Data-Driven Decision-Making
In Enterprise Applications

Dynamic Pricing in Competitive Markets

Rainer Schlosser

Hasso Plattner Institute (EPIC)

June 4, 2020

Outline

- Recall Dynamic Programming

  - Selling Airline Seats

  - Knapsack Problem

- Simulation of Competitive Markets

  - Simple approaches to model Customer Choice

  - Simulation of Customer Decisions

  - Simulation of mutual Price Responses

General Dynamic Programming Framework

- $t = 0, 1, 2, \ldots, T$ Time periods
- $s \in S_t$ States (at time *t*)
- $a \in A_t(s)$ Actions (at time *t* in state s)
- $i \in I_t(a, s)$ Potential Events (at time *t* in state *s* under action *a*)
- $P_t(i, a, s)$ Event Probabilities (for event *i* at *t* in *s* under *a*)
- $R_t(i, a, s)$ Reward Function (for event *i* at *t* in *s* under *a*)
- $\Gamma_t(i, a, s)$ State Transitions (from *s* to new state)
- Solution: Recursive Solution Principle (Bellman Equation)

General Solution Approach (Bellman Equation)

- $V_t(s)$ Value function: "Best expected discounted future rewards in (t,s)"

- $V_T(s) \overset{e.g.}{:=} 0$ Terminal condition for all $s \in S_T$

- Recursive computation of the Value function, $\forall s \in S_t$, $t = 0,...,T-1$,

$$V_t(s) = \max_{a \in A_t(s)} \left\{ \sum_{i \in I_t(a,s)} \underbrace{P(i,a,s)}_{probability} \cdot \left(\underbrace{R_t(i,a,s)}_{today's\ reward} + \underbrace{\delta \cdot V_{t+1}\left(\Gamma_t(i,a,s)\right)}_{best\ disc.\ exp. future\ reward\ of\ new\ state} \right) \right\}$$

- The optimal strategy $a_t^*(s) := \arg\max_{a \in A_t(s)} \{...\}$, $\forall s \in S_t$, $t = 0,...,T-1$,

Example I: Selling Airline Tickets (Monopoly)

- $t = 0, 1, 2, ..., T$ — Time points and periods (t,t+1)
- $s \in S_t$ — **Number of tickets left** (at time t) $S := \{0, 1, ..., N\}$
- $a \in A_t(s)$ — **Offer price** (at period (t,t+1) in state s) $A := \{10, 20, ..., 500\}$
- $i \in I_t(a, s)$ — **Number of requested tickets (Demand)** $I := \{0, 1, ...\}$
- $P_t(i, a, s)$ — **Sales probabilities** (for i tickets at in period t at price a)
- $R_t(i, a, s)$ — Reward function: $R(i, a, s) := a \cdot \min(i, s)$ **(Profits)**
- $\Gamma_t(i, a, s)$ — State transitions: $\Gamma(i, s) := \max(0, s - i)$ **(Items left)**
- Solution: — Recursive Solution Principle (Bellman Equation)

Example II: Knapsack Problem

- $V_t(s)$ — Best utility of "from item set $\{t,\ldots,T\}$ and capacity s"
- $t = 1, 2, \ldots, T+1$ — **Consider sets of items** $\{t,\ldots,T\}$ (T=N total items)
- $s \in \{0, 1, \ldots, C\}$ — **State: Potential capacity left** (max capacity C)
- $a \in A_t(s) := \{0, 1_{\{c_t \leq s\}}\}$ — **Whether to take item *t*** (if the item fits)
- $i \in I_t(a, s)$, $P_t(i, a, s)$ — *No random events (deterministic problem)*
- $R_t(a, s)$ — Reward function: $R_t(a) := a \cdot u_t$ (**utility of item *t***)
- $\Gamma_t(a, s)$ — New state: $\Gamma_t(a, s) := s - a \cdot c_t$ (**remaining capacity**)

DP Solution for the Knapsack Problem

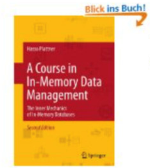
- $V_t(s)$ Best utility of "having the item set $\{t,\ldots,T\}$ and capacity s"
- $V_{T+1}(s) := 0$ Terminal condition for all $s \in \{0,\ldots,C\}$
- Recursive computation of the Value function, $\forall s \in S_t$, $t = 0,\ldots,T-1$,
- $$V_t(s) = \max_{a \in \{0,1_{\{c_t \le s\}}\}} \left\{ \underbrace{a \cdot u_t}_{\text{utility item } t} + \underbrace{\delta \cdot V_{t+1}\left(s - a \cdot c_t\right)}_{\text{best utility of item set } \{t+1,\ldots,T\} \text{ with new capacity}} \right\}$$
- Exercise: Reconstruct the optimal strategy $a_t^*(C)$, $t = 1,\ldots,T$,

II Simulation of Competitive Markets

Motivation Pricing Competition

- Big picture: Modelling dynamic pricing competition
- Separable components: Customers, Markets, Merchants
- How to describe *Customer Behavior*?
- We look for a general model which is simple yet reasonable
- How do you decide?

Example: Buying Books on Amazon



A Course in In-Memory Data Management: The Inner Mechanics of In-Memory Databases (Gebundene Ausgabe)

von Hasso Plattner (Autor)

Schreiben Sie die erste Bewertung

Optimieren durch Alles löschen

Versand
- [ ] ✓Prime
- [ ] Versandkostenfrei

Zustand
- [ ] Neu
- [x] Gebraucht
  - [x] Wie neu
  - [x] Sehr gut
  - [x] Gut
  - [x] Akzeptabel

Preis + Versand (inkl. USt)	Zustand	Verkäufer-Information	Lieferung
EUR 44,90 + EUR 3,00 Versandkosten	**Gebraucht - Akzeptabel** Einband intakt und in sehr gutem Zustand, einige Seiten haben kle... » Weitere Informationen	**ialvamani** ★★★★★ 100% positiv. (4 alle Bewertungen) Verkäuferinformationen, Impressum, AGB, Widerrufsrecht.	• **Ankunft zwischen** April 26 - Mai 2. • Versandtarife
EUR 45,00 + EUR 3,00 Versandkosten	**Gebraucht - Sehr gut** Versand aus Deutschland / We dispatch from Germany via Air Mail. ... » Weitere Informationen	**lange_und_springer_antiquariat** ★★★★★ 98% positiv in den letzten 12 Monaten. (28.584 Bewertungen insgesamt) Verkäuferinformationen, Impressum, AGB, Widerrufsrecht.	• **Ankunft zwischen** April 27 - Mai 2. • Versand aus Deutschland • Versandtarife
EUR 65,60 + EUR 3,00 Versandkosten	**Gebraucht - Wie neu** New, Excellent customer service. Satisfaction guaranteed!!	**Totalbookstore** ★★★★☆ 89% positiv in den letzten 12 Monaten. (439 Bewertungen insgesamt) Verkäuferinformationen, Impressum, AGB, Widerrufsrecht.	• **Ankunft zwischen** Mai 3-20. • Versandtarife
EUR 79,56 + EUR 3,00 Versandkosten	**Gebraucht - Sehr gut** Publisher: Springer Date of Publication: 2014 Binding: hard... » Weitere Informationen	**Herb Tandree Philosophy Books** ★★★★☆ 90% positiv in den letzten 12 Monaten. (338	• **Ankunft zwischen** Mai 2-6. • Versand aus Vereinigtes Königreich • Versandtarife

Customer Choice based on a given Market Situation

seller	price	quality	rating	feedback	shipping
k	p_k	q_k	r_k	f_k	c_k
1	**44.90**	**akzeptabel**	**100%**	**4**	**5 Tage**
2	45.00	sehr gut	98%	**28,584**	6 Tage
3	65.60	**wie neu**	**89%**	439	**11 Tage**
4	**79.56**	sehr gut	90%	338	10 Tage
. . .					
K			. . .		

Simulate Strategic Interaction in Competitive Markets

- Task: Understand & describe customers' decisions over time
- Assume: A product with multiple features (price, quality, ratings)

  A list of competitors' offers (market situation)

  Stream of interested customers + buying decisions
- Goal: Simulate arriving customer and their buying decision

  given a simulated set of competitors' offers

(1) Stream of Arriving Customer

- Any ideas?
- Simulate random delays (waiting times) between two customers
- Use, e.g., *Uniform distributions* or *Exponential distributions*
- Is this doable?

(2) Merchants' Offers & Market Situations

- Simulate offers, i.e., random numbers for prices, quality, ratings

seller k	price p_k	quality q_k	rating r_k
1	**44.90**	**akzeptabel (4)**	**100%**
2	45.00	sehr gut (2)	98%
3	65.60	**wie neu (1)**	**89%**
4	**79.56**	sehr gut (2)	90%
. . .			
K			. . .

(3) Customers' Decision

- Assume: A customer arrives at time t – how does he/she decide?
- Approach I: Always choose the cheapest offer
- Approach II: Use distribution of sales and price rank
- Approach III: Use (randomized) scoring functions
- Other: Combinations, data-driven, etc.

Approach I: Cheapest Offer

- Idea: An interested customer always chooses the cheapest offer
- Easy / deterministic?
- In case of identical prices use probabilities:

$$P(k,\vec{s}) = P(k,\vec{p},\ldots) = \begin{cases} \dfrac{1}{\left|\left\{k=1,\ldots,K : p_k = \min\limits_{i=1,..,K} p_i\right\}\right|} & ,k=1,\ldots,K : p_k = \min\limits_{i=1,..,K} p_i \\ 0 & ,k=1,\ldots,K : p_k > \min\limits_{i=1,..,K} p_i \end{cases}$$

Approach II: Sales vs. Price Rank

- Idea: Relative frequency of sales and price ranks

- Example: 100 sales $\rightarrow$ #60 rank 1, #30 rank 2, #10 rank 3, . . .

  i.e., H sales - $h_1 = 60$, $h_2 = 30$, $h_3 = 10$, . . .

- Simulate the buying probability $P(k, \vec{s})$ that rank k is chosen, $k = 1,...,K$

where $$P(k,\vec{s}) = P(k,\vec{p},...) = \frac{h_{rank(p_k,\vec{p})}}{\sum_{i=1,..,K} h_i}$$

Approach III: Randomized Scoring

- Idea: Different customers use different scoring functions
- Customer Type1: $\arg\min_{k=1,\ldots,K} \left\{ p_k + 0.1 \cdot q_k - 0.01 \cdot r_k - 0.01 \cdot f_k^{0.5} \right\}$
- Customer Type 2: $\arg\min_{k=1,\ldots,K} \left\{ p_k + 0.15 \cdot q_k - 0.005 \cdot r_k - 0.03 \cdot f_k^{0.5} \right\}$
- Customer Type 3: $\arg\min_{k=1,\ldots,K} \left\{ p_k + 0.2 \cdot q_k - 0.05 \cdot r_k - 0.02 \cdot f_k^{0.5} \right\}$

  . . .

- We can model the decision of a random customer as follows:

$$\arg\min_{k=1,\ldots,K} \left\{ p_k + U(0,0.2) \cdot q_k - U(0,0.1) \cdot r_k - U(0,0.05) \cdot f_k^{0.5} \right\}$$

How to Simulate Customer Choice?

- We need: Realisations of (stochastic) buying behavior for various market situations in our models
- Approach I+II: "*Inverse Verteilungsmethode* for $P(k,\vec{s})$ via U(0,1)"
- Approach III: - simulate random scoring coefficients, e.g., U(0,0.05)
  - compute scores for all K offers
  - choose the offer with the best score
- Do you think you can do this?

(4) Combination: Arriving and Buying Customers

- Assume a generated (current) market situation
- Simulate arriving customers over time
- Simulate customers’ individual decisions
- Doable?

(5) Extensions: Changing Market Situations

- (i) Entry / Exit of firms
- (ii) Price adjustments
- Simulate streams of points in time of the merchants' actions ("arrivals")
- Doable?

(6) Demand Learning

- Idea: explain the „dependent variable“ by „explanatory variables“
- „Dependent variable“: number of sales y (of our firm within periods)
- „Explanatory variables“: price rank r

  price difference to best competitor’s price

  ratings, shipping time, . . .

- Remember: Derive the β^* – coefficients for every explanatory variable by linear/logistic regression
- Doable?

(7) Response Strategies

- Assume a merchant can place his/her action at time *t*

- Apply a rule-based price reaction strategy

  (i) Use a random price

  (ii) Undercut the cheapest competitor price

  (iii) Undercut others or raise the price if prices are too cheap

  (iv) Maximize short-term profit

- Doable?

Potential Projects

(1) Index Selection (LP)

(2) Data Placement Problems (LP)

(3) Oligopoly Market Simulation with focus:
Demand Learning (ML) and/or Pricing (DP) and/or Ordering (DP)

(4) Duopoly Competition + Response Strategies (DP)

(5) Own Suggestions

Group Homework: Linear Programming / Dynamic Programming

Project Goals (Teams of 2 – 4)

- Understand & describe your decision problem
- Derive solution approaches
- Apply learned optimization concepts & implement solution
- Simulate results & measure performance
- Presentation: Problem, approach, and early results
- Documentation: Summary of what has been done (until Aug 31)

Overview

Week	Dates	Topic	
1	April 27/30	Introduction + Linear Programming	
2	May 4/ (7)	Linear Programming II	
3	May 11	Exercise Implementations	
4	May 18	Linear + Logistic Regression	(Thu May 21 “Himmelfahrt”)
5	May 25	Dynamic Programming	
6	June 4	Dynamic Pricing Competition	(Mon June 1 “Pfingstmontag”)
7	**June 8**	**Project Assignments**	
8	June 15/18	Robust + Nonlinear Optimization	
9	June 22/25	Work on Projects: Input/Support	
10	June 29/2	Work on Projects: Input/Support	
11	July 6/9	Work on Projects: Input/Support	
12	July 13/16	Work on Projects: Input/Support	
13	July/Aug	Finish Documentation (Deadline: Aug 31)	

(1) Index Selection

(1) Index Selection

Indexes can speed up the execution of queries.

But – indexes require memory and memory is limited.

Further, the impact of indexes is coupled:

The "best" indexes might not form the best selection!

What is "*index interaction*" (IIA)?

➡ The world's best players do not form the best team!

(1) Index Selection – Problem Description

Context: Assume queries with different involved attributes (columns).

Suitable indexes can speed up queries, but require memory.

Decisions: Which *subset* of potential indexes to store?

Note, sets of index candidates and combinations are enormous

Impact: (i) What-if optimizer based costs *(no cost model!*)

(ii) Index interaction! (an index' utility is affected by others)

Constraints: Memory for indexes has a given limit (budget constraint)

Objective: *Minimize runtime s.t. the budget constraint*

(1) Index Selection – LP Formulation

Objective: minimize: Expected runtime (linear)

s.t.
- one index decision only for each query $j=1,\ldots,Q$
- index i used at all?
- budget constraint

Extensions: Stochastic workloads

Robust decisions

(2) Partial Replication

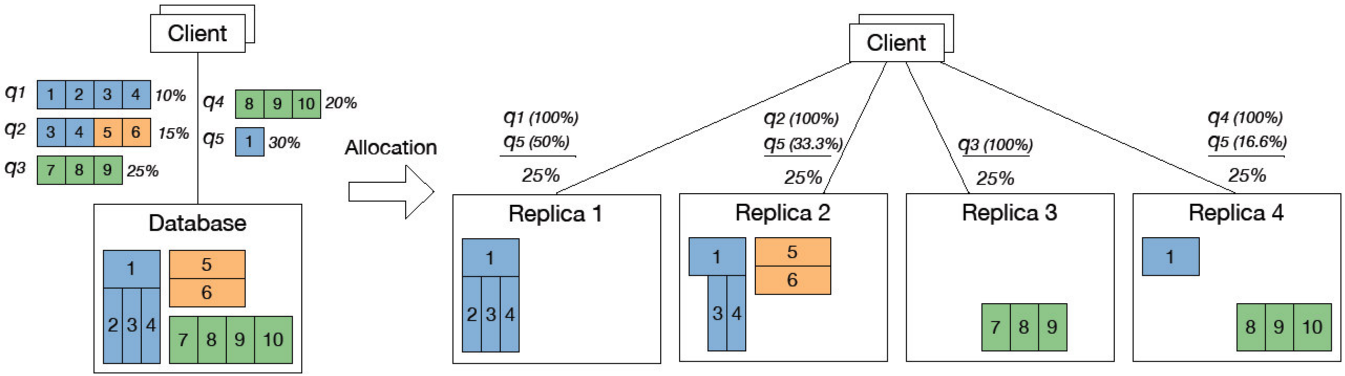
(2) Data Placement for Replication

If the workload exceeds a database's capabilities *replicas* are used (scale-out).

We consider large *read-only* analytical workloads.

Workload can be distributed – but, storing data on replicas is costly!

How can we help the DBA to *balance* workloads with *minimal replicated* data?



(2) Data Placement for Replication

Context: Assume analytical read-only queries using different data fragments.

Workload is generated by queries (frequencies × costs).

Replica nodes take load from the master node.

Decisions: (i) which fragments to put on which replica (data placement)

(ii) which replica shall run which share of a query's workload

Impact: Deterministic

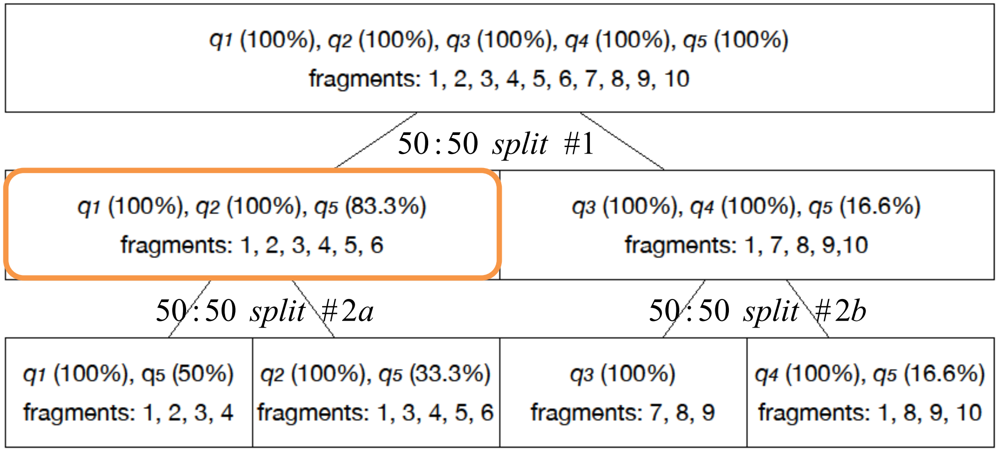
Constraints: (i) *Balance workload **evenly** on replicas*

(ii) To run a query on a replica *all* data fragments are needed

Objective: ***Minimize** costs of replicas (sum of required replicated data)*

(2) Solution Approach: LP-Based Decomposition

Optimization: LP-based decomposition (with scalable sub-problems)

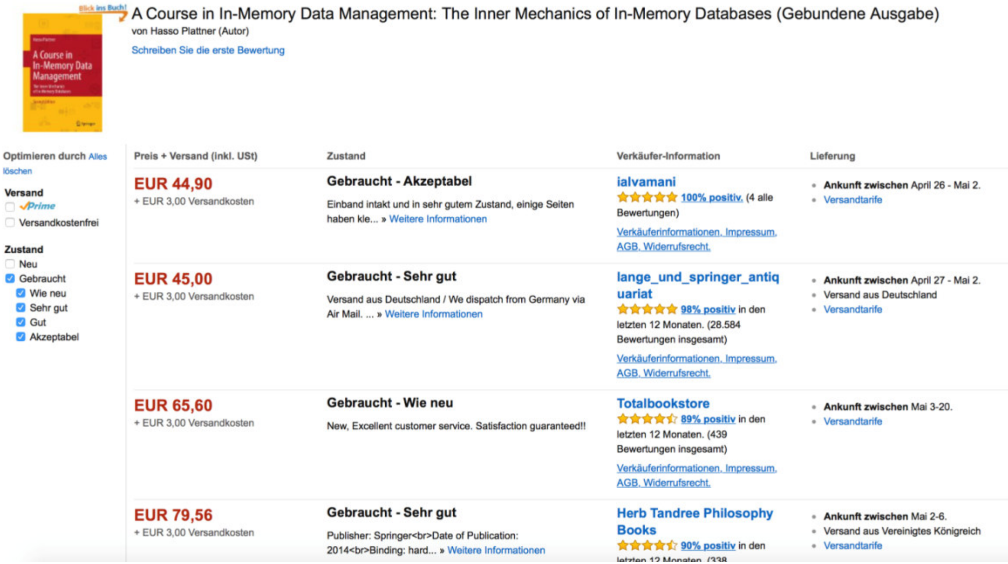


Extensions: Stochastic workloads

Robust decisions

(3) Markets & Demand Learning

(3) Market Simulation & Demand Learning



How can we assist an e-commerce merchant in optimizing his/her prices?

(3) Problem Description

Use-Case: A large merchant sells used books on Amazon Marketplace

Context:
(i) Many distinct items (ISBN), no reordering
(ii) Active competitors, changing environments
(iii) Multiple offer dimensions (quality, ratings, etc.)

Objective: Optimize expected profits & balance profitability vs. speed of sales

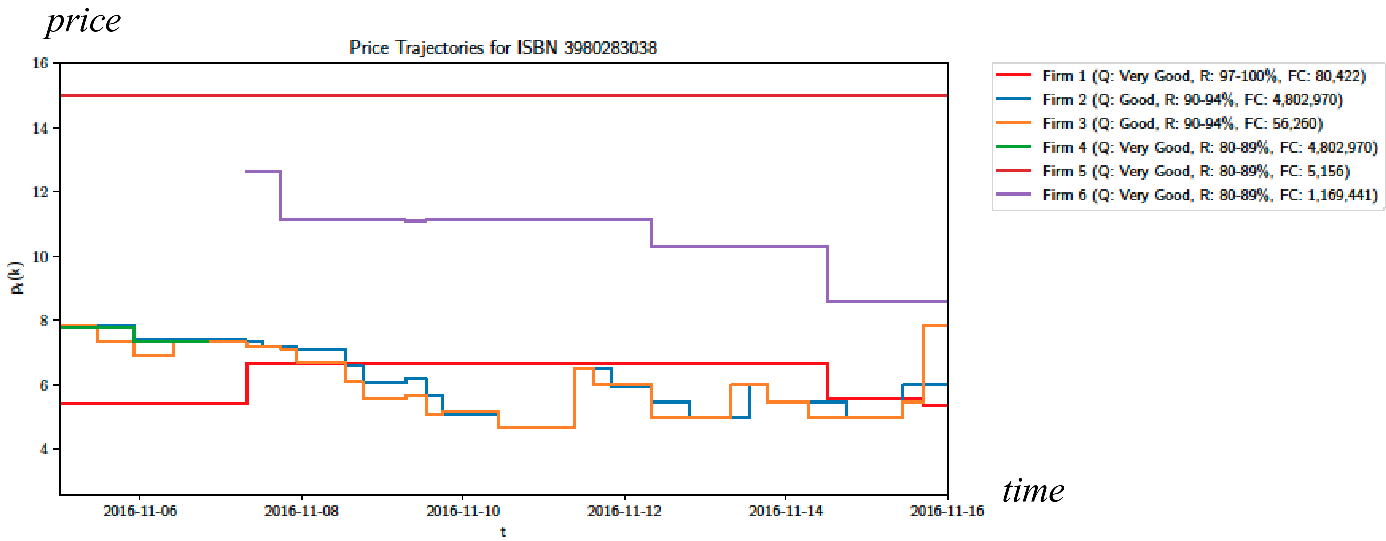
Decisions: Price updates

Impact: Effects of price updates **have to be estimated** from market data

Constraints: Limited inventory, limited price updates/hour

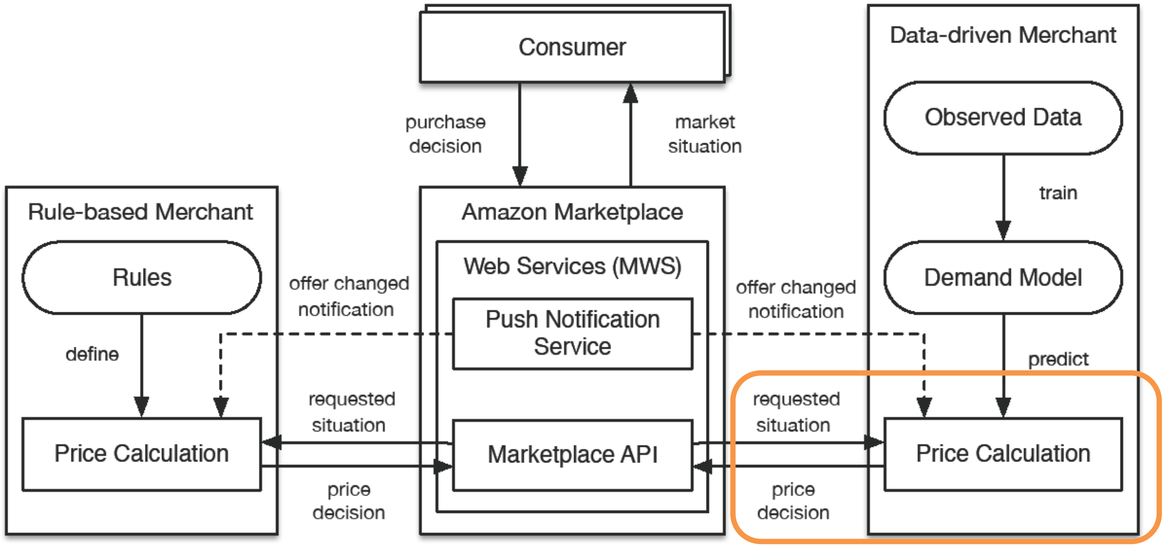
(3) Problem Description

Characteristics:
- Exits & entries of competitors
- Active and passive competitors
- Price cycles



(3) Process

- Price update process on Amazon: (i) request a market situation (ii) optimize price based on demand model, (iii) send price update



(3) Estimation of Demand and Optimization

(1) Estimation of Sales Probabilities

- ca. 10 market situations/day/item with 1-20 firms (100 Mio obs.)
- ca. 2 000 sales/month (1 year of data)
- Predict sales probabilities (for time intervals and market situations)

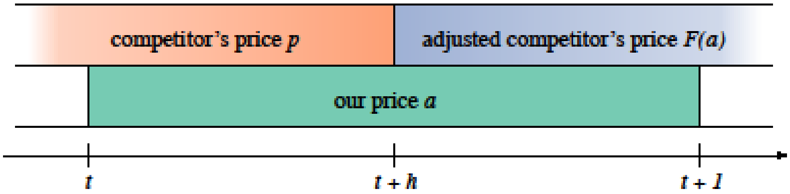
(2) Price *Optimization*

- Maximize expected discounted long-term profit
- Dynamic programming (with relaxed market anticipations)
- Computation time: should be fast

(4) Duopoly Competition

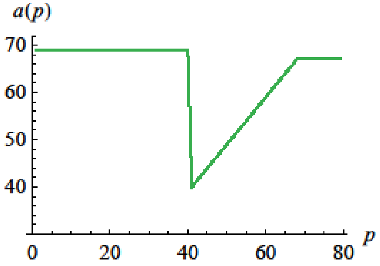
(4) Duopoly Competition & Response Strategies

Question: How do *optimal* price adjustment strategies look like?

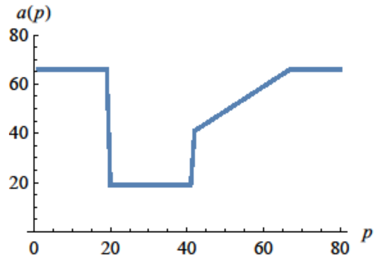


Setting: Infinite horizon, competitor's response strategy *is known*

Results:



against $F(a) := \max(a-1, 1)$

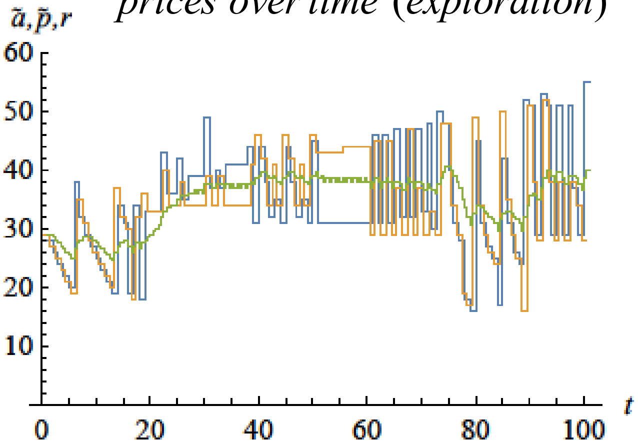


mutual optimal (equilibrium)

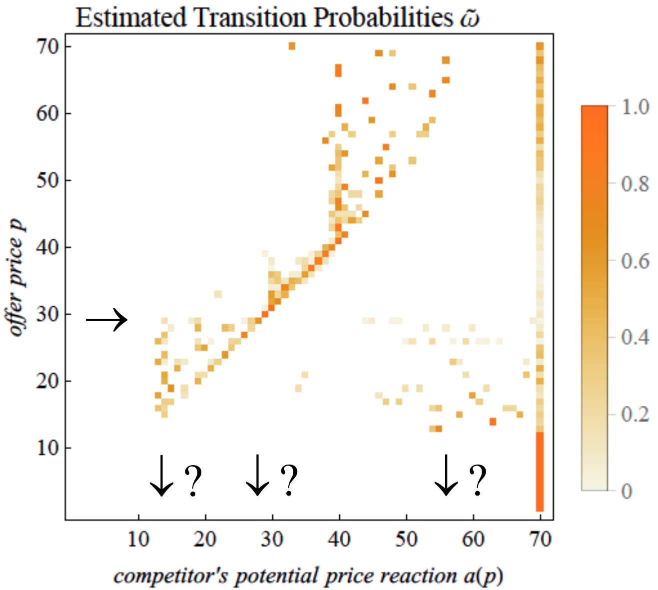
(4) Interaction of Self-Adapting Strategies (Short-Term)

- Now, price responses *have to be learned*!
- Both players update their strategies
- Do equilibria exist?

prices over time (exploration)



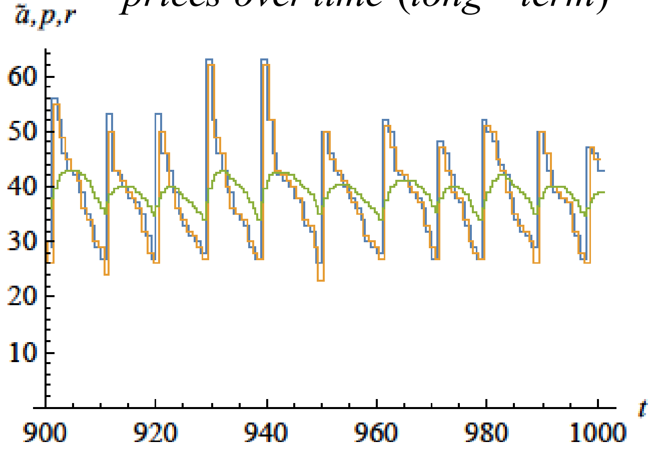
anticipated price reactions



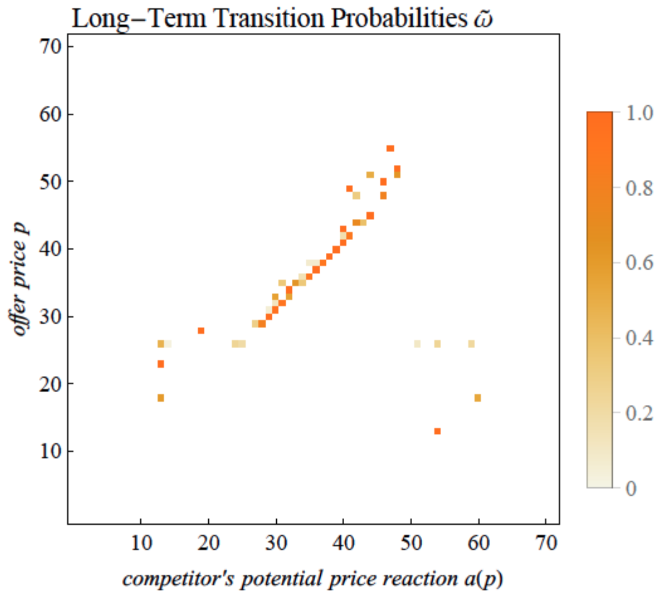
(4) Interaction of Self-Adapting Strategies (Long-Term)

- Now, price responses *have to be learned*!
- Both players update their strategies
- Equilibria in *mixed* strategies

prices over time (long – term)



long – term price reactions



Overview

Week	Dates	Topic	
1	April 27/30	Introduction + Linear Programming	
2	May 4/ (7)	Linear Programming II	
3	May 11	Exercise Implementations	
4	May 18	Linear + Logistic Regression	(Thu May 21 “Himmelfahrt”)
5	May 25	Dynamic Programming	
6	June 4	Dynamic Pricing Competition	(Mon June 1 “Pfingstmontag”)
7	**June 8**	**Project Assignments**	
8	June 15/18	Robust + Nonlinear Optimization	
9	June 22/25	Work on Projects: Input/Support	
10	June 29/2	Work on Projects: Input/Support	
11	July 6/9	Work on Projects: Input/Support	
12	July 13/16	Work on Projects: Input/Support	
13	July/Aug	Finish Documentation (Deadline: Aug 31)	