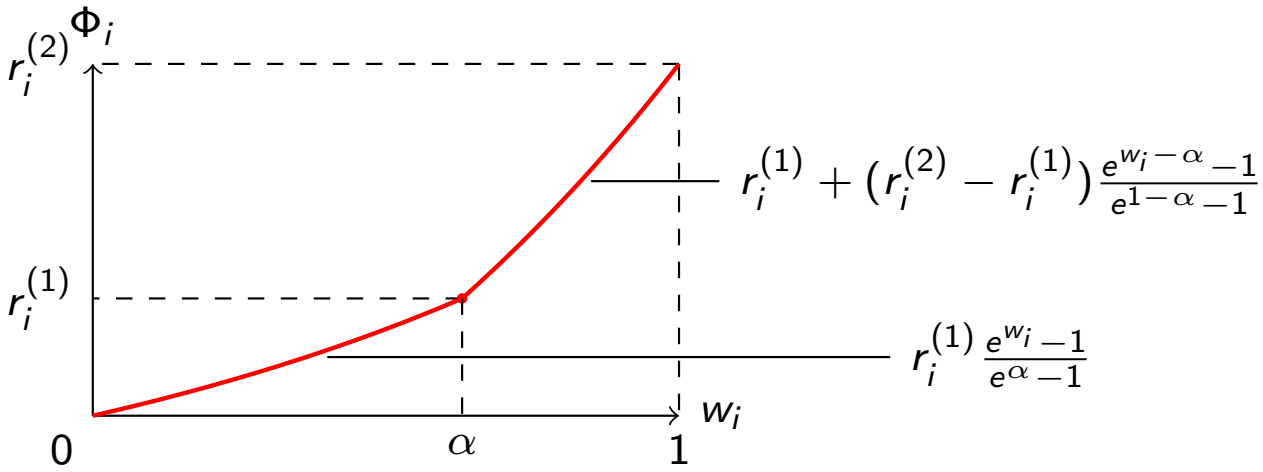
- $\alpha = \ln \frac{2(r-1)}{\sqrt{1+4r(r-1)/e}-1}$, where r is the ratio of high price to low price
- different than the optimal *single-item* booking limit $\sigma = \frac{r}{2r-1}$ of Ball/Queyranne



General Value Function Φ_j for 2 Prices

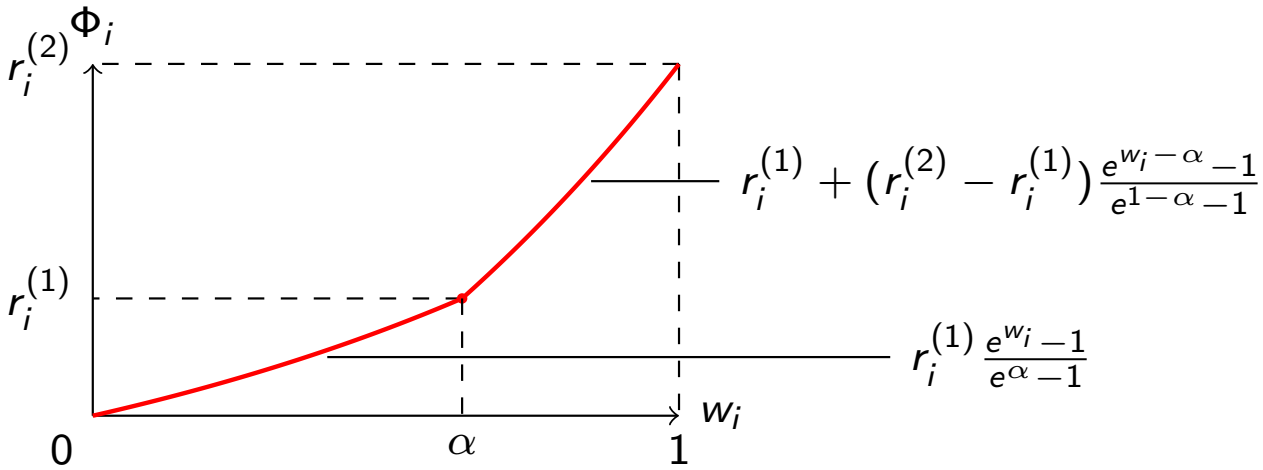
General Value Function Φ_i for 2 Prices

- when $m_i = 2$, Φ_i is piecewise-convex with 2 pieces, separated by α:



General Value Function Φ_i for 2 Prices

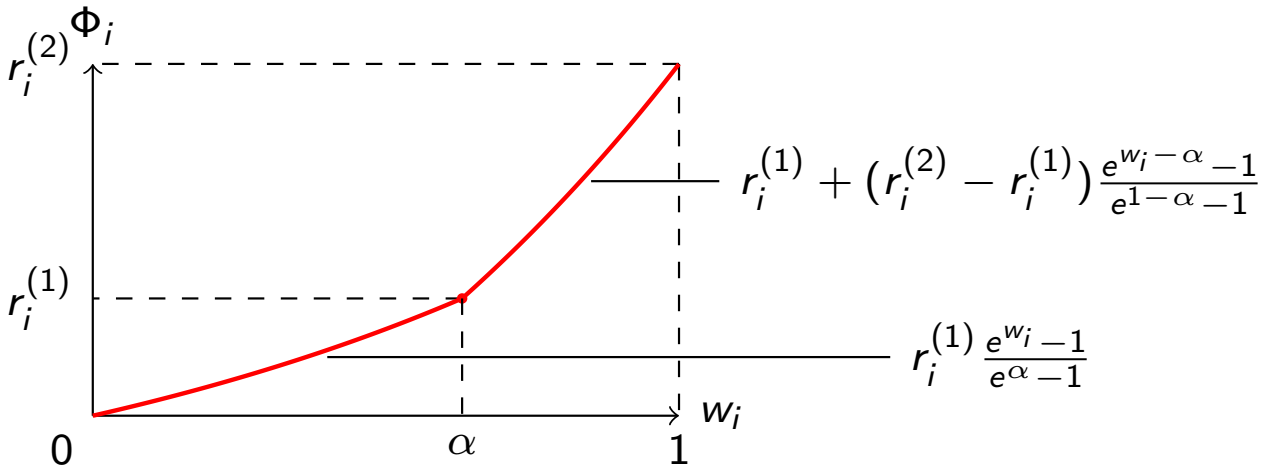
- when $m_i = 2$, Φ_i is piecewise-convex with 2 pieces, separated by α:



- α is given by $\alpha(r_i^{(2)}/r_i^{(1)})$, where $\alpha(r) = \ln \frac{2(r-1)}{\sqrt{1+4r(r-1)/e}-1}$

General Value Function Φ_i for 2 Prices

- when $m_i = 2$, Φ_i is piecewise-convex with 2 pieces, separated by α:



- α is given by $\alpha(r_i^{(2)}/r_i^{(1)})$, where $\alpha(r) = \ln \frac{2(r-1)}{\sqrt{1+4r(r-1)/e}-1}$
- the competitive ratio associated with Φ_i, CR_i, is then $1 - e^{-\alpha}$

General Value Function Φ_j for Multiple Prices

General Value Function Φ_i for Multiple Prices

- for any number m_i of prices, Φ_i is designed, based on $r_i^{(1)}, \ldots, r_i^{(m_i)}$, to maximize CR_i

General Value Function Φ_i for Multiple Prices

- for any number m_i of prices, Φ_i is designed, based on $r_i^{(1)}, \ldots, r_i^{(m_i)}$, to maximize CR_i
- we derive Φ_i by solving a differential equation arising from a primal-dual analysis

General Value Function Φ_i for Multiple Prices

- for any number m_i of prices, Φ_i is designed, based on $r_i^{(1)}, \ldots, r_i^{(m_i)}$, to maximize CR_i
- we derive Φ_i by solving a differential equation arising from a primal-dual analysis
- in general, Φ_i is piecewise-convex with m_i pieces, separated by $m_i - 1$ booking limits $\alpha_i^{(1)}, \ldots, \alpha_i^{(m_i-1)}$

General Value Function Φ_i for Multiple Prices

- for any number m_i of prices, Φ_i is designed, based on $r_i^{(1)}, \ldots, r_i^{(m_i)}$, to maximize CR_i
- we derive Φ_i by solving a differential equation arising from a primal-dual analysis
- in general, Φ_i is piecewise-convex with m_i pieces, separated by $m_i - 1$ booking limits $\alpha_i^{(1)}, \ldots, \alpha_i^{(m_i-1)}$
- $\alpha_i^{(1)}, \ldots, \alpha_i^{(m_i-1)}$ correspond to roots of a degree-m_i polynomial, which can be computed using bisection search

Overall Algorithm

Bid-price Control Policy:

$$\max_{i,j} p_{t,i}^{(j)}(r_i^{(j)} - \lambda_i)$$

Overall Algorithm

Bid-price Control Policy:

$$\max_{i,j} p_{t,i}^{(j)}(r_i^{(j)} - \lambda_i)$$

Having derived $\Phi_1, \ldots, \Phi_n$, *two* different algorithms are possible:

Overall Algorithm

Bid-price Control Policy:

$$\max_{i,j} p_{t,i}^{(j)}(r_i^{(j)} - \lambda_i)$$

Having derived $\Phi_1, \ldots, \Phi_n$, *two* different algorithms are possible:

1. Algorithm 1:
   - $\lambda_i = \Phi_i(w_i)$, where w_i is the fraction of item i's starting inventory sold
   - w_i is dynamically incremented as sales are realized

Overall Algorithm

Bid-price Control Policy:

$$\max_{i,j} p_{t,i}^{(j)}(r_i^{(j)} - \lambda_i)$$

Having derived $\Phi_1, \ldots, \Phi_n$, *two* different algorithms are possible:

1. Algorithm 1:
   - $\lambda_i = \Phi_i(w_i)$, where w_i is the fraction of item i's starting inventory sold
   - w_i is dynamically incremented as sales are realized
2. Algorithm 2:

Overall Algorithm

Bid-price Control Policy:

$$\max_{i,j} p_{t,i}^{(j)}(r_i^{(j)} - \lambda_i)$$

Having derived $\Phi_1, \ldots, \Phi_n$, *two* different algorithms are possible:

1. Algorithm 1:
   - $\lambda_i = \Phi_i(w_i)$, where w_i is the fraction of item i's starting inventory sold
   - w_i is dynamically incremented as sales are realized
2. Algorithm 2:
   - more applicable when the starting inventories k_i are small

Overall Algorithm

Bid-price Control Policy:

$$\max_{i,j} p_{t,i}^{(j)}(r_i^{(j)} - \lambda_i)$$

Having derived $\Phi_1, \ldots, \Phi_n$, *two* different algorithms are possible:

1. Algorithm 1:
   - $\lambda_i = \Phi_i(w_i)$, where w_i is the fraction of item i's starting inventory sold
   - w_i is dynamically incremented as sales are realized
2. Algorithm 2:
   - more applicable when the starting inventories k_i are small
   - assume WOLOG that $k_i = 1$ for all i

Overall Algorithm

Bid-price Control Policy:

$$\max_{i,j} p_{t,i}^{(j)}(r_i^{(j)} - \lambda_i)$$

Having derived $\Phi_1, \ldots, \Phi_n$, *two* different algorithms are possible:

1. Algorithm 1:
   - $\lambda_i = \Phi_i(w_i)$, where w_i is the fraction of item i's starting inventory sold
   - w_i is dynamically incremented as sales are realized
2. Algorithm 2:
   - more applicable when the starting inventories k_i are small
   - assume WOLOG that $k_i = 1$ for all i
   - then $\lambda_i = \Phi_i(W_i)$, where $W_i \sim \text{Unif}[0, 1]$

Overall Algorithm

Bid-price Control Policy:

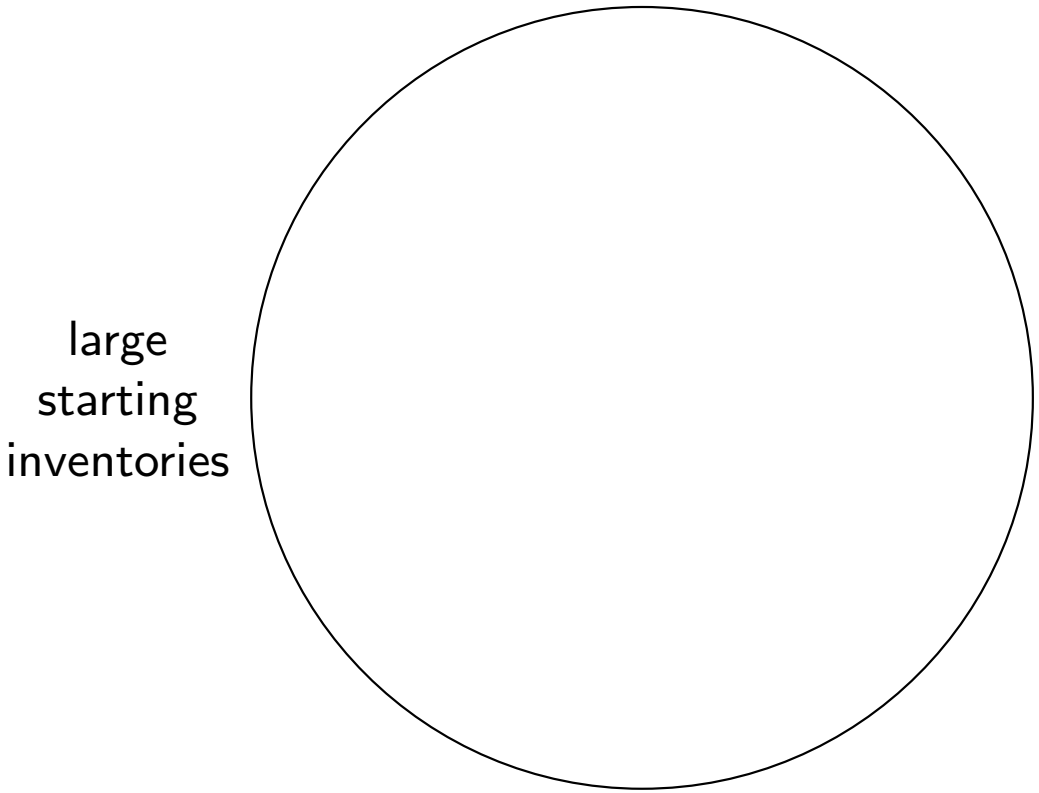
$$\max_{i,j} p_{t,i}^{(j)}(r_i^{(j)} - \lambda_i)$$

Having derived $\Phi_1, \ldots, \Phi_n$, *two* different algorithms are possible:

1. Algorithm 1:
   - $\lambda_i = \Phi_i(w_i)$, where w_i is the fraction of item i's starting inventory sold
   - w_i is dynamically incremented as sales are realized
2. Algorithm 2:
   - more applicable when the starting inventories k_i are small
   - assume WOLOG that $k_i = 1$ for all i
   - then $\lambda_i = \Phi_i(W_i)$, where $W_i \sim \text{Unif}[0, 1]$
   - W_i is a *random seed* which is initialized independently for each i

The Competitive Ratio

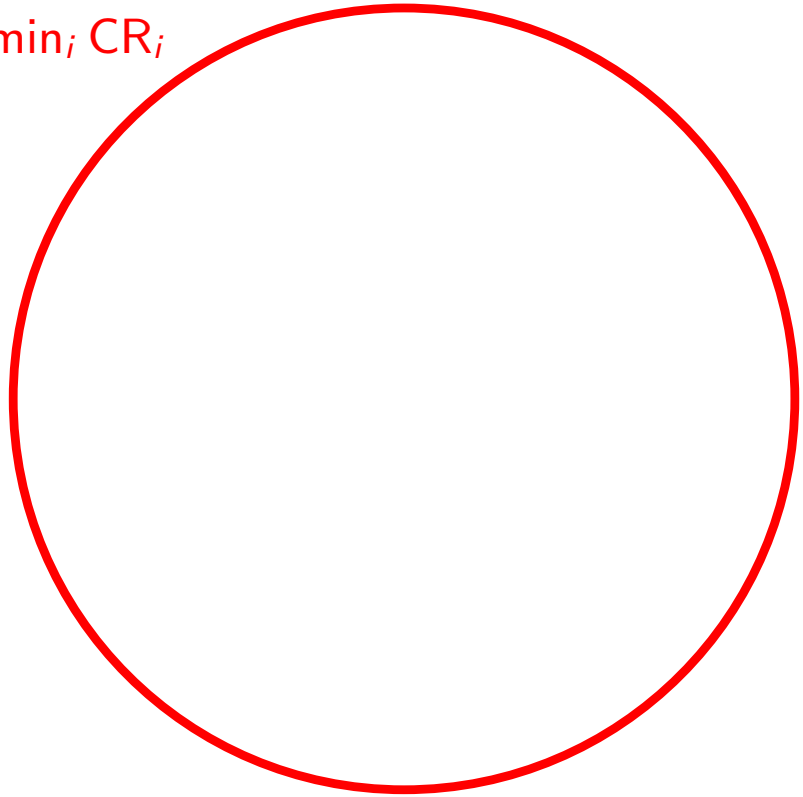
The Competitive Ratio



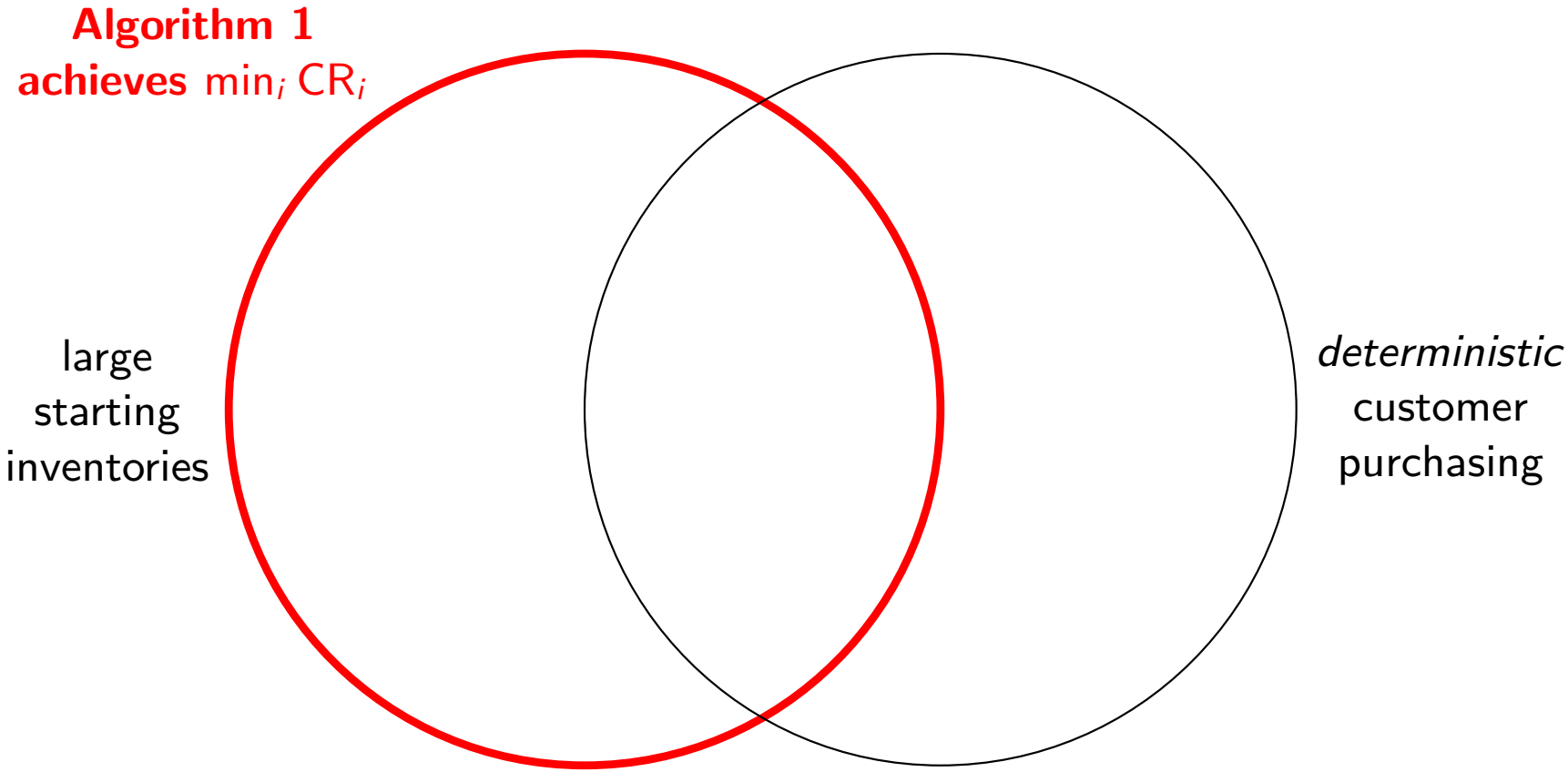
The Competitive Ratio

Algorithm 1
achieves $\min_i CR_i$

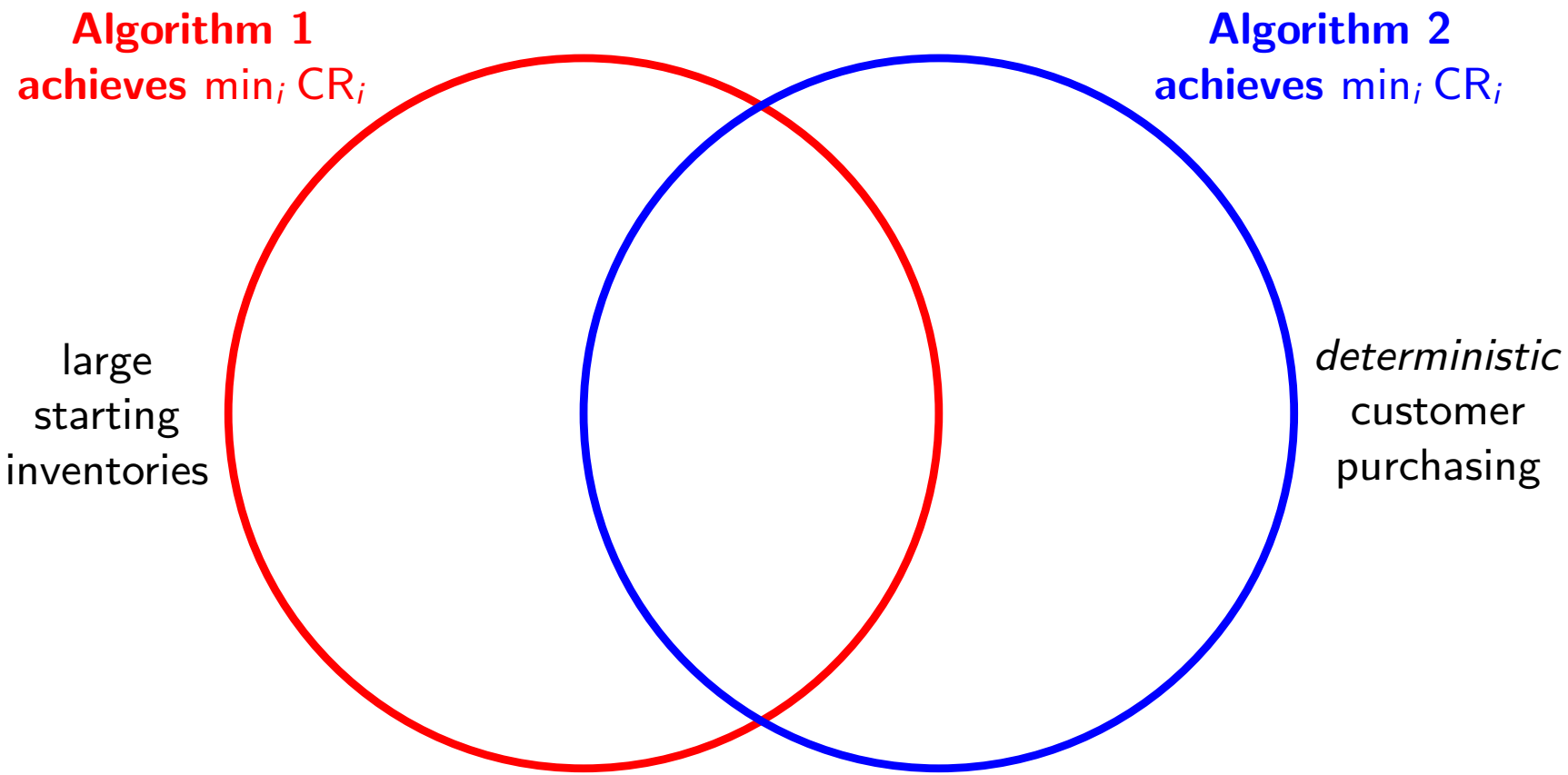
large
starting
inventories



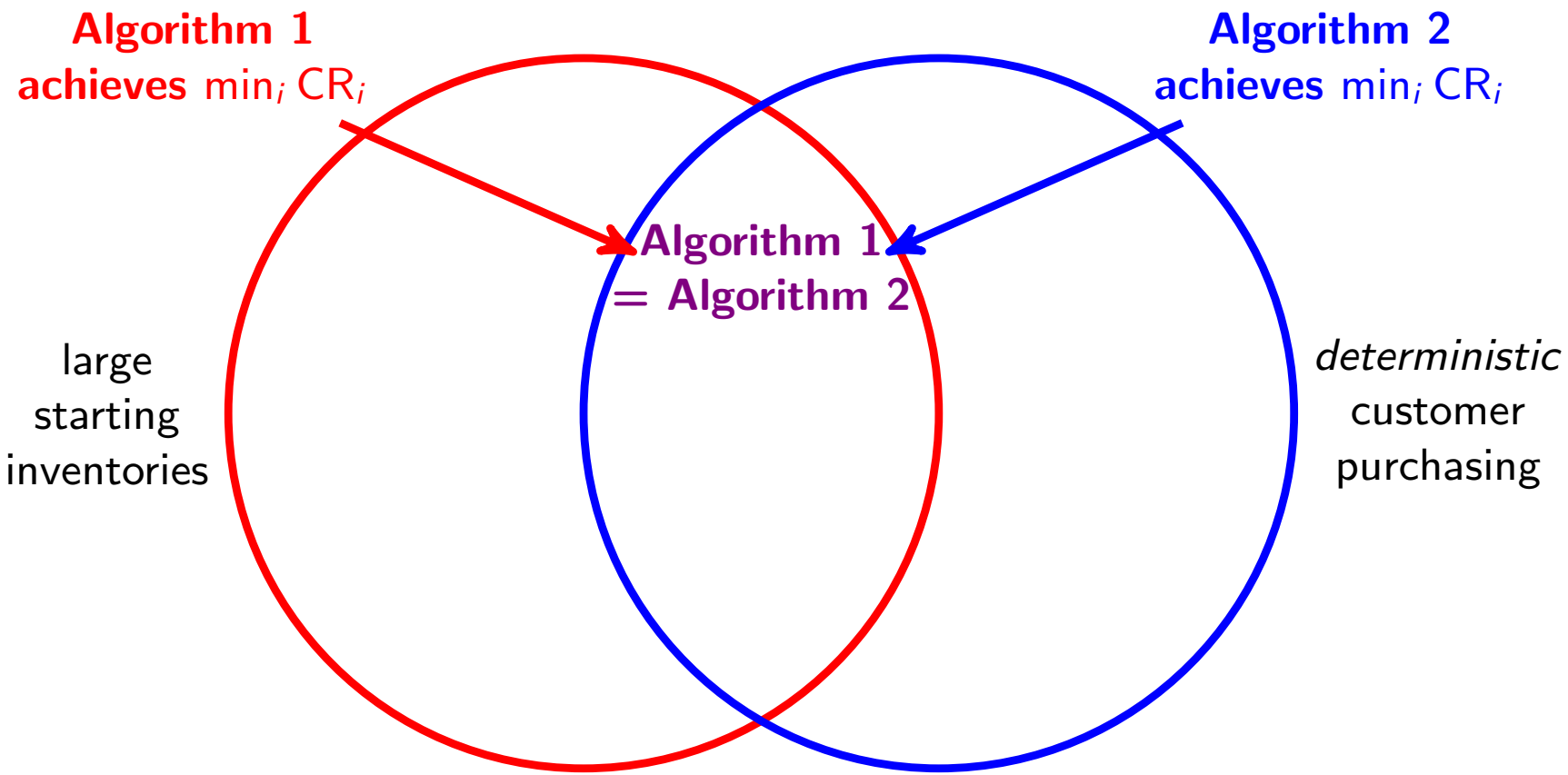
The Competitive Ratio



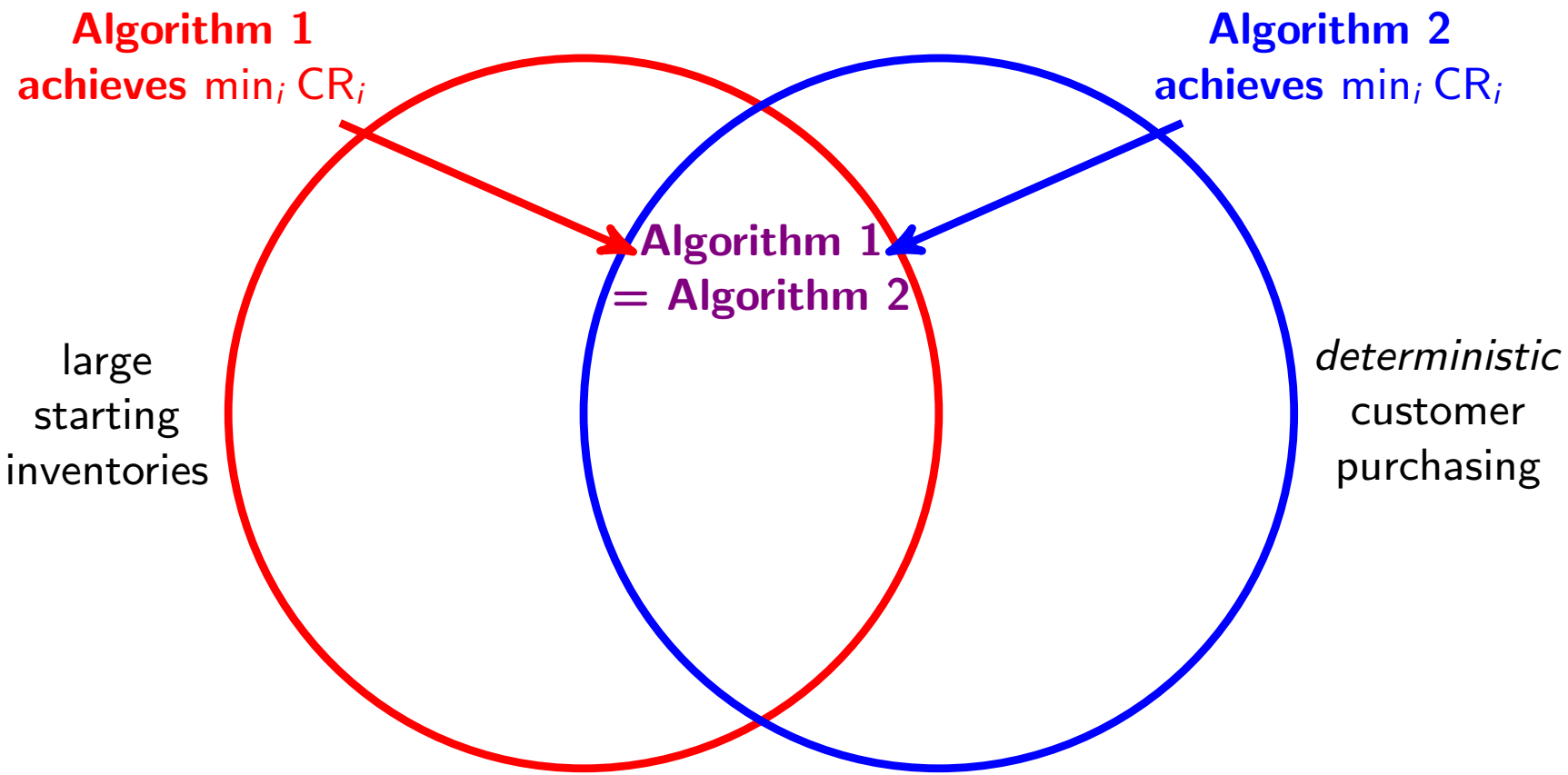
The Competitive Ratio



The Competitive Ratio

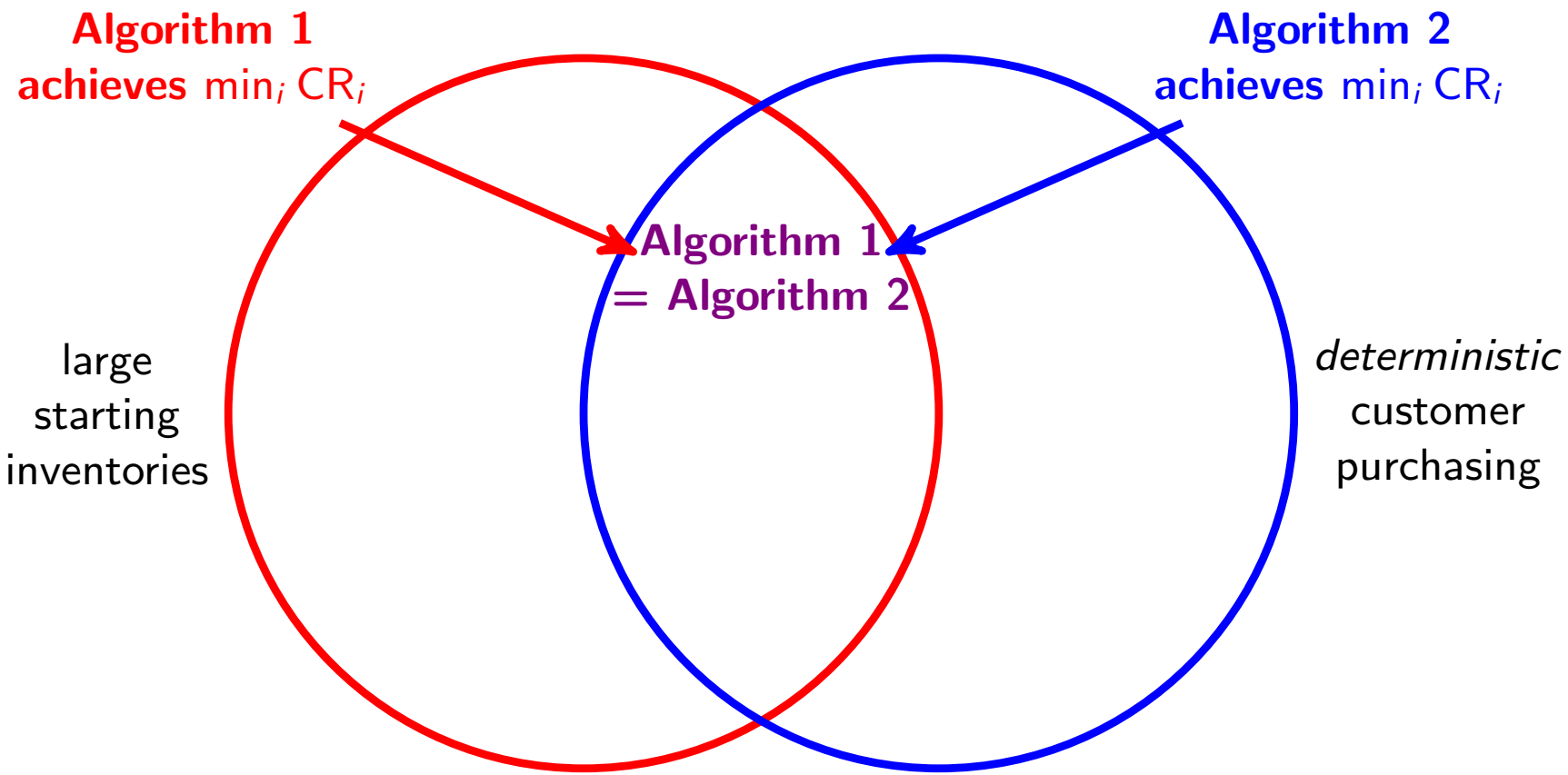


The Competitive Ratio



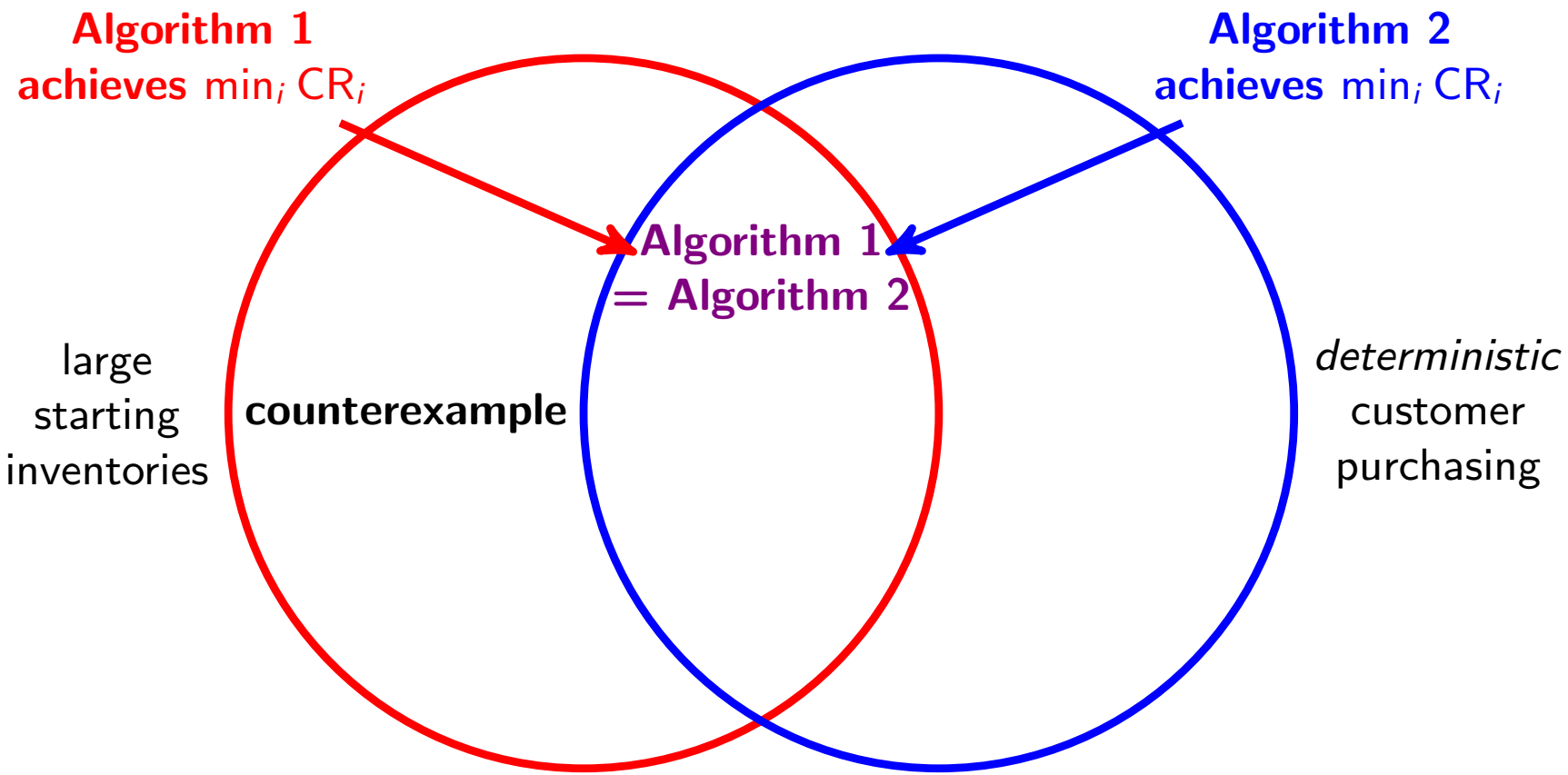
We construct a **counterexample** showing our results are tight, i.e. no online algorithm can achieve a competitive ratio better than $\min_i CR_i$

The **Optimal** Competitive Ratio



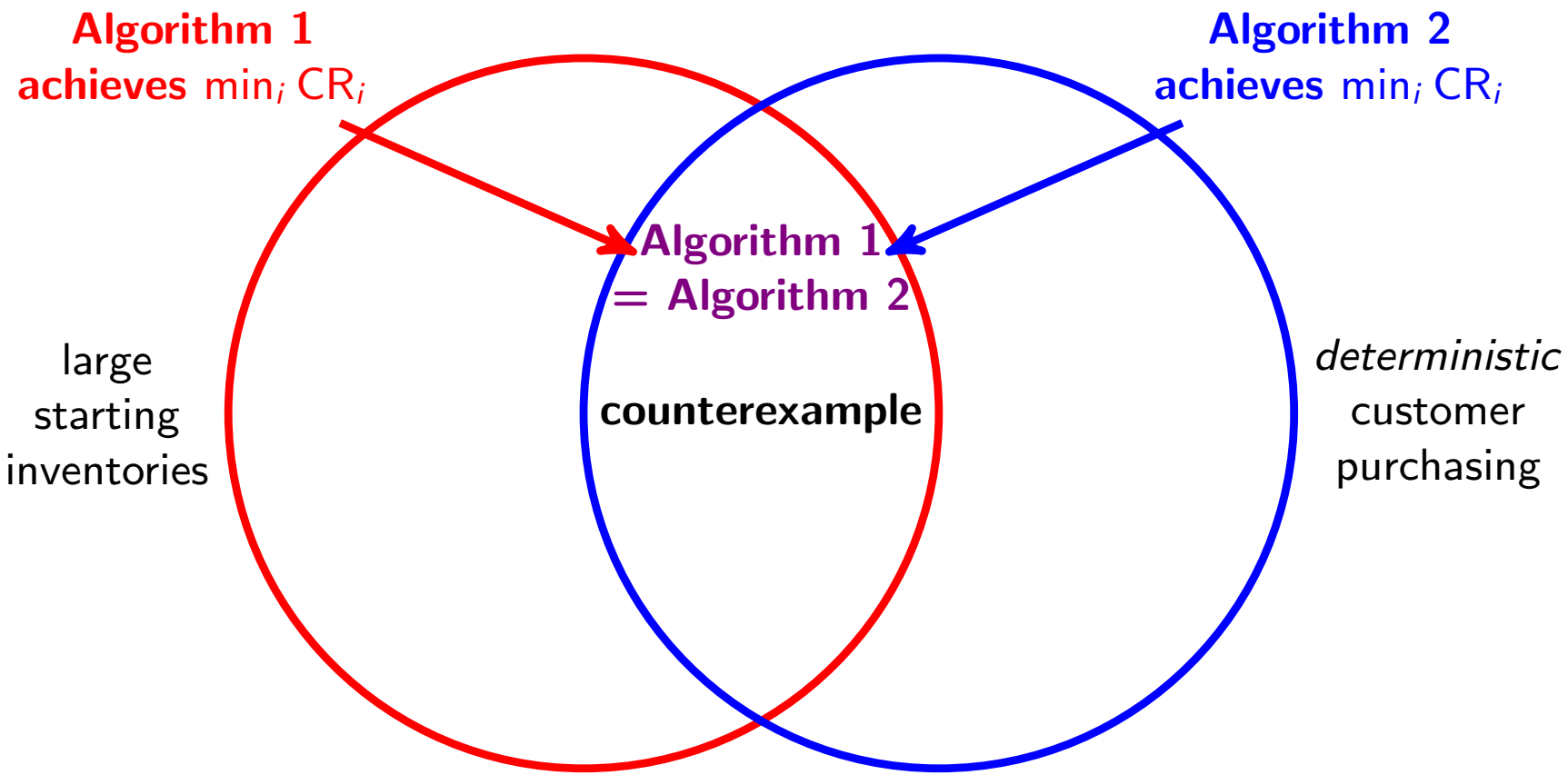
We construct a **counterexample** showing our results are tight, i.e. no online algorithm can achieve a competitive ratio better than $\min_i CR_i$

The **Optimal** Competitive Ratio



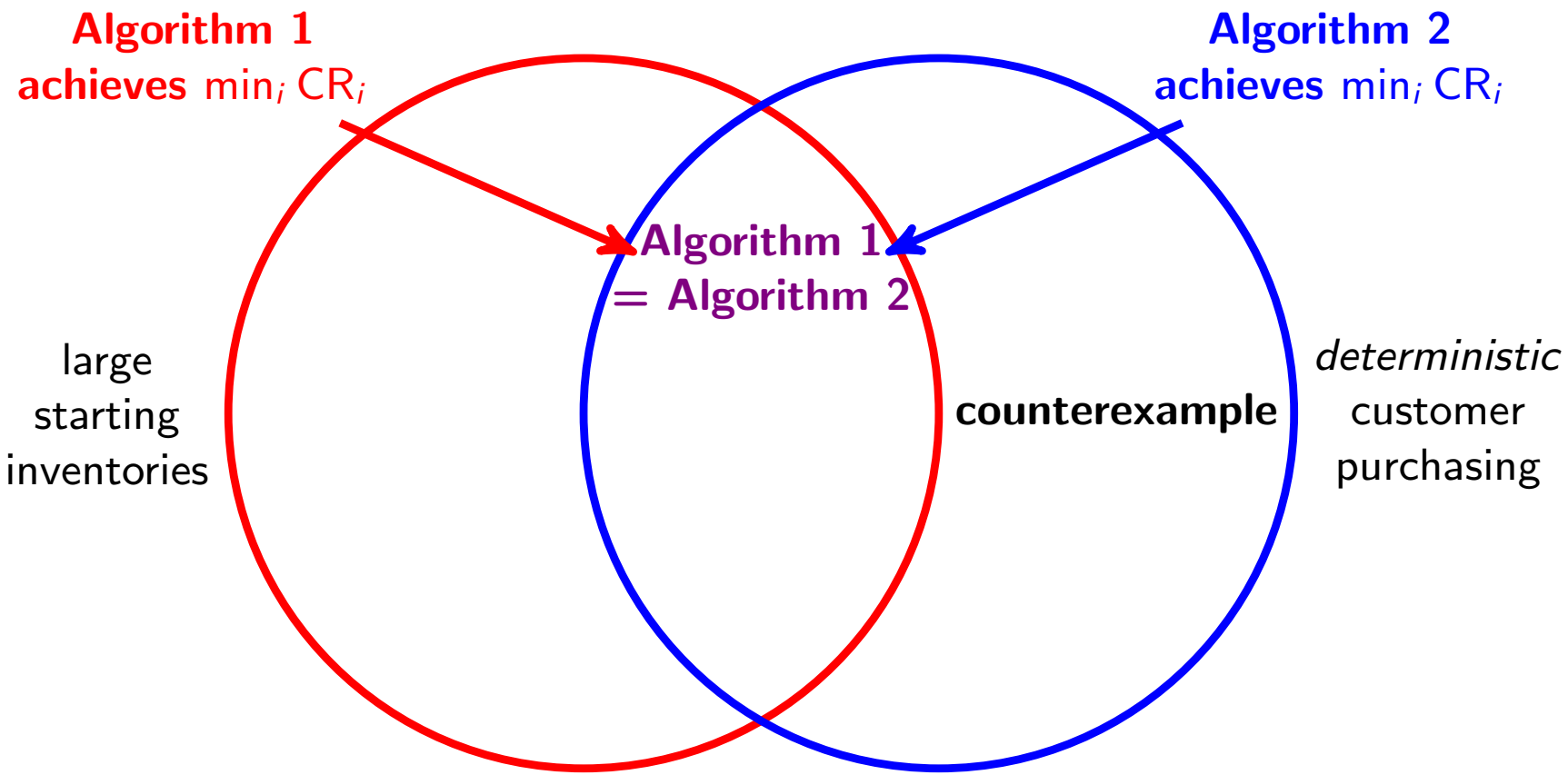
We construct a **counterexample** showing our results are tight, i.e. no online algorithm can achieve a competitive ratio better than $\min_i CR_i$

The **Optimal** Competitive Ratio



We construct a **counterexample** showing our results are tight, i.e. no online algorithm can achieve a competitive ratio better than $\min_i CR_i$

The **Optimal** Competitive Ratio



We construct a **counterexample** showing our results are tight, i.e. no online algorithm can achieve a competitive ratio better than $\min_i CR_i$

Illustration of Competitive Ratios with Two Prices per Item

General Stochastic Setting

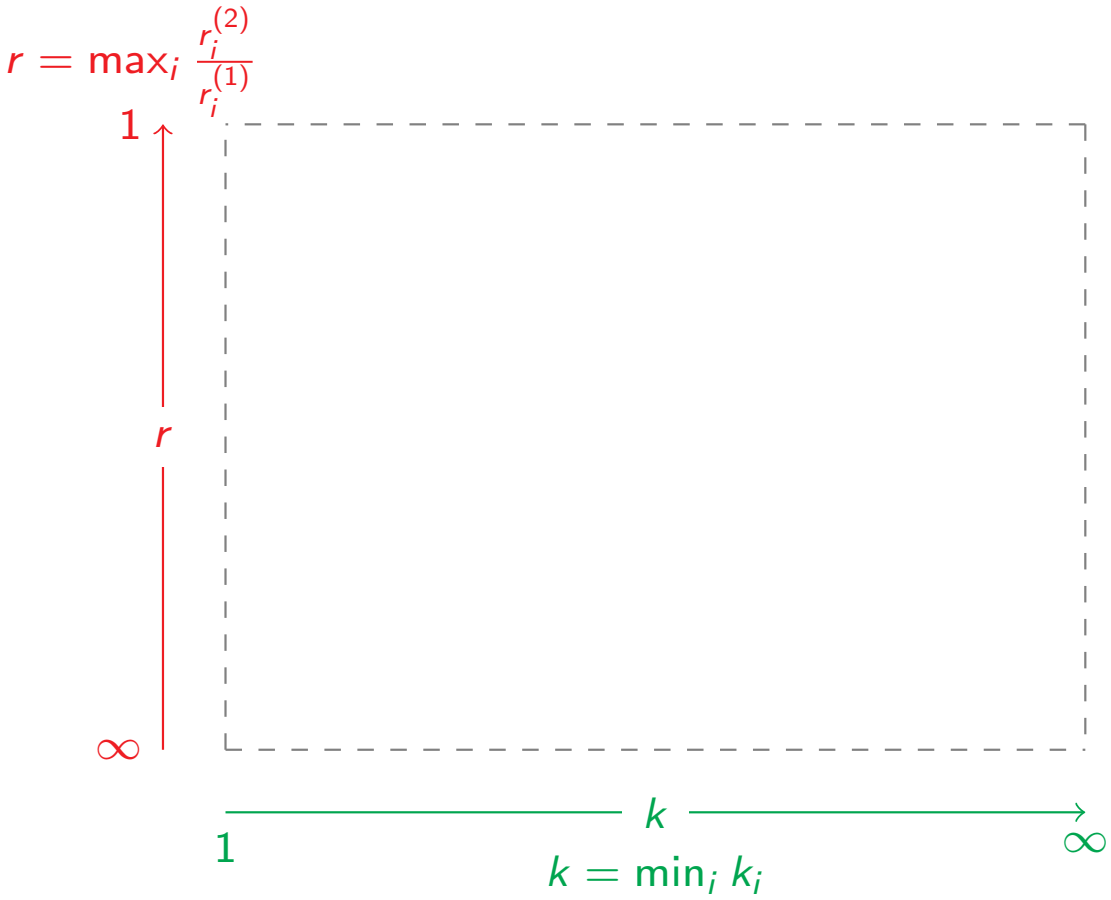


Illustration of Competitive Ratios with Two Prices per Item

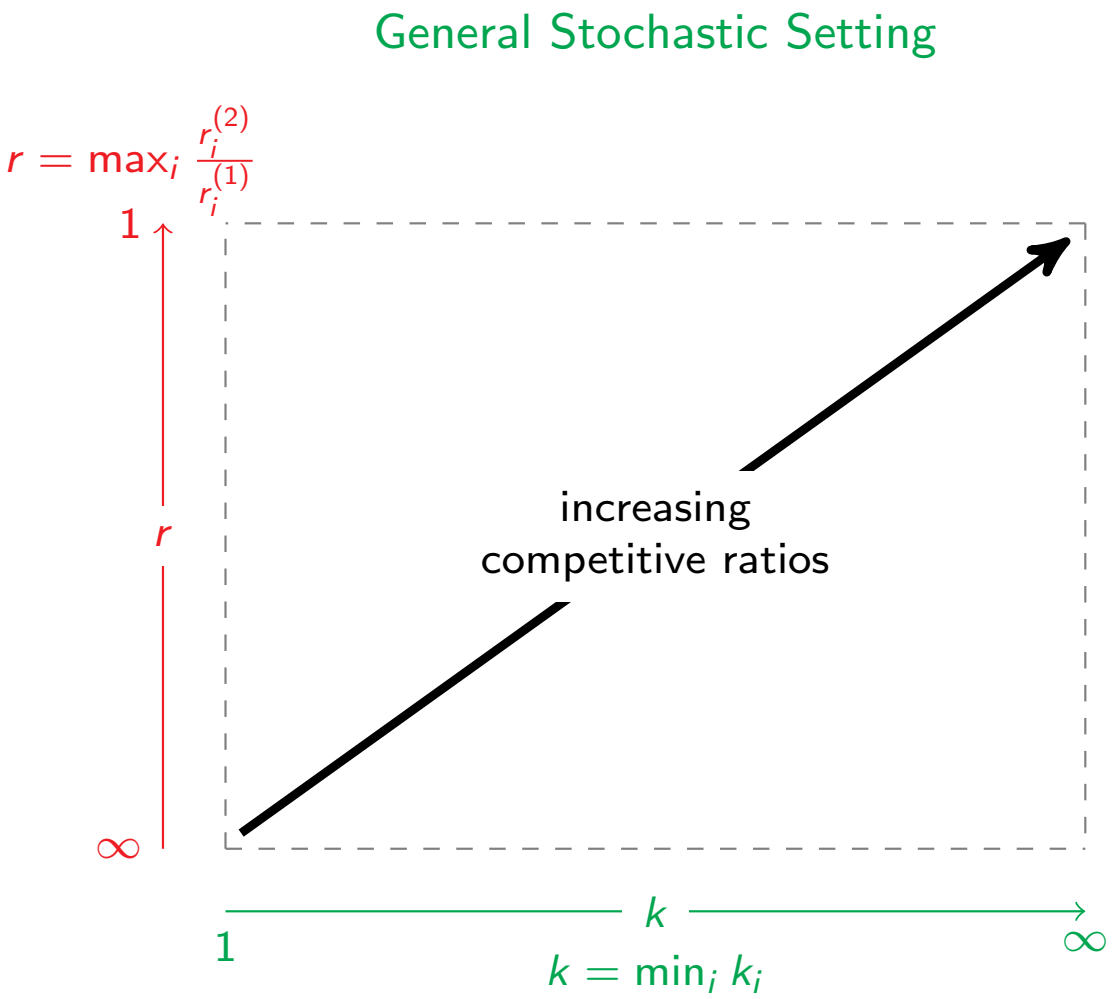
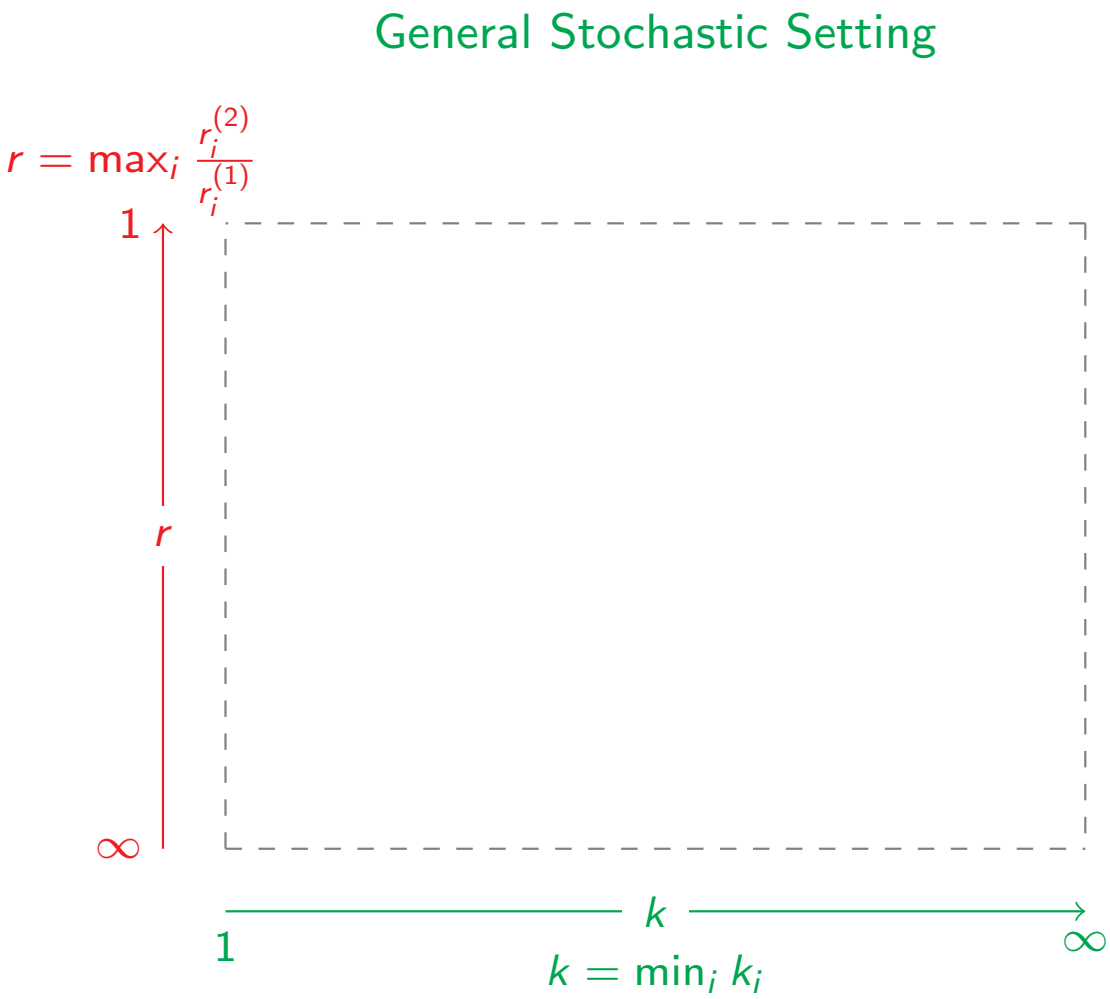


Illustration of Competitive Ratios with Two Prices per Item



tight results

non-tight results

Illustration of Competitive Ratios with Two Prices per Item

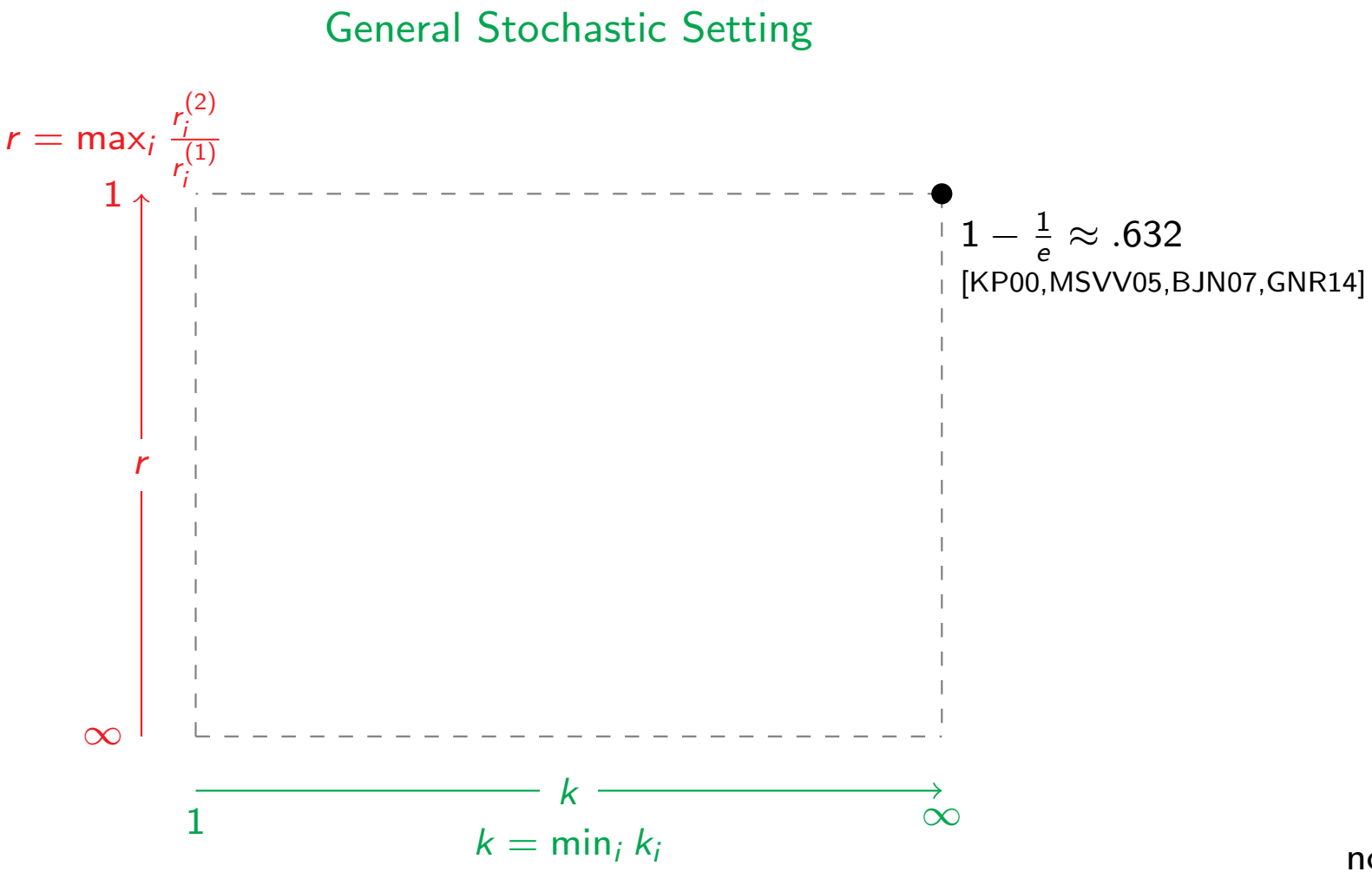


Illustration of Competitive Ratios with Two Prices per Item

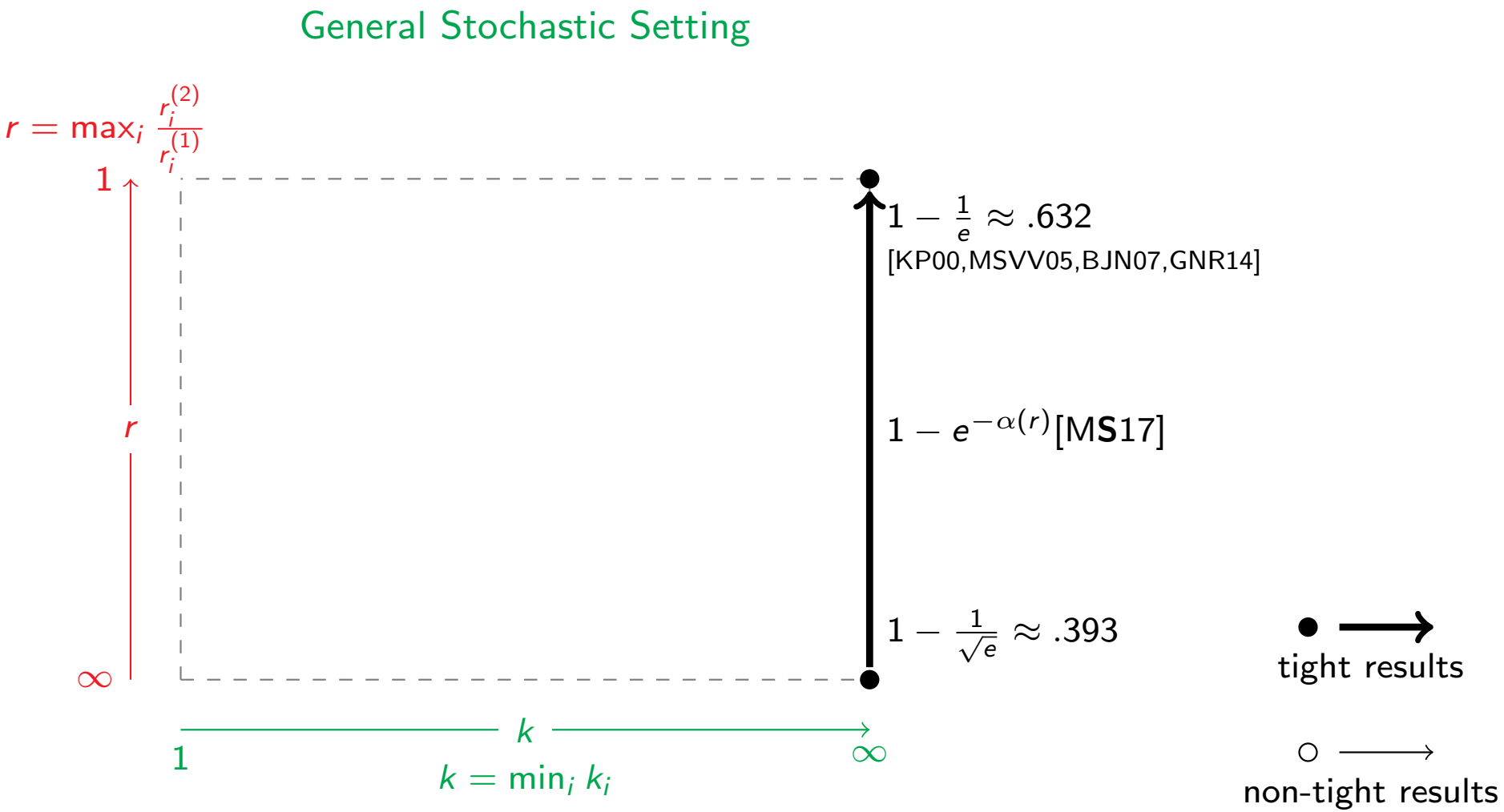


Illustration of Competitive Ratios with Two Prices per Item

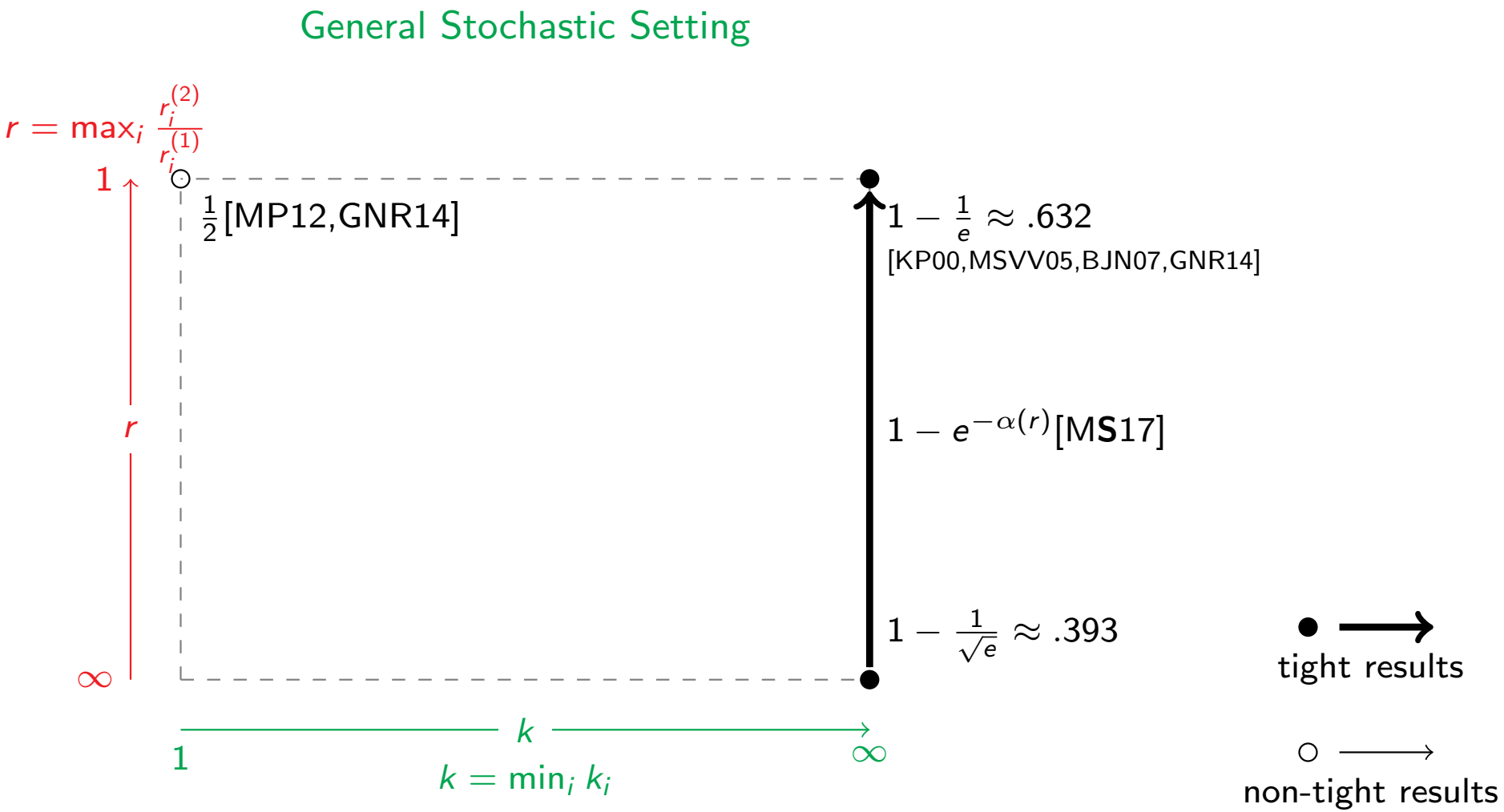


Illustration of Competitive Ratios with Two Prices per Item

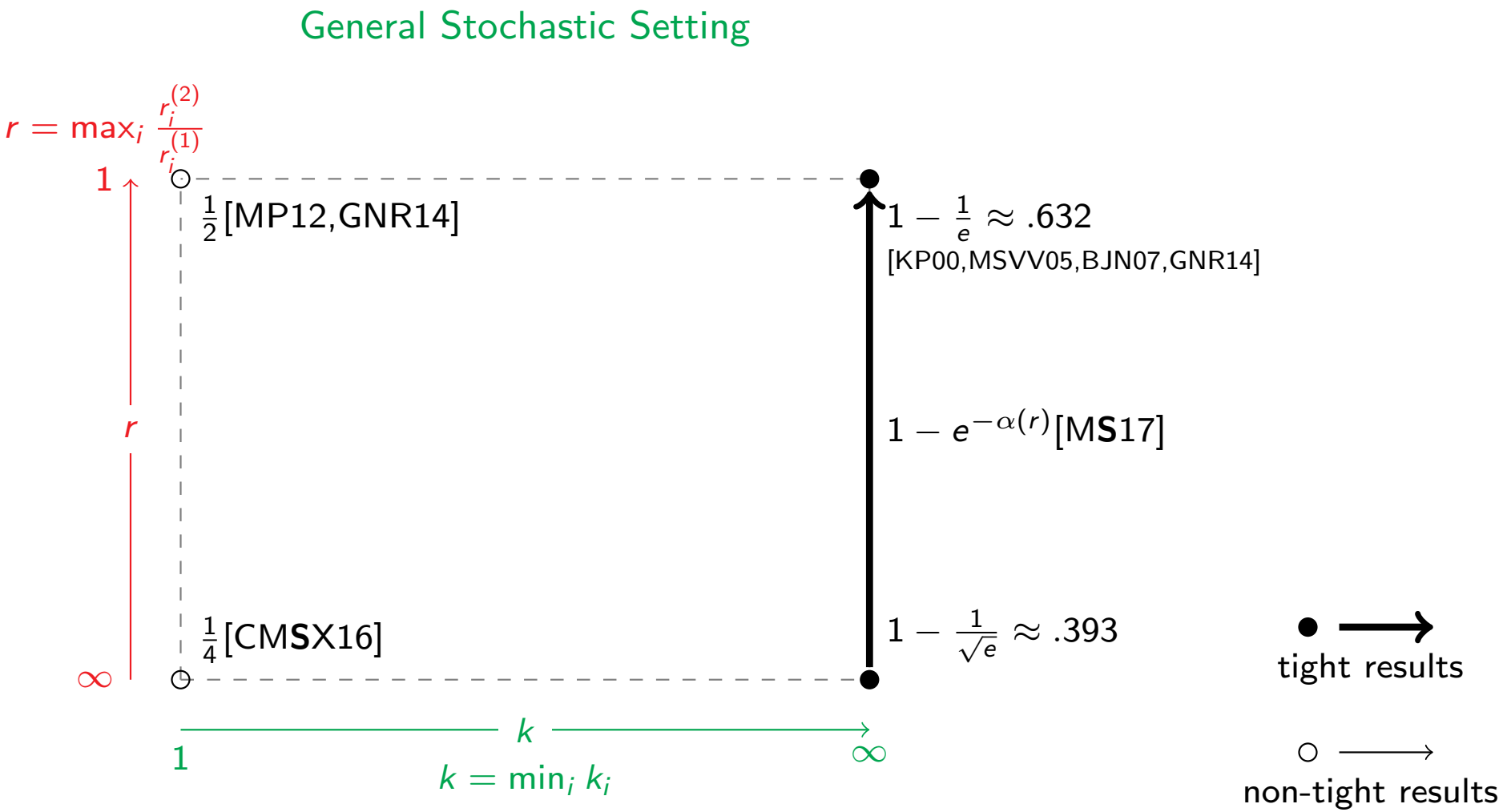


Illustration of Competitive Ratios with Two Prices per Item

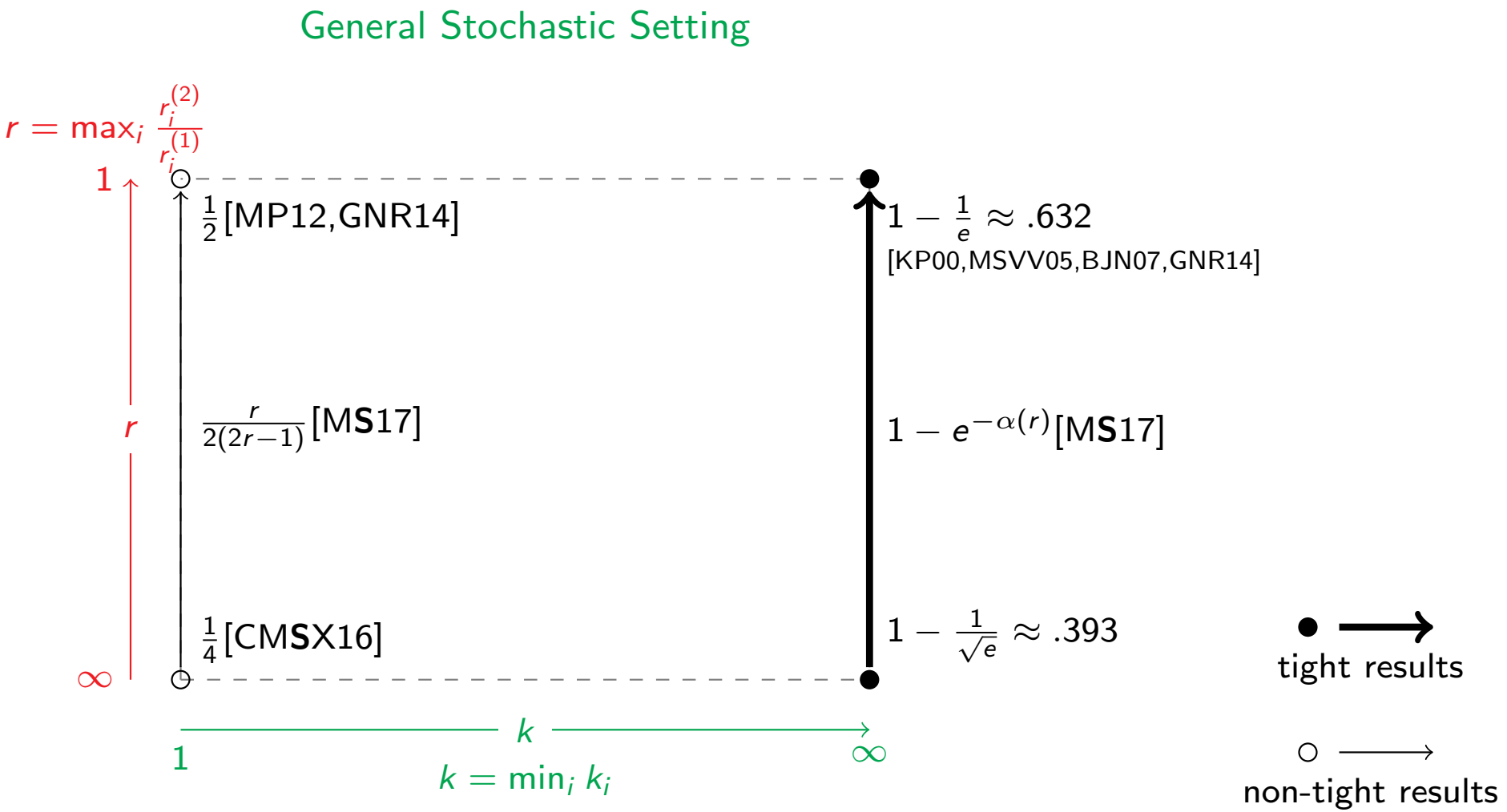


Illustration of Competitive Ratios with Two Prices per Item

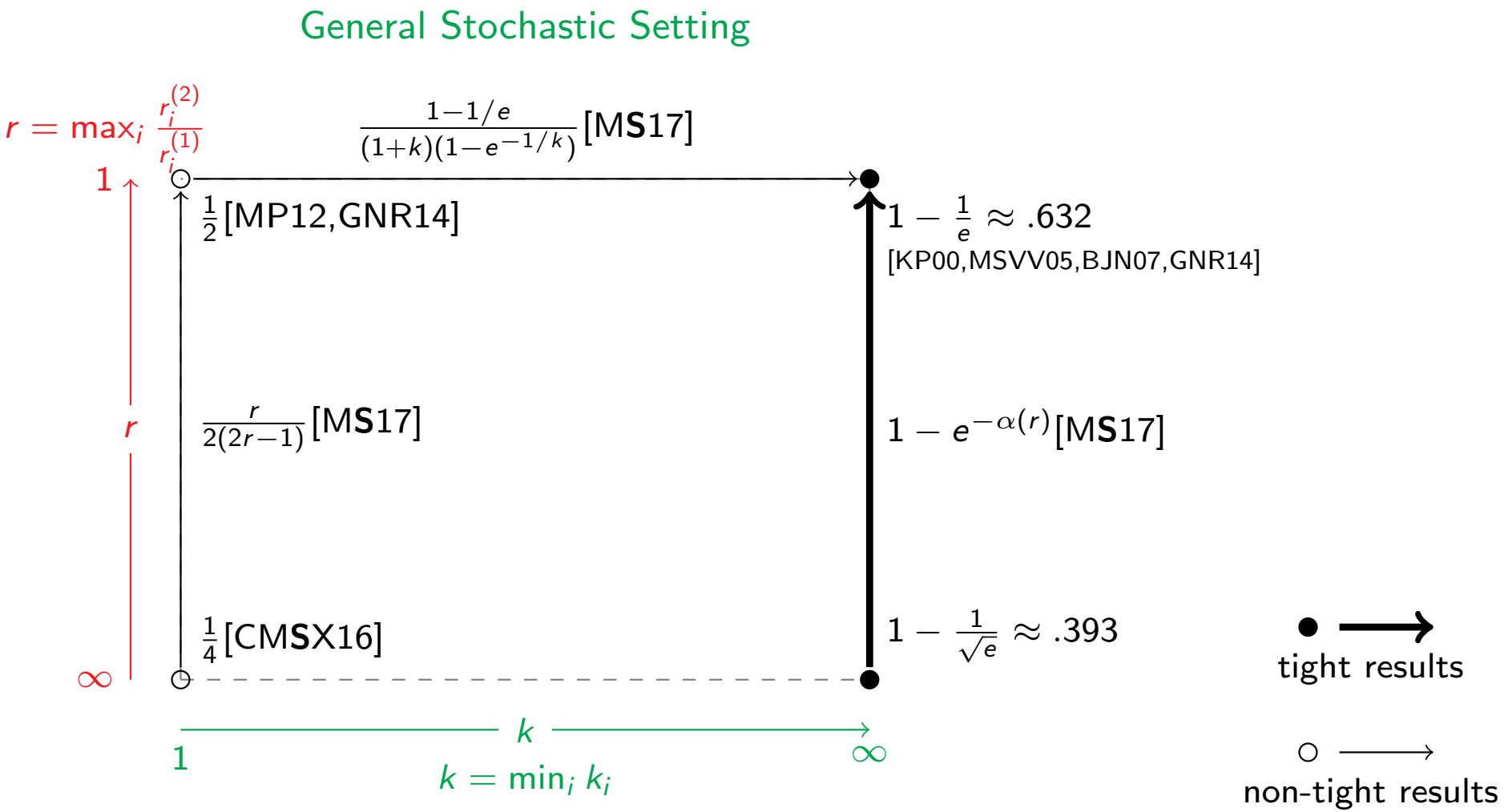


Illustration of Competitive Ratios with Two Prices per Item

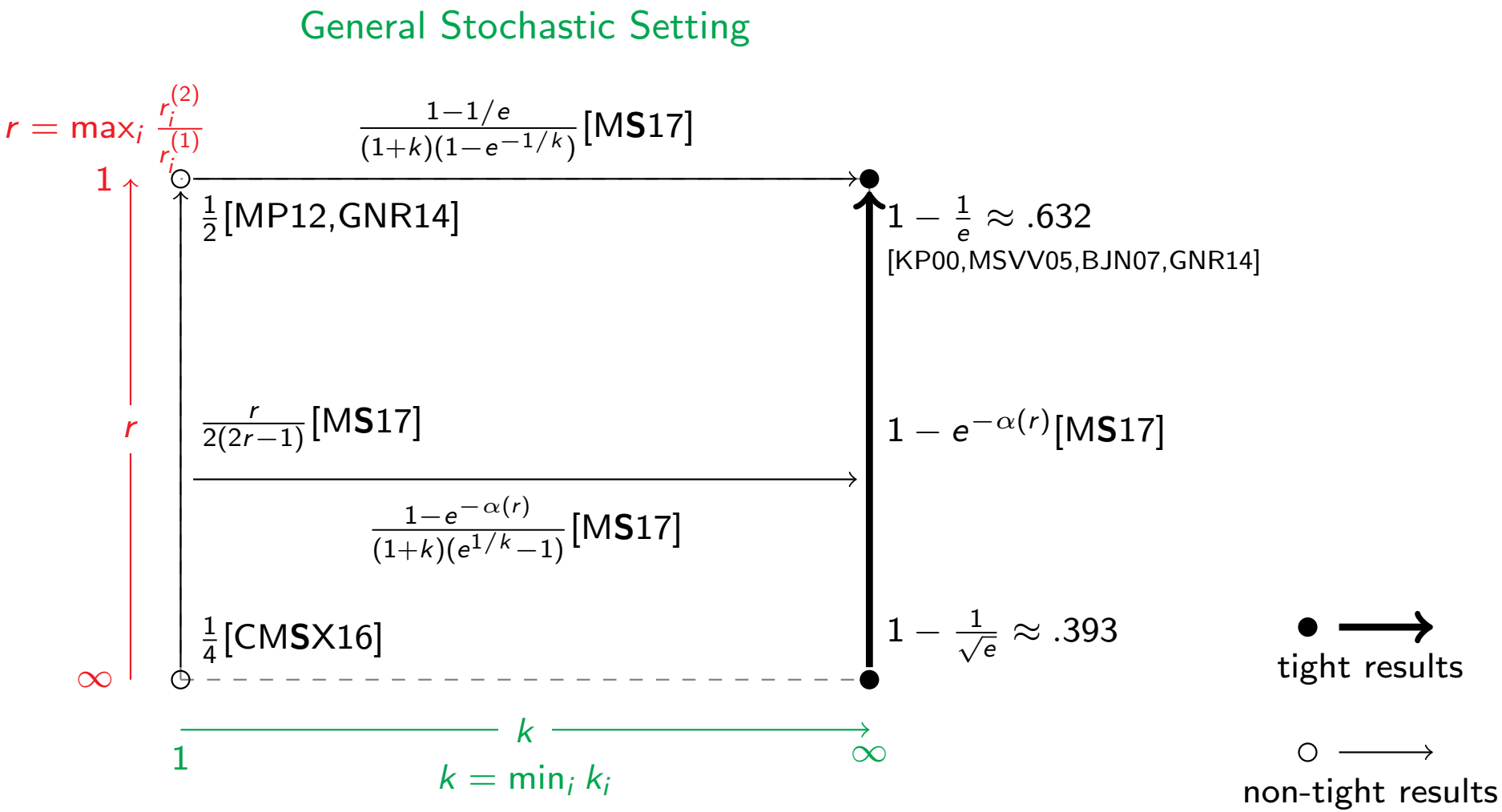
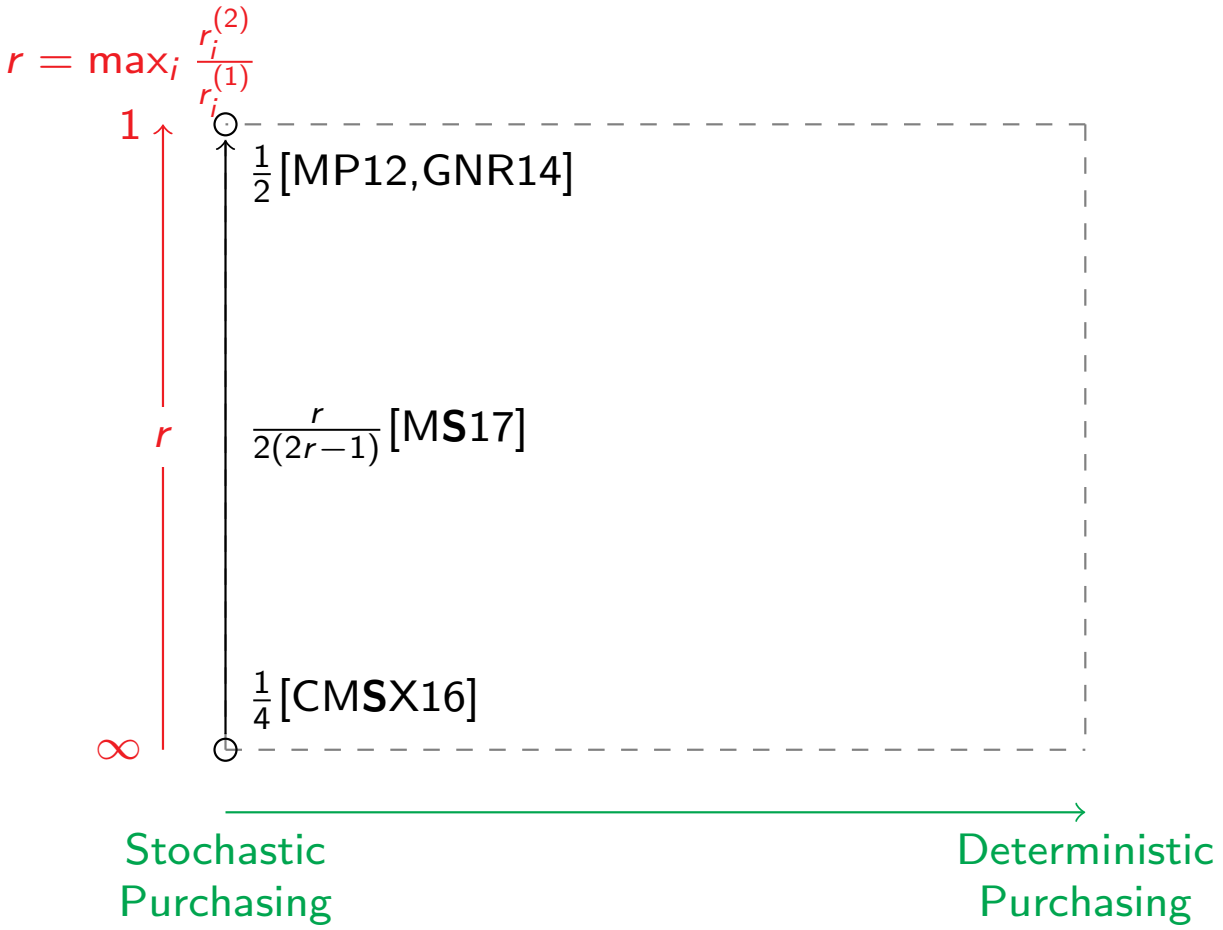


Illustration of Competitive Ratios with Two Prices per Item



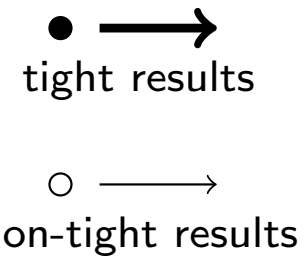


Illustration of Competitive Ratios with Two Prices per Item

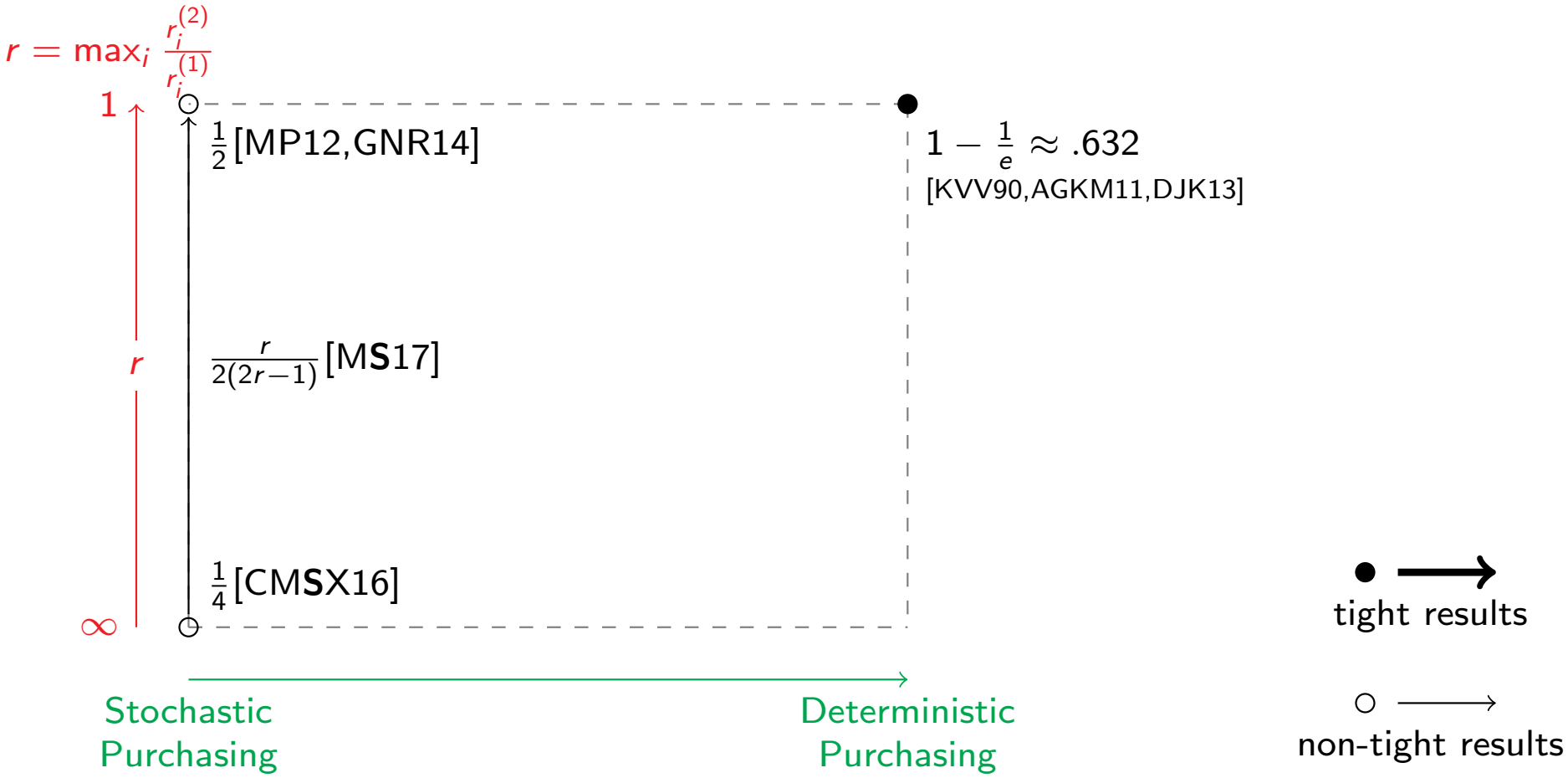


Illustration of Competitive Ratios with Two Prices per Item

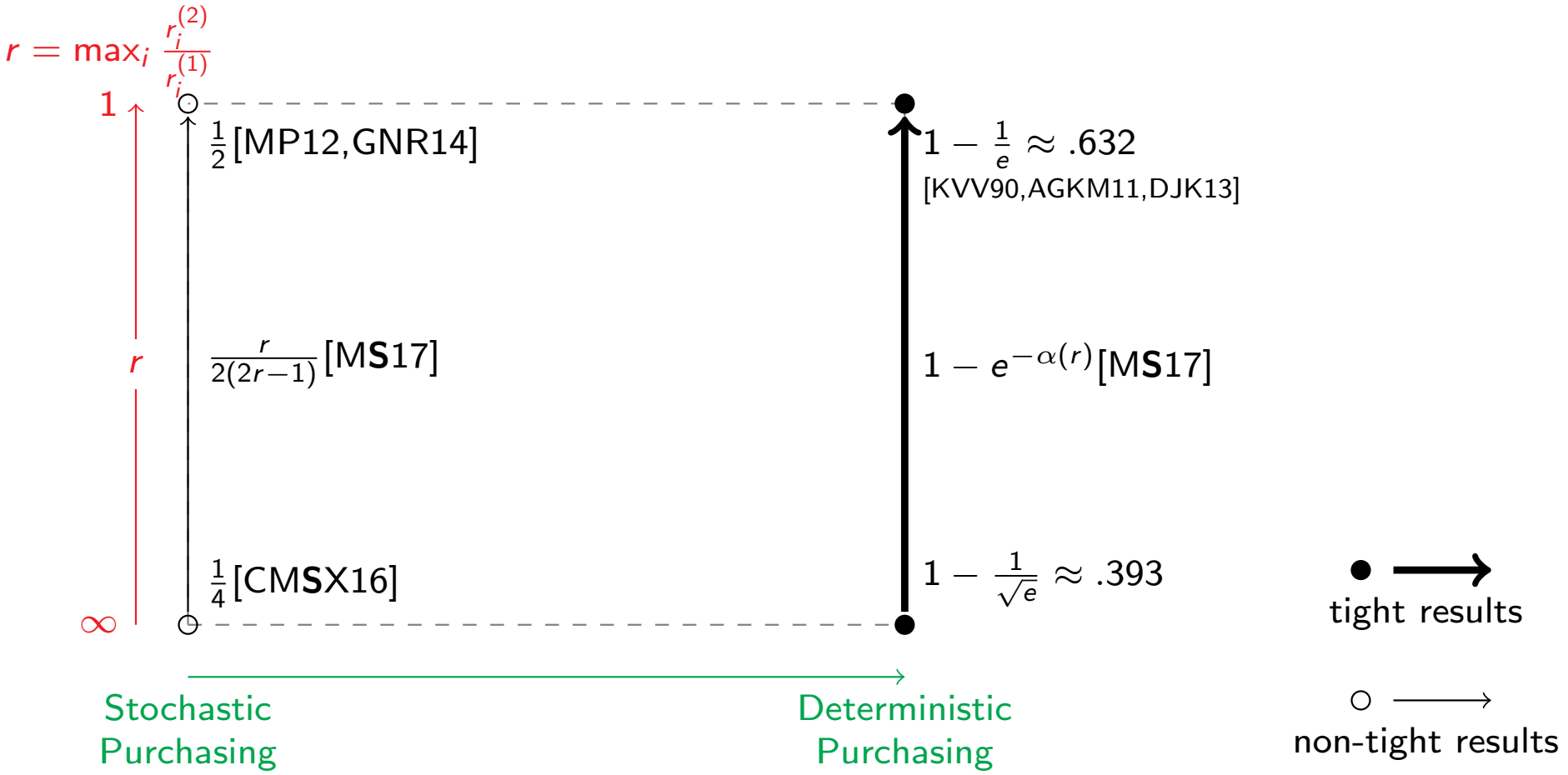


Illustration of Competitive Ratios with Two Prices per Item

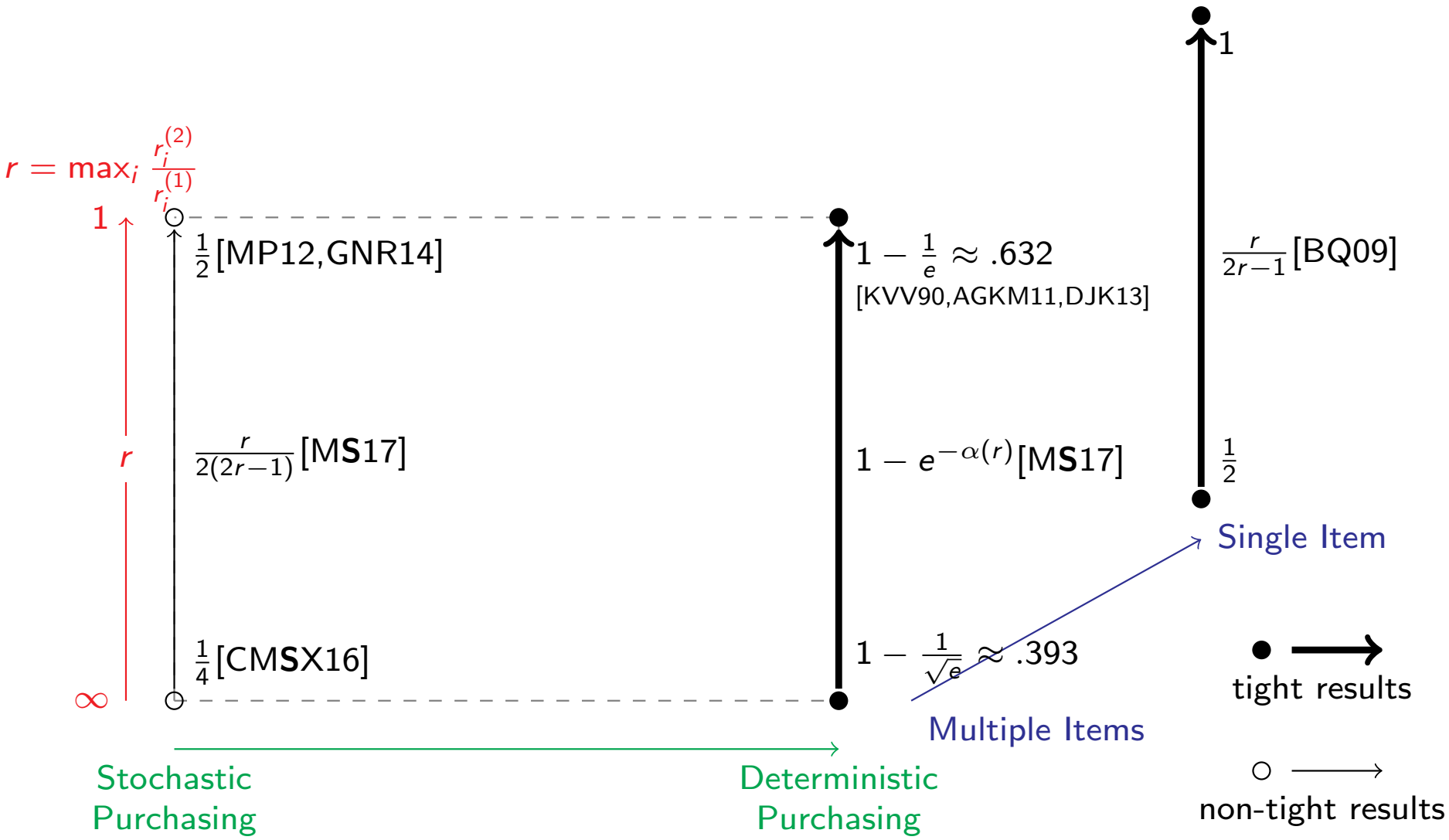


Illustration of Competitive Ratios with Two Prices per Item

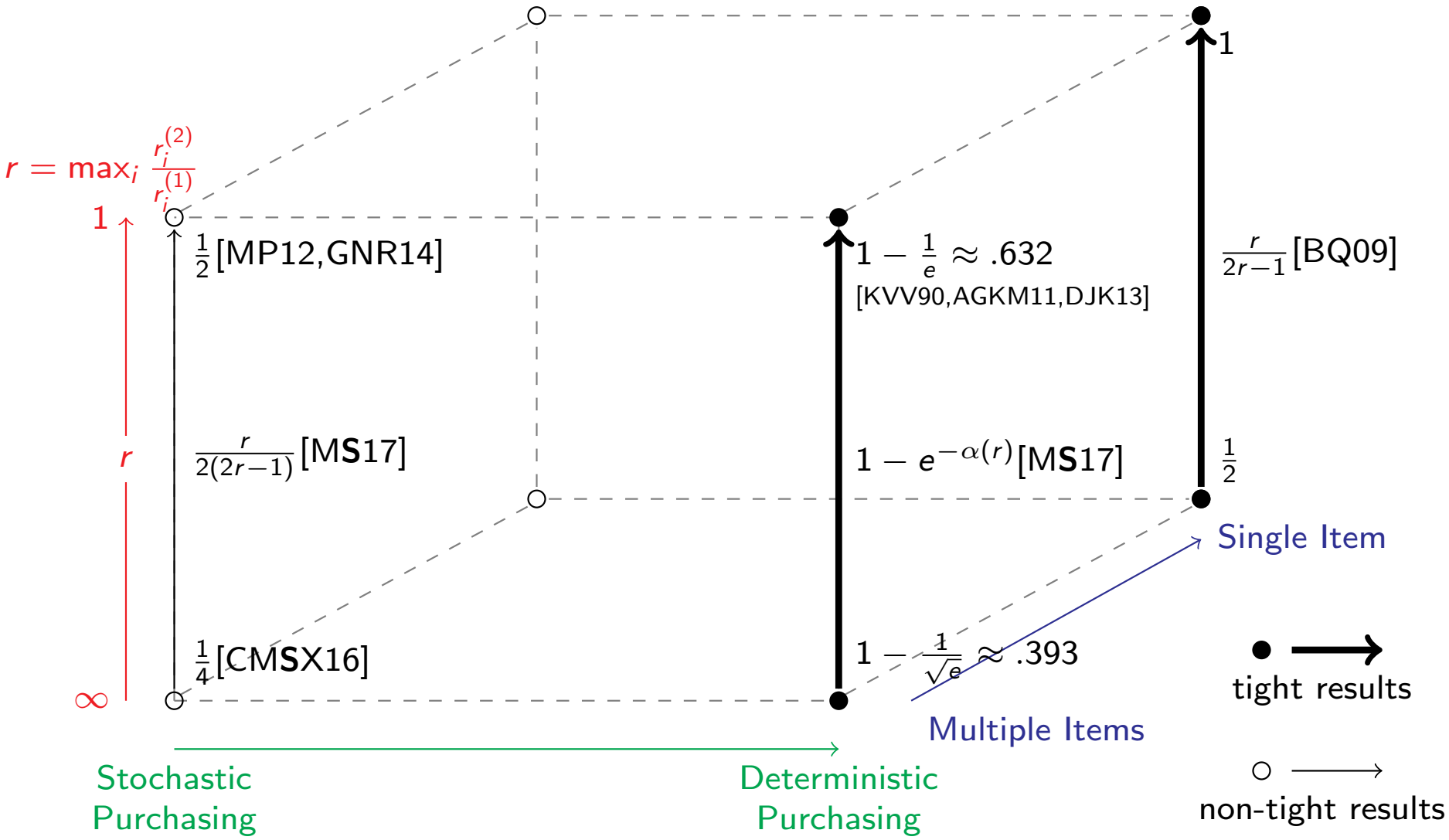


Illustration of Competitive Ratios with Two Prices per Item

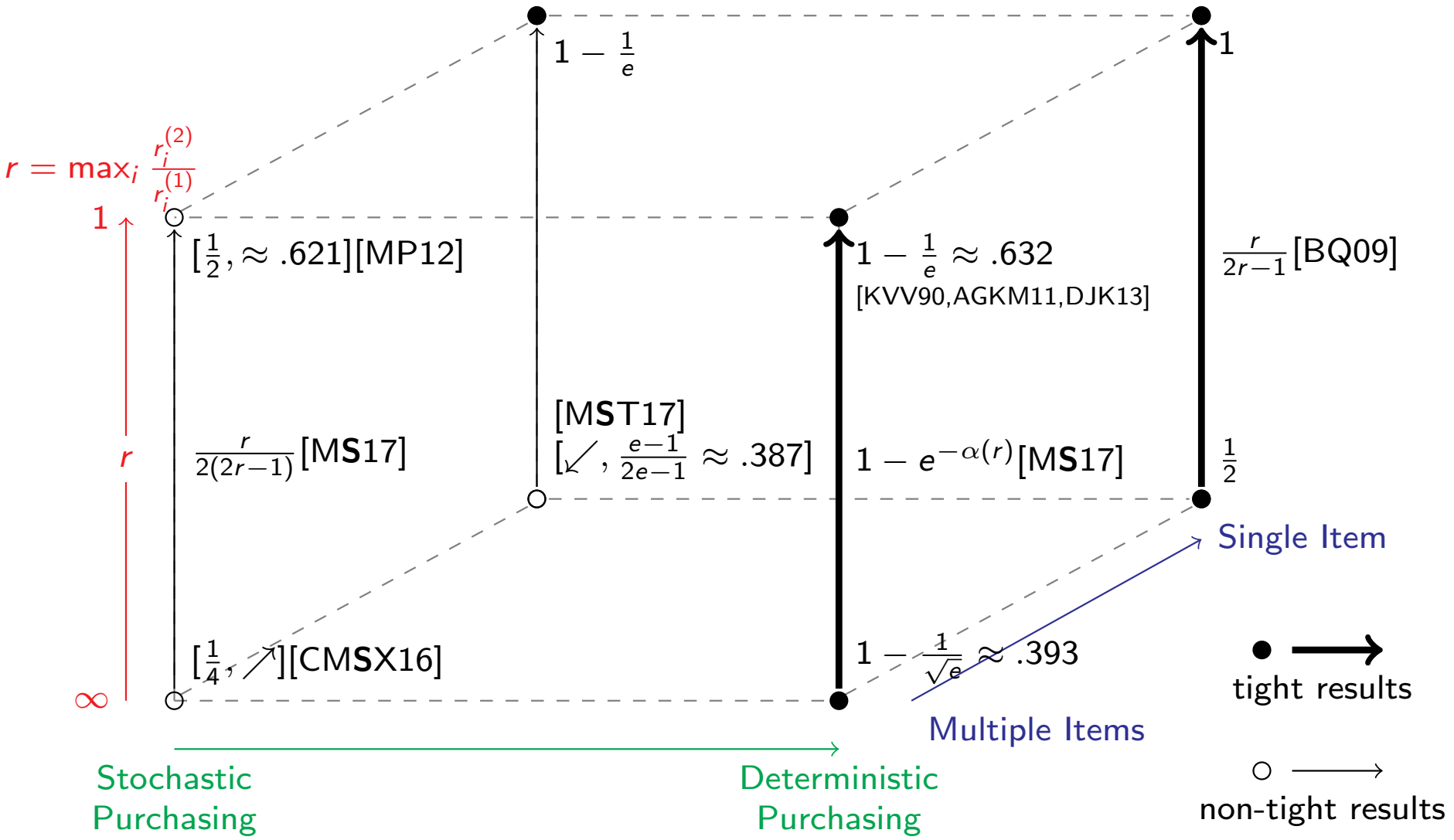
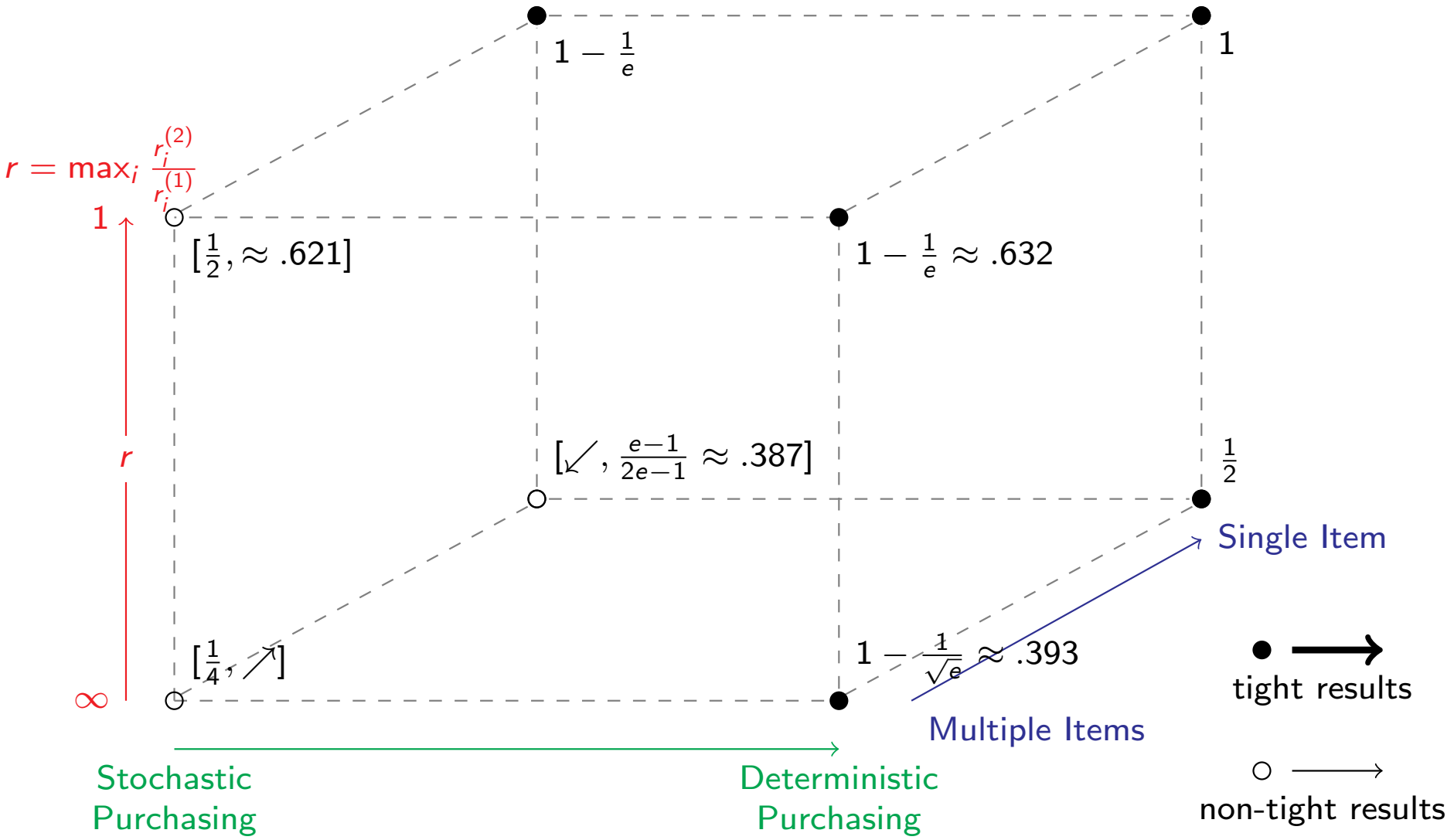


Illustration of Competitive Ratios with Two Prices per Item



Simulations on Hotel Data Set

Simulations on Hotel Data Set

- publicly-accessible hotel data set from MSOM journal (Bodea/Ferguson/Garrow '09)

Simulations on Hotel Data Set

- publicly-accessible hotel data set from MSOM journal (Bodea/Ferguson/Garrow '09)
- multiple items (room categories) with multiple prices (rates)

Room Category	Discounted Rate	Rack Rate	Number of Rooms
King	\$307	\$361	440
Queen	\$304	\$361	130
Suite	\$384	\$496	110
Two Double	\$306	\$342	170

Simulations on Hotel Data Set

- publicly-accessible hotel data set from MSOM journal (Bodea/Ferguson/Garrow '09)
- multiple items (room categories) with multiple prices (rates)

Room Category	Discounted Rate	Rack Rate	Number of Rooms
King	\$307	\$361	440
Queen	\$304	\$361	130
Suite	\$384	\$496	110
Two Double	\$306	\$342	170

- data is given for check-in dates in March–April 2007

Simulations on Hotel Data Set

- publicly-accessible hotel data set from MSOM journal (Bodea/Ferguson/Garrow '09)
- multiple items (room categories) with multiple prices (rates)

Room Category	Discounted Rate	Rack Rate	Number of Rooms
King	\$307	\$361	440
Queen	\$304	\$361	130
Suite	\$384	\$496	110
Two Double	\$306	\$342	170

- data is given for check-in dates in March–April 2007
- we assume that each check-in date is a **separate** problem instance

Simulations on Hotel Data Set

- publicly-accessible hotel data set from MSOM journal (Bodea/Ferguson/Garrow '09)
- multiple items (room categories) with multiple prices (rates)

Room Category	Discounted Rate	Rack Rate	Number of Rooms
King	\$307	\$361	440
Queen	\$304	\$361	130
Suite	\$384	\$496	110
Two Double	\$306	\$342	170

- data is given for check-in dates in March–April 2007
- we assume that each check-in date is a **separate** problem instance
- the arrival sequence for each check-in date is given by the transaction history for that date (over the year before check-in)

Simulations on Hotel Data Set

- publicly-accessible hotel data set from MSOM journal (Bodea/Ferguson/Garrow '09)
- multiple items (room categories) with multiple prices (rates)

Room Category	Discounted Rate	Rack Rate	Number of Rooms
King	\$307	\$361	440
Queen	\$304	\$361	130
Suite	\$384	\$496	110
Two Double	\$306	\$342	170

- data is given for check-in dates in March–April 2007
- we assume that each check-in date is a **separate** problem instance
- the arrival sequence for each check-in date is given by the transaction history for that date (over the year before check-in)
- on average, 1340 total arrivals per check-in date (scaled by 10)

Algorithms Compared

Algorithms Compared

Algorithm decides which room/fare combinations should be offered to different customers from different channels (website, CRO/CRS, GDS):

Algorithms Compared

Algorithm decides which room/fare combinations should be offered to different customers from different channels (website, CRO/CRS, GDS):

- **Forecast-dependent Algorithms**
  - update these assortments based on remaining inventory,
  - relative to the forecasted distribution of customers still to come (this depends on the number of days until check-in);

Algorithms Compared

Algorithm decides which room/fare combinations should be offered to different customers from different channels (website, CRO/CRS, GDS):

- **Forecast-dependent Algorithms**
  - update these assortments based on remaining inventory,
  - relative to the forecasted distribution of customers still to come (this depends on the number of days until check-in);
- **Forecast-independent Algorithms**
  - optimize these assortments for the worst case,
  - dynamically hedging against the worst case as inventory is sold;

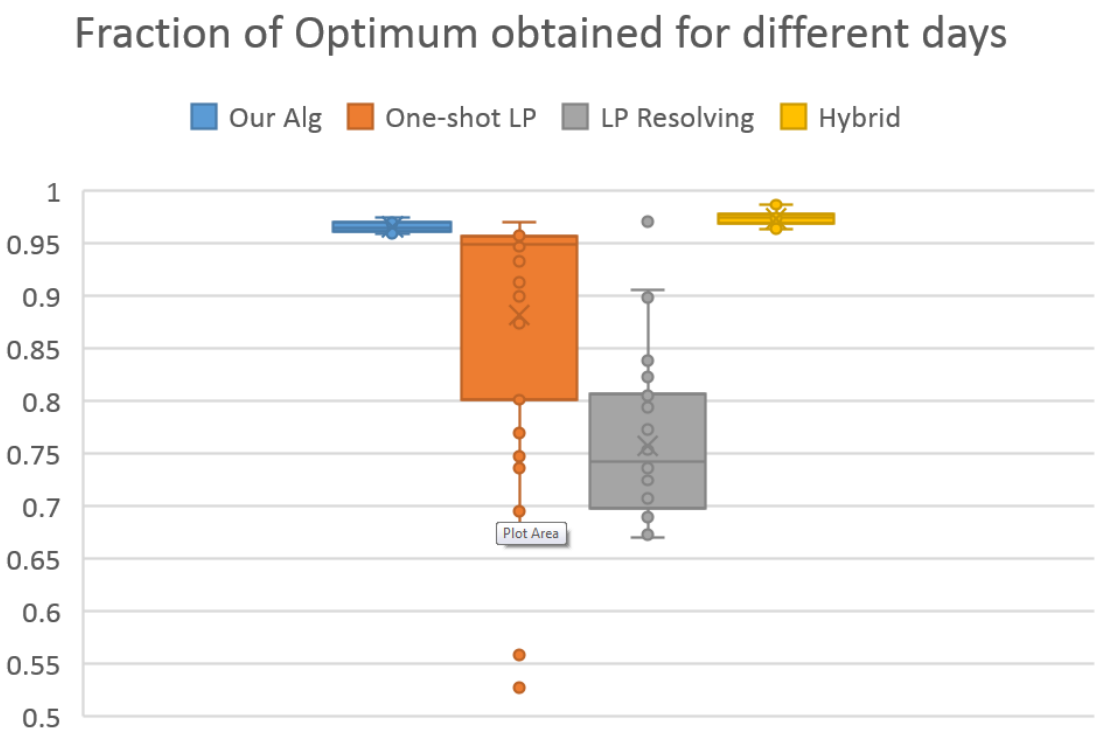
Algorithms Compared

Algorithm decides which room/fare combinations should be offered to different customers from different channels (website, CRO/CRS, GDS):

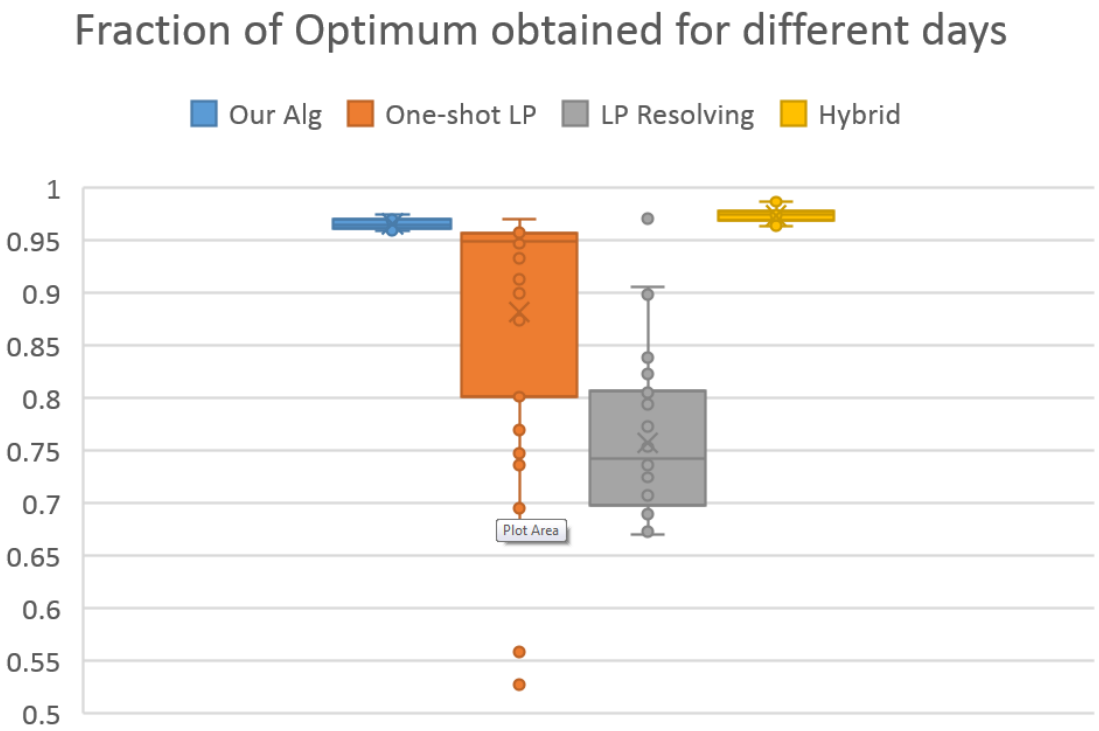
- **Forecast-dependent Algorithms**
  - update these assortments based on remaining inventory,
  - relative to the forecasted distribution of customers still to come (this depends on the number of days until check-in);
- **Forecast-independent Algorithms**
  - optimize these assortments for the worst case,
  - dynamically hedging against the worst case as inventory is sold;
- **Hybrid Algorithms**
  - follow the forecast-dependent algorithm during each time step,
  - however, if the decision prescribed is suboptimal in the worst-case beyond a certain threshold,
  - then it uses the decision of the forecast-independent algorithm instead.

Results

Results

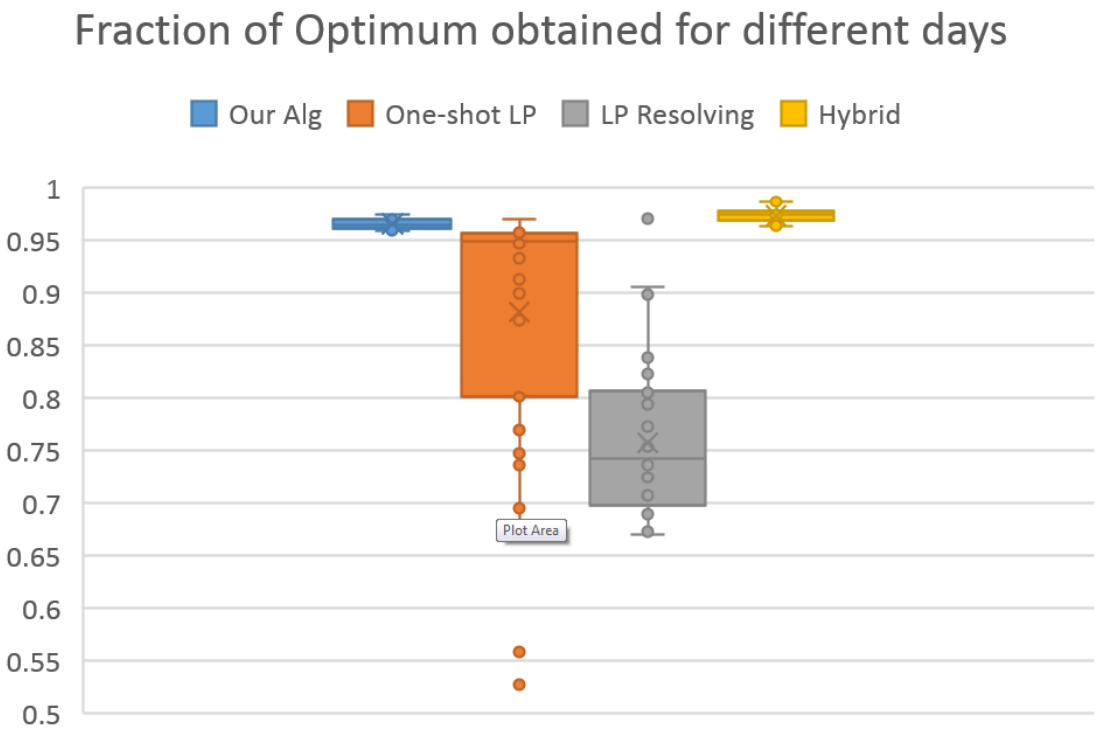

Fraction of Optimum obtained for different days
Our Alg
One-shot LP
LP Resolving
Hybrid
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
Plot Area

Results



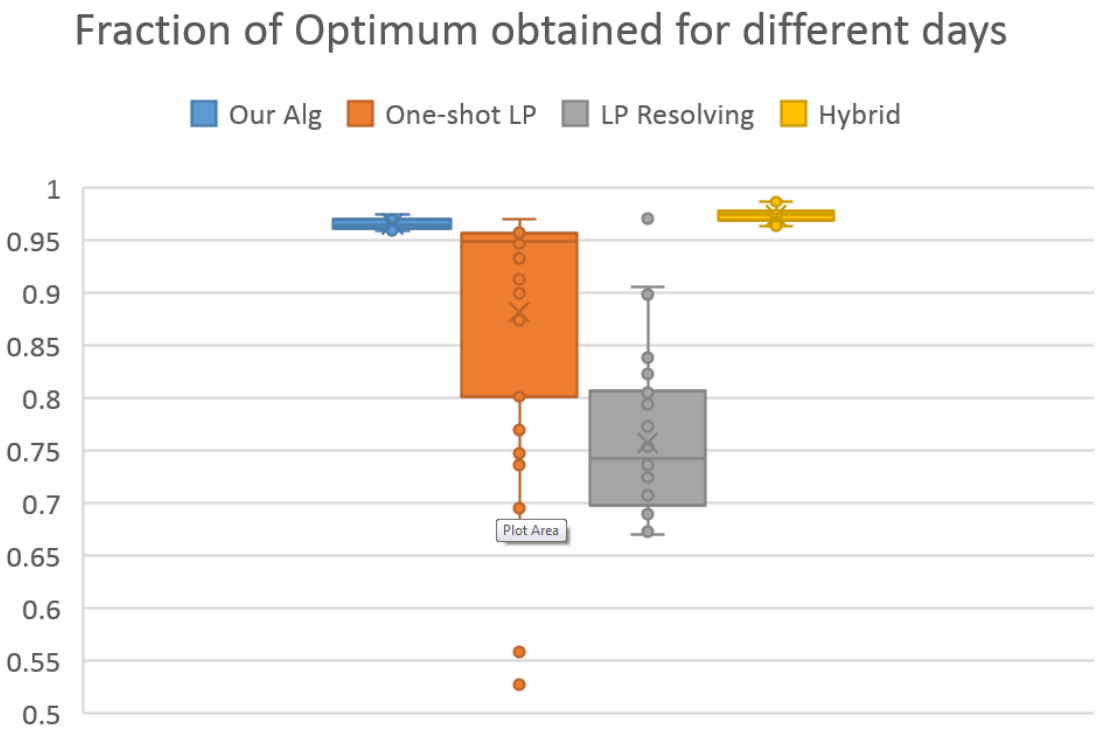
- our algorithm extracts a large fraction of optimum on all instances

Results



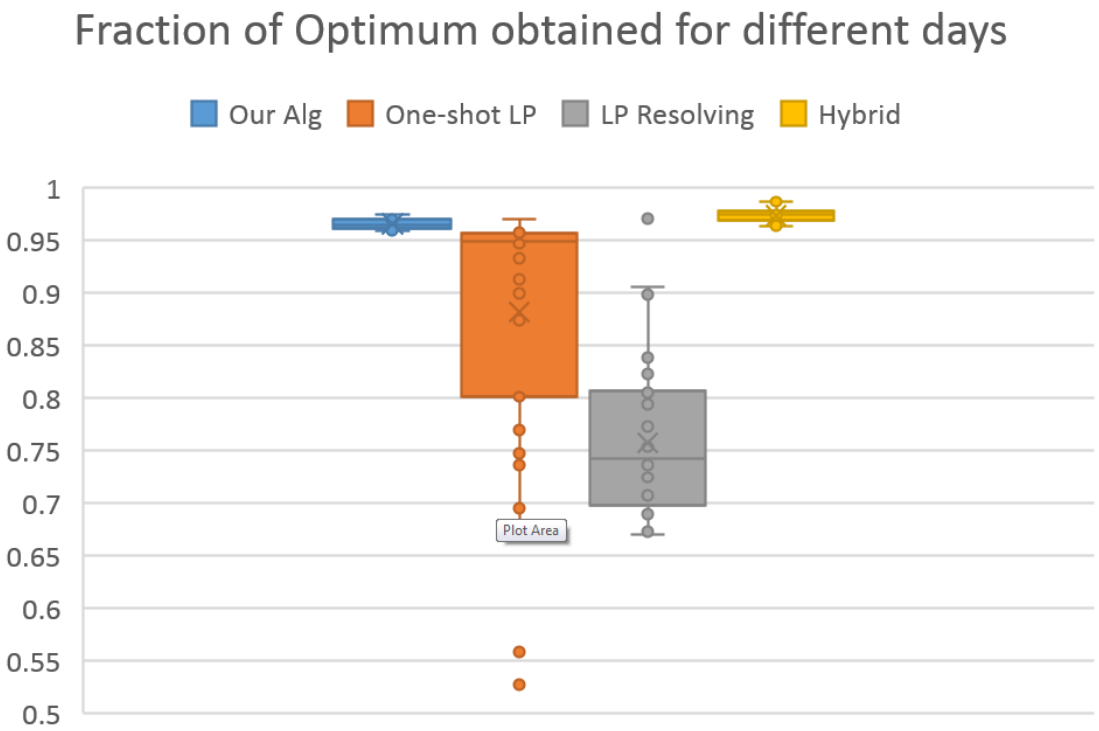
- our algorithm extracts a large fraction of optimum on all instances
- the Hybrid of our algorithm and LP Resolving performs even better

Results



- our algorithm extracts a large fraction of optimum on all instances
- the Hybrid of our algorithm and LP Resolving performs even better
- the forecast-dependent algorithms exhibit a lot more variance

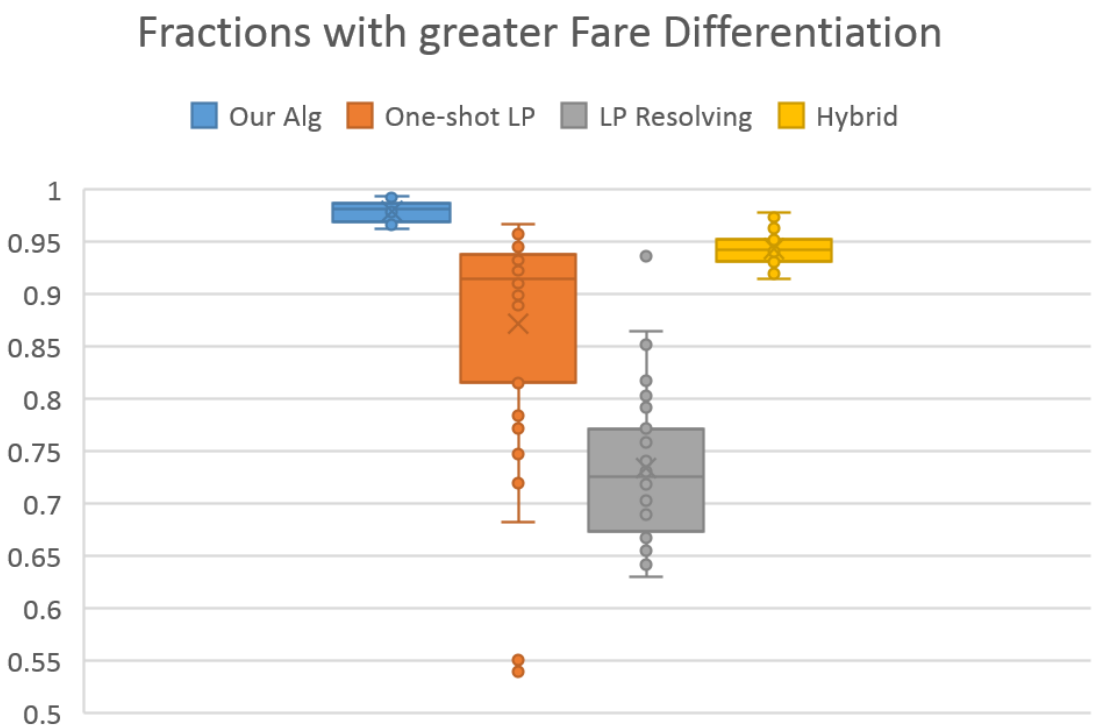
Results



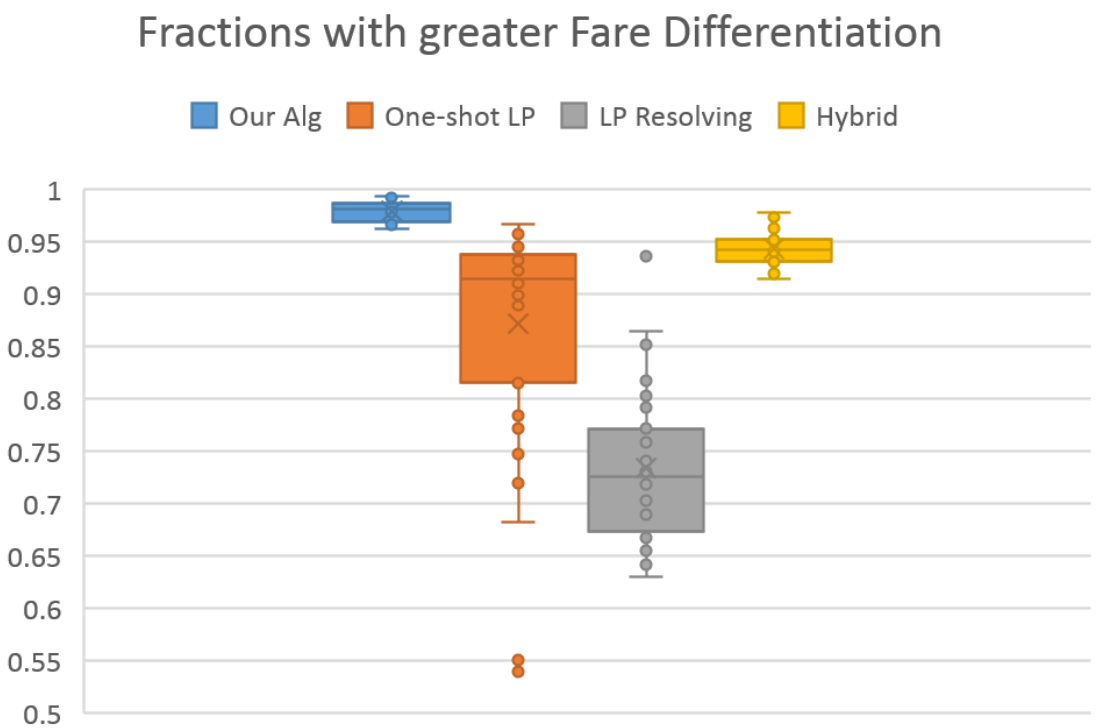
- our algorithm extracts a large fraction of optimum on all instances
- the Hybrid of our algorithm and LP Resolving performs even better
- the forecast-dependent algorithms exhibit a lot more variance
- we also tested additional forecast-independent algorithms

Results—Greater Fare Differentiation

Results—Greater Fare Differentiation



Results—Greater Fare Differentiation



- our algorithm performs better than the Hybrid in this situation

Summary

Summary

- We derive the "worst-case-optimal" function for the value of inventory, when there are both multiple items and multiple prices for each item

Summary

- We derive the "worst-case-optimal" function for the value of inventory, when there are both multiple items and multiple prices for each item
- We extend the applicability of the worst-case approach in revenue management, providing a forecast-independent benchmark which can always be referenced while making decisions

Summary

- We derive the "worst-case-optimal" function for the value of inventory, when there are both multiple items and multiple prices for each item
- We extend the applicability of the worst-case approach in revenue management, providing a forecast-independent benchmark which can always be referenced while making decisions

Complements to Competitive Ratio Analysis:

- Heuristics when the stochastic process generating future arrivals is given (Chan/Farias '09, Ciocan/Farias '12, Chen/Farias '13, Jasin/Kumar '12, Gallego et al. '15)
- Improved algorithms/bounds assuming the arrivals appear in a random order (Kesselheim et al. '13)
- Analyze difference instead of ratio (Reiman/Wang '08, Badanidiyuru/Kleinberg/Slivkins '13, Ferreira/S.-L./Wang '16, Cheung/S.-L. '16)

Thanks!

Email

David Simchi-Levi: dslevi@mit.edu

Papers

1. *Tight Weight-dependent Competitive Ratios for Online Edge-weighted Matching, with Application to Revenue Management*, with **Will Ma** (available on SSRN, 2017)
2. *Dynamic Recommendation at Checkout under Inventory Constraint*, with **Xi Chen**, **Will Ma**, and **Linwei Xin** (available on SSRN, 2016)