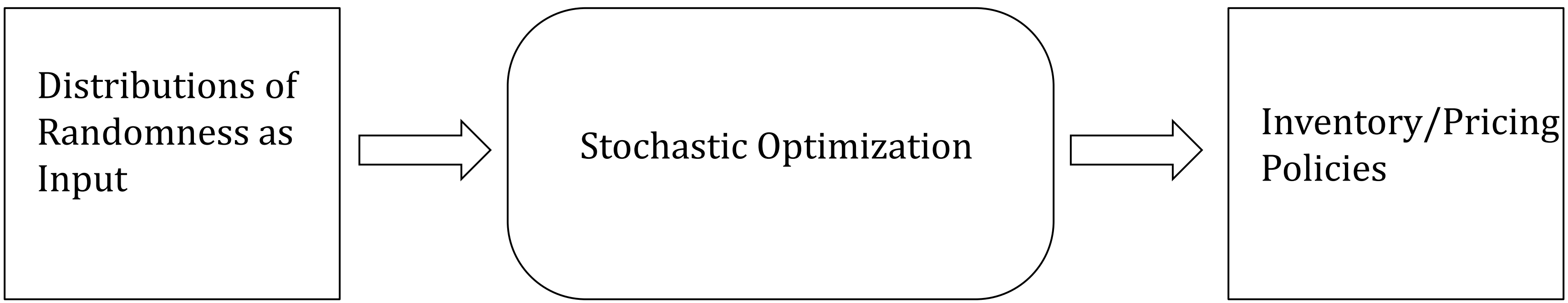
Online Learning Algorithms in Operations Management

VG441 Summer 2020

Cong Shi
Industrial & Operations Engineering
University of Michigan

Conventional View

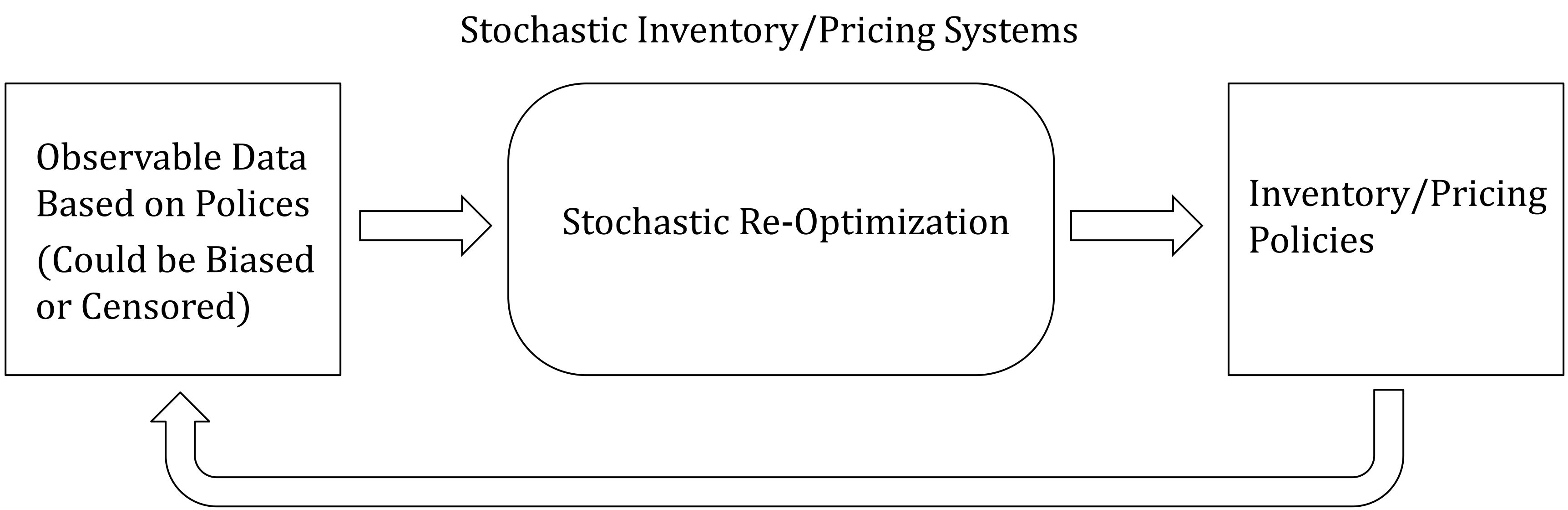
Stochastic Inventory/Pricing Systems



- Model misspecification/estimation errors
- No feedback/self-correcting mechanisms

Optimization as a process

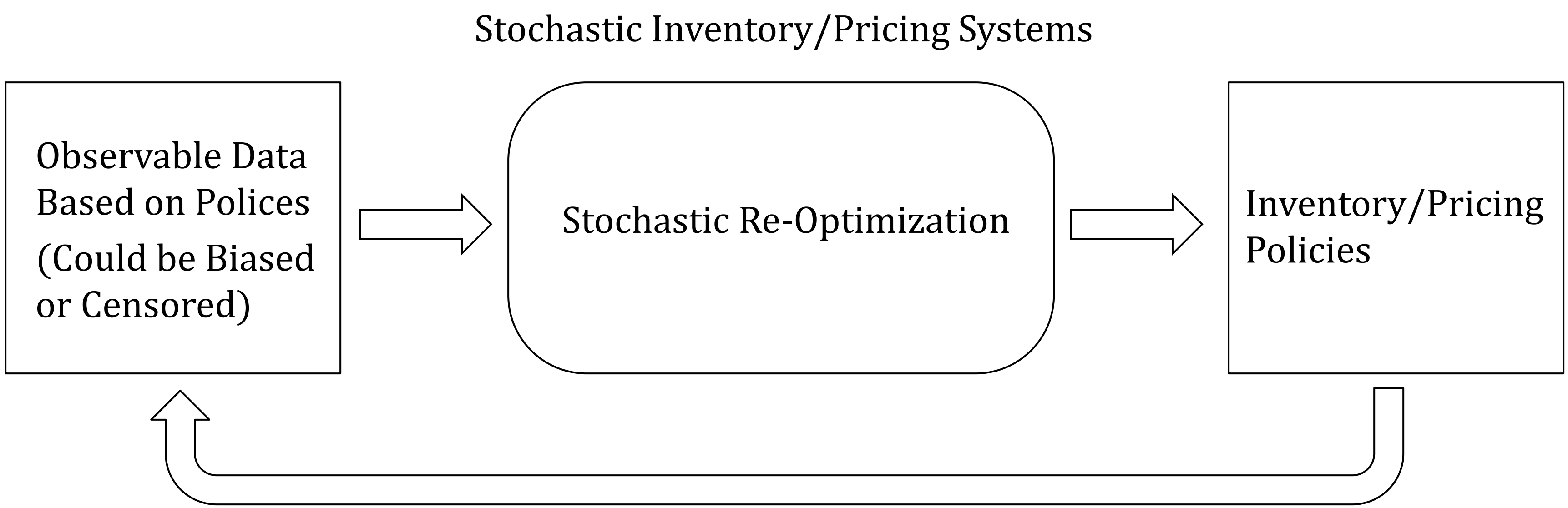
Optimization as a process (a more robust view):
Apply an optimization method that learns as one goes along, learning from experience as more aspects of the problem are observed



- The distributional information on randomness is not known a priori
- No parametric family of distributions is assumed
- Dependence between data and policies
    - Censored demand (can observe sales quantity only, lost-sales quantity is censored)
    - Price-dependent demand

Performance Measure

Optimization as a process (a more robust view)



Devise an efficient nonparametric learning algorithm so that the policy converges to the clairvoyant optimal policy with a provable convergence rate

Cumulative regret = the revenue of the clairvoyant optimal policy – the revenue of the proposed policy <= g(T) which is sublinear in T (preferably also matches the lower bound)

Network Revenue Management with Online Inverse Batch Gradient Descent Method

VG441 Summer 2020

Cong Shi
Industrial & Operations Engineering
University of Michigan

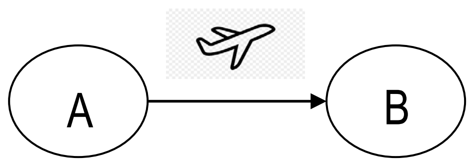
Motivation

Canonical Price-Based Revenue Management (RM)

Airline

Fast Fashion

Hotel



- A single product (e.g., a flight leg A-B)
- A finite initial inventory (e.g., flight leg A-B has 300 seats)
- A finite selling horizon (e.g., $T = 180$ days)
- Customer arrival and valuation process (e.g., willingness to buy given prices)

Question: how to dynamically price the product over $[1, T]$ so as to maximize the total expected revenue?

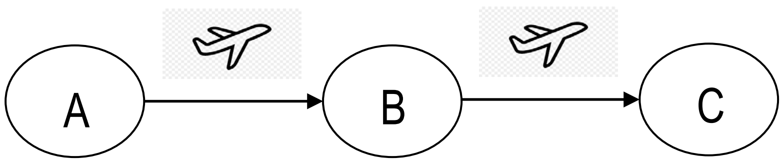
Motivation

Canonical Price-Based **Network** Revenue Management (RM)

Airline

Fast Fashion

Hospitality



- I resources (e.g., leg A-B, leg B-C) with initial inventories (e.g., 300, 200 seats)
- J products (e.g., trip A-B, trip B-C, trip A-B-C)
- A finite selling horizon (e.g., $T = 180$ days)
- Customer arrival and valuation process (e.g., willingness to buy given prices)

Question: how to dynamically price the products over $[1, T]$
so as to maximize the total expected revenue?

Motivation

Canonical Price-Based **Network** Revenue Management (RM)

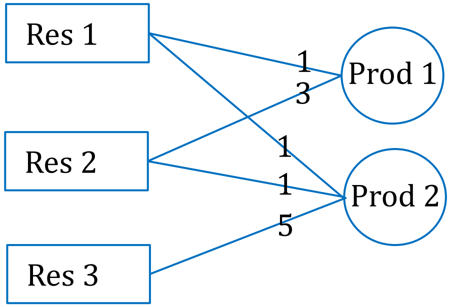
Imagine a real and complex flight network...

General Model

Canonical Price-Based **Network** Revenue Management (RM)

- I resources (initial inventory $\mathbf{x}_0 \triangleq [x_{0,1}, \cdots, x_{0,I}]$)
- J products (how to price them over $[1, T]$?)
- A bill-of-material Matrix

$$\mathbf{A} \triangleq \begin{bmatrix} A_{11} & \dots & A_{1J} \\ \vdots & \ddots & \vdots \\ A_{I1} & \dots & A_{IJ} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 3 & 1 \\ 0 & 5 \end{bmatrix}$$



- One customer arrives in each period
  and chooses one from J products with probabilities (price-dependent)

For example, take the multinomial logit model (MNL)
Given $p_1 = \$100, p_2 = \$80$, then $q_1 = 20\%, q_2 = 70\%, 1 - q_1 - q_2 = 10\%$ (no purchase option)

$$\{\boldsymbol{Q}(\boldsymbol{p}) : \forall\, \boldsymbol{p} \in \mathcal{D}_{\boldsymbol{p}}\} = \{\boldsymbol{q} \geq 0,\ ||\boldsymbol{q}||_1 \leq 1\} \triangleq \mathcal{D}_{\boldsymbol{q}}$$

price space quantile space (purchasing probability)

General Model

Canonical Price-Based **Network** Revenue Management (RM)

$$\{\boldsymbol{Q}(\boldsymbol{p}) : \forall\, \boldsymbol{p} \in \mathcal{D}_{\boldsymbol{p}}\} = \{\boldsymbol{q} \geq 0,\ ||\boldsymbol{q}||_1 \leq 1\} \triangleq \mathcal{D}_{\boldsymbol{q}}$$

Price Space

Quantile (Demand) Space

- Revenue rate function in two equivalent forms:

$$\boldsymbol{q} \triangleq \boldsymbol{Q}(\boldsymbol{p})$$

$$R(\boldsymbol{p}) \triangleq \boldsymbol{p} \cdot \boldsymbol{Q}(\boldsymbol{p}) \qquad R(\boldsymbol{q}) \triangleq \boldsymbol{Q}^{-1}(\boldsymbol{q}) \cdot \boldsymbol{q}$$

Non-concave in $\boldsymbol{p}$

Concave in $\boldsymbol{q}$

- When the function $\boldsymbol{Q}$ is known, simply solve the problem using $R(\boldsymbol{q})$!
- Our work focus on the incomplete information problem (with unknown $\boldsymbol{Q}$)

Complete Information Problem

Canonical Price-Based **Network** Revenue Management (RM)

- Revenue rate function in the quantile space:

$$R(\boldsymbol{q}) \triangleq \boldsymbol{q} \cdot \boldsymbol{Q}^{-1}(\boldsymbol{q})$$

- If the function $\boldsymbol{Q}$ were known, we could solve the problem using $R(\boldsymbol{q})$!
- There are two general approaches to the network RM problem
  - Dynamic Programming (and Approximate Dynamic Programming)
  - Deterministic Linear or Convex Programming (and Re-Optimization)

Two General Approaches

Dynamic Programming (and Approximate Dynamic Programming)

- Let $J^*(\boldsymbol{x}_0, T)$ be the optimal value given initial inventory $\boldsymbol{x}_0$ and remaining time T

reward for selling product j

reward for not selling

$$J^*(\mathbf{x},t) = \max_{\mathbf{q}_t \in \mathcal{D}_{\mathbf{q}}(\mathbf{x},t)} \left\{ \sum_{j=1}^{J} \boxed{q_j \left[Q_j^{-1}(\mathbf{q}_t) + J^*\left(\mathbf{x} - A_j, t-1\right) \right]} + \boxed{q_0 J^*\left(\mathbf{x}, t-1\right)} \right\}$$

- Boundary conditions: $J^*(\mathbf{0}, \cdot) = 0, \qquad J^*(\cdot, 0) = 0.$

Exact computation suffers from the curse of dimensionality!
ADP approaches, e.g., Zhang and Adelman (2009), Kunnumkal and Topaglolu (2010)

Two General Approaches (with known Q)

Deterministic Linear or Convex Programming (DLP)

- Fluid approximation (replacing demand with its expectation)

(DLP)

$$\max_{\boldsymbol{q}} \quad TR(\boldsymbol{q})$$

$$\text{s.t.} \quad \boldsymbol{q} \in \mathcal{D}_{\boldsymbol{q}} \cap \mathcal{D}_{\boldsymbol{I}},$$

$$\mathcal{D}_{\boldsymbol{q}} \triangleq \{\boldsymbol{q} \geq \boldsymbol{0}, \ ||\boldsymbol{q}||_1 \leq 1\}$$

$$\mathcal{D}_{\boldsymbol{I}} \triangleq \{T\boldsymbol{A}\boldsymbol{q} \leq \boldsymbol{x}_0\}$$

Inventory or capacity constraint is enforced on expectation (instead of a.s.)

- Seminal result by Gallego and van Ryzin 1997
  - A tractable upper bound: $J^*(\boldsymbol{x}_0, T) \leq TR(\boldsymbol{q}^*)$
  - If the problem size is scaled by k, the expected revenue loss is $O(\sqrt{k})$
  - The static control is asymptotically optimal

- Can be improved by re-optimization (e.g., Jasin 2014)

Incomplete Information (Demand Learning)

Canonical Price-Based **Network** Revenue Management (RM)

- Revenue rate function in the quantile space:

$$R(\boldsymbol{q}) \triangleq \boldsymbol{q} \cdot \boldsymbol{Q}^{-1}(\boldsymbol{q})$$

- When the function $\boldsymbol{Q}$ is unknown, we learn $\boldsymbol{Q}$ and measure regret:

$$\mathrm{Regret}^{\pi}(\boldsymbol{x}_0, T) \quad \leq \quad T \cdot R(\boldsymbol{q}^*(0)) - \mathsf{E}\left[\sum_{t=1}^{T} R(\boldsymbol{Q}(\boldsymbol{\pi}_t))\right]$$

- Many results on price-based single-leg RM with learning (e.g., Besbes and Zeevi (2009), Araman and Caldentey (2009), Broder and Rusmevichientong (2012), Aviv and Vulcano (2012), Keskin and Zeevi (2014), Wang et al. (2014), Lei et al. (2014), den Boer and Zwart (2015), den Boer (2015))

- Relatively fewer results on price-based network RM with learning
  - Separating exploration and exploitation (Besbes and Zeevi (2012))
  - Spline approximation based approach (Chen, Jasin, Duenyas (2019))
  - Multi-armed bandit: UCB (Badanidiyuru, Kleinberg, and Slivkins (2013))
  - Multi-armed bandit: Thompson Sampling (Ferreira, Simchi-Levi, and Wang (2018))
  - Primal-dual approach (Chen and Gallego (2019))

Two General Approaches with Learning

Separating Exploration and Exploitation

- Blind Network Revenue Management (Besbes and Zeevi 2012)

Exploration Phase:
- Discretize the price space equally $(p_1, \ldots\ p_k)$
- Each price is offered for an equal number of periods

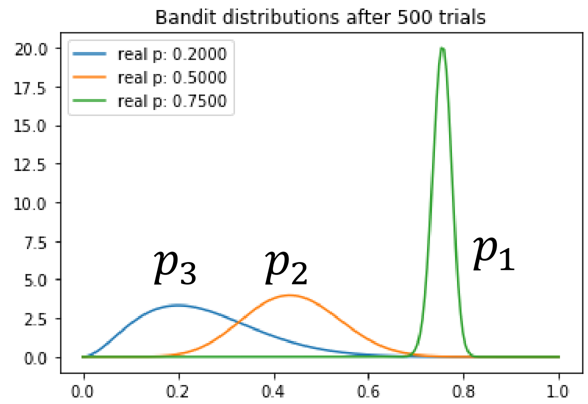
Exploitation Phase:
- Treat empirical demand as true demand
- Solve a DLP and use the static control

- Spline Approximation (Chen, Jasin, and Duenyas 2019)
  - Construct a spline approximation of demand function at the end of exploration

Two General Approaches with Learning

Multi-Armed Bandit Based Approaches (UCB, Thompson Sampling)

- Thompson Sampling (Ferreira, Simchi-Levi, and Wang 2019)
  - Learning while doing type
  - Maintain a prior-posterior distribution on mean demand for each price
  - Sample mean demand from posterior distribution
  - Solve a DLP (using the sampled mean demand) and use static solution
  - Observe demand and update the posterior distribution



- Construct upper confidence reward and lower confidence constr and solve a DLP

Incomplete Information (Demand Learning)

- Existing methods do not exploit concavity of revenue rate function fully
  - Separating exploration and exploitation – losing concavity at the end of exploration, slow performance in high dimensional problems
  - MAB based approaches – doing well only in the discrete price (arm) settings, slow performance in high dimensional problems

- **Can we exploit concavity fully and do stochastic gradient descent (SGD)?**
  - **SGD is fast**
  - **SGD is easy-to-implement**
  - **SGD is dimension-independent**
  - **SGD works very well in a wide range of machine learning settings**

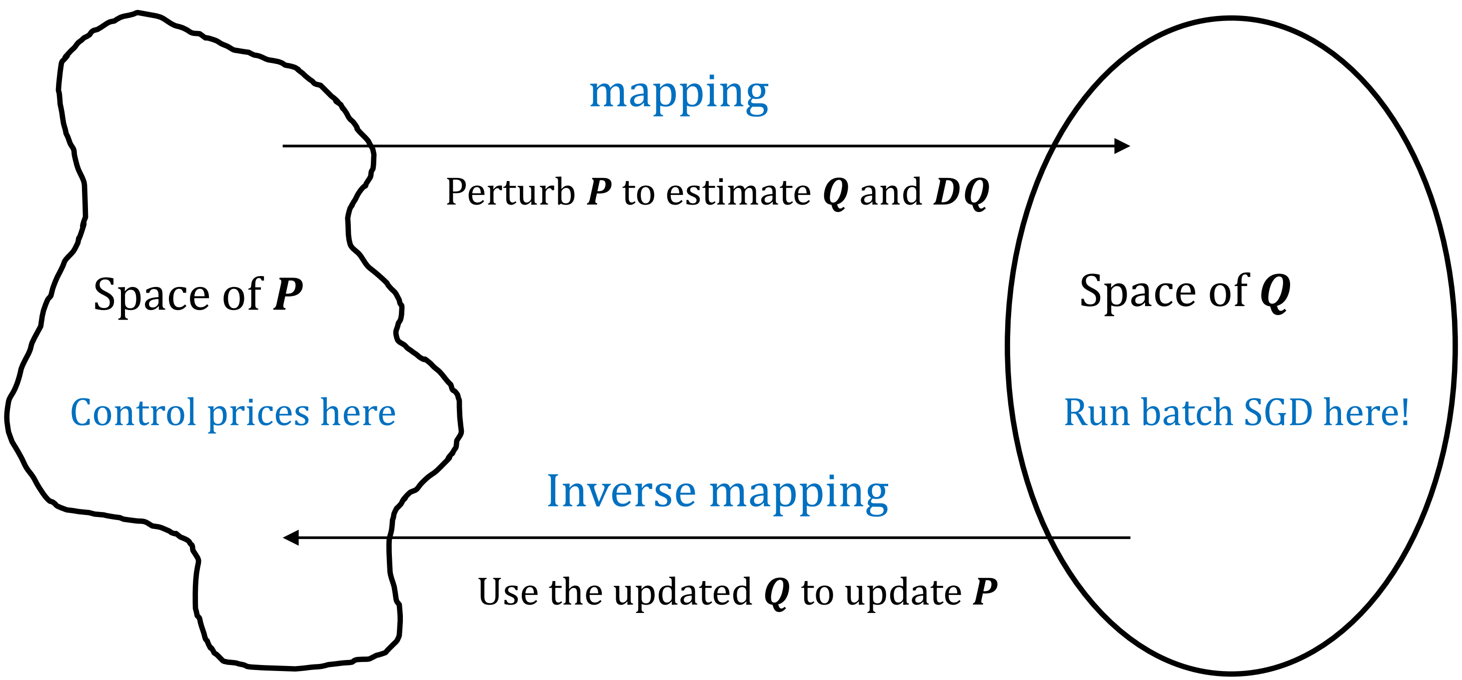
- Major challenges: $R(\boldsymbol{p}) \triangleq \boldsymbol{p} \cdot \boldsymbol{Q}(\boldsymbol{p})$ $\quad R(\boldsymbol{q}) \triangleq \boldsymbol{q} \cdot \boldsymbol{Q}^{-1}(\boldsymbol{q})$

Non-concave in $\boldsymbol{p}$ $\quad$ Concave in $\boldsymbol{q}$

- Cannot directly apply SGD in the price space
- Can only hope to apply SGD in the quantile space (however, $\boldsymbol{Q}$ is unknown!)
- Can only control $\boldsymbol{p}$ directly but not $\boldsymbol{q}$

Inverse Stochastic Gradient Descent

Our IGD method at a very high level



Without “nice” structure in objective

With concave structure in objective

Projected Gradient Descent (Review)

If $\boldsymbol{Q}$ is known, optimize $R(\boldsymbol{q})$ using gradient descent!

1. (Initialization) Number of periods T, a step size η, initial $\boldsymbol{q}_1 \in \mathcal{D}_{\boldsymbol{q}} \cap \mathcal{D}_{\boldsymbol{I}}$

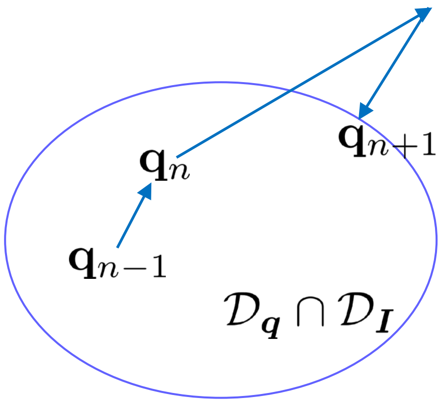
2. (Iteration) For $n \in \{1, \dots, T-1\}$, determine $\boldsymbol{q}_{n+1}$ by

$$\boldsymbol{q}_{n+1} = \mathcal{P}^{\mathrm{GD}} \left(\boldsymbol{q}_n + \eta \nabla_{\boldsymbol{q}} R \left(\boldsymbol{q}_n \right) \right)$$

projecting onto

$$\mathcal{D}_{\boldsymbol{q}} \triangleq \{\boldsymbol{q} \geq \boldsymbol{0},\ ||\boldsymbol{q}||_1 \leq 1\}$$

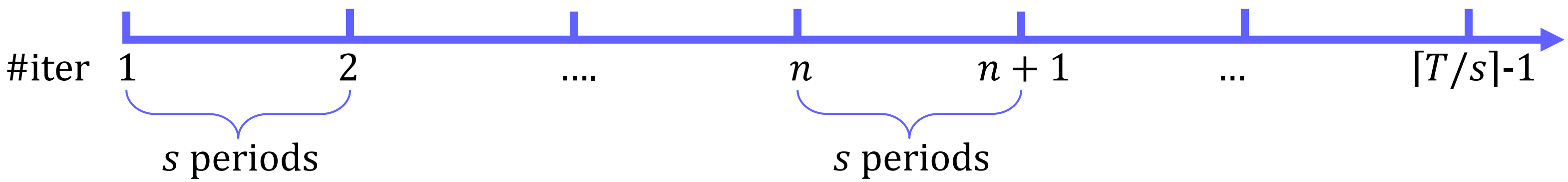
$$\mathcal{D}_{\boldsymbol{I}} \triangleq \{T\boldsymbol{A}\boldsymbol{q} \leq \boldsymbol{x}_0\}$$



Inverse Stochastic Gradient Descent

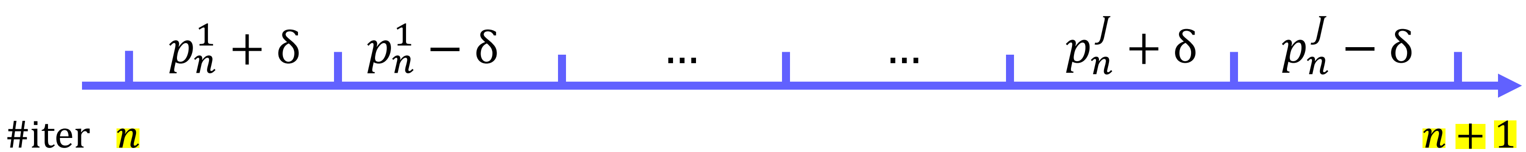
When $\boldsymbol{Q}$ is unknown, develop an inverse stochastic gradient descent

(Initialization) #periods T, step size η, initial $\boldsymbol{p}_1 \in \mathcal{D}_{\boldsymbol{p}}$
batch size s, buffer ϵ, price perturbation δ



(Iteration) Focus on the n-th iteration,

- perturb the reference price vector (J-dim) $\boldsymbol{p}_n$ by $\pm\delta$
- post each of $2J$ prices for $s/(2J)$ periods



Inverse Stochastic Gradient Descent

(Iteration) Focus on the n-th iteration,

- perturb the reference price vector (J-dim) $\boldsymbol{p}_n$ by $\pm\delta$
- post each of $2J$ prices for $s/(2J)$ periods
- compute the empirical quantile $\widehat{\boldsymbol{q}}_n$ and marginal quantile $\widehat{\boldsymbol{DQ}}(\boldsymbol{p})$

	$p_n^1+\delta$	$p_n^1-\delta$	...	...	$p_n^J+\delta$	$p_n^J-\delta$	
#iter n	$\boldsymbol{q}_n^{1,+}$ (sales/#periods)	$\boldsymbol{q}_n^{1,-}$	...	...	$\boldsymbol{q}_n^{J,+}$	$\boldsymbol{q}_n^{J,-}$	$n+1$

$$\hat{\boldsymbol{q}}_n = \frac{1}{2J}\sum_{j=1}^{J}\left(\boldsymbol{q}_n^{j,+} + \boldsymbol{q}_n^{j,-}\right)$$

Kiefer-Wolfowitz type estimator

to approximate

$\boldsymbol{Q}(\boldsymbol{p}_n)$ quantile

marginal quantile

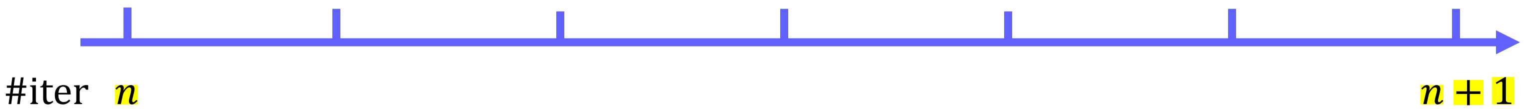
$$\widehat{\boldsymbol{DQ}}_n = \begin{bmatrix} \frac{q_{n,1}^{1,+}-q_{n,1}^{1,-}}{p_{n,1}^{1,+}-p_{n,1}^{1,-}} & \cdots & \frac{q_{n,1}^{J,+}-q_{n,1}^{J,-}}{p_{n,J}^{J,+}-p_{n,J}^{J,-}} \\ \vdots & \ddots & \vdots \\ \frac{q_{n,J}^{1,+}-q_{n,J}^{1,-}}{p_{n,1}^{1,+}-p_{n,1}^{1,-}} & \cdots & \frac{q_{n,J}^{J,+}-q_{n,J}^{J,-}}{p_{n,J}^{J,+}-p_{n,J}^{J,-}} \end{bmatrix}$$

$$\boldsymbol{DQ}(\boldsymbol{p}) \triangleq \begin{bmatrix} \frac{\partial Q_1(\boldsymbol{p})}{\partial p_1} & \cdots & \frac{\partial Q_1(\boldsymbol{p})}{\partial p_J} \\ \vdots & \ddots & \vdots \\ \frac{\partial Q_J(\boldsymbol{p})}{\partial p_1} & \cdots & \frac{\partial Q_J(\boldsymbol{p})}{\partial p_J} \end{bmatrix}$$

Inverse Stochastic Gradient Descent

(Iteration) Focus on the n-th iteration,

- obtain empirical quantile $\hat{\boldsymbol{q}}_n$ and marginal quantile $\widehat{\boldsymbol{DQ}}(\boldsymbol{p})$



Since we can express

$$\nabla_{\boldsymbol{q}} R(\boldsymbol{q}) = \boldsymbol{Q}^{-1}(\boldsymbol{q}) + \boldsymbol{DQ}\left(\boldsymbol{Q}^{-1}(\boldsymbol{q})\right)^{-1,T} \boldsymbol{q}$$

the natural unbiased gradient estimator is

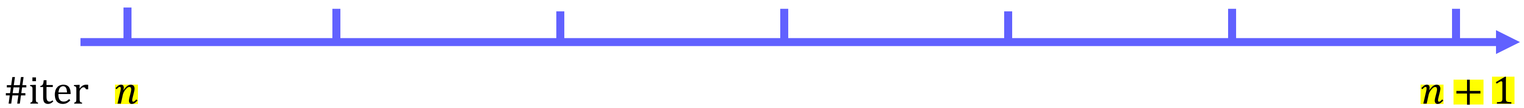
$$\widehat{\nabla R}_n = \boldsymbol{p}_n + \widehat{\boldsymbol{DQ}}_n^{-1,T} \hat{\boldsymbol{q}}_n$$

We use this in the SGD in the quantile space!

Inverse Stochastic Gradient Descent

(Iteration) Focus on the n-th iteration,

- obtain empirical quantile $\widehat{\boldsymbol{q}}_n$ and marginal quantile $\widehat{\boldsymbol{DQ}}(\boldsymbol{p})$



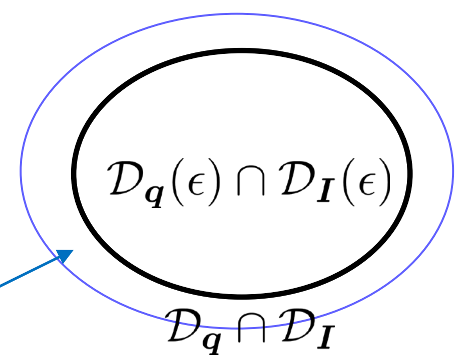
Use $\widehat{\nabla R}_n = \boldsymbol{p}_n + \widehat{\boldsymbol{DQ}}_n^{-1,T} \hat{\boldsymbol{q}}_n$ to carry out SGD in the quantile space:

$$\bar{\boldsymbol{q}}_{n+1} = \mathcal{P}^{\mathrm{IGD}}\left(\hat{\boldsymbol{q}}_n + \eta \widehat{\nabla R}_n\right)$$

projecting onto

$$\mathcal{D}_{\boldsymbol{q}}(\epsilon) \triangleq \left\{\boldsymbol{q} \geq \epsilon \boldsymbol{e}, ||\boldsymbol{q}||_1 \leq 1 - J^{1/2}\epsilon\right\}$$

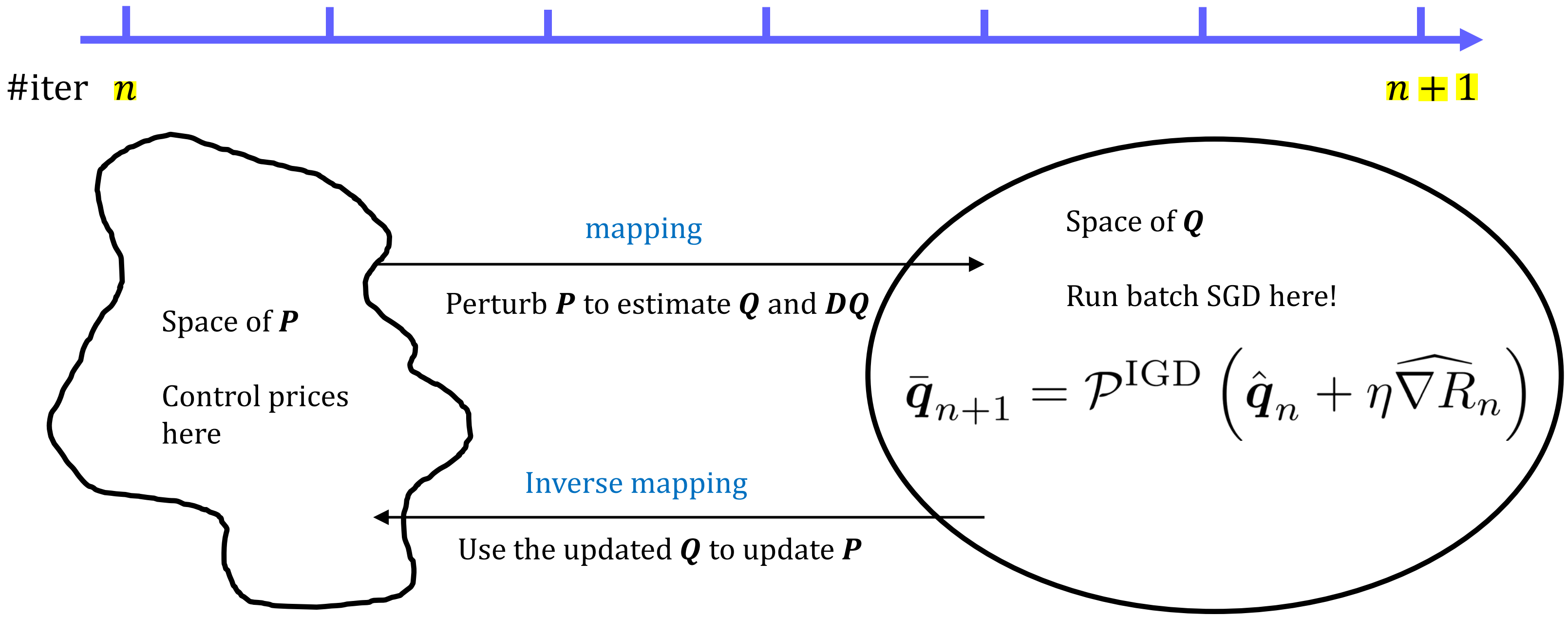
$$\mathcal{D}_{\boldsymbol{I}}(\epsilon) \triangleq \{T\boldsymbol{A}\boldsymbol{q} \leq \boldsymbol{x}_0 - T\boldsymbol{A}\boldsymbol{e}\epsilon\}$$



The small buffer ϵ helps ensure feasibility w.h.p.

Inverse Stochastic Gradient Descent

(Iteration) Focus on the n-th iteration



Inversely map the update back in the price space

$$\boldsymbol{p}_{n+1} = \mathcal{P}_{\boldsymbol{p}}\left(\boldsymbol{p}_n + \boxed{\widehat{\boldsymbol{DQ}}_n^{-1}\left(\bar{\boldsymbol{q}}_{n+1} - \hat{\boldsymbol{q}}_n\right)}\right).$$

Use $\boldsymbol{p}_{n+1}$ in the next iteration

First-order approximation of $\boldsymbol{Q}^{-1}$

Formal Statements of Results

Assumption (1-differentiable, Lipschitz on the first-order):

$||\boldsymbol{DQ}(\boldsymbol{p})||_2 \leq L_1, \; ||\boldsymbol{DQ}(\boldsymbol{p})^{-1}||_2 \leq L_2, \; ||\boldsymbol{DQ}(\boldsymbol{p}+\boldsymbol{\Delta}) - \boldsymbol{DQ}(\boldsymbol{p})||_2 \leq L_3||\boldsymbol{\Delta}||_2$

Theorem: Consider a seq. of instances indexed by k. In the kth problem, all resource's initial inventory is $k\boldsymbol{x}_0$ and horizon is kT. Set batch size $s = k^{3/5}$, buffer size $\epsilon = k^{-1/5}$, price perturbation $\delta = k^{-1/5}$, step size $\eta = k^{-1/5}$

$$\text{Regret}^{\text{IGD}}(k\boldsymbol{x}_0, kT) \quad \leq \quad O\left(k^{4/5}\log k\right)$$

- Dimension independent bound (regardless of #resources #products)
- Projection onto inventory constraints, buffer size ensures feasibility w.h.p.
- Batch size ↑ k, price perturbation ↓ k, step size ↓ k

Prior Results

Besbes and Zeevi (2012):	$k^{\frac{2+J}{3+J}}$, with dependence on #products J
Chen and Gallego (2020):	$k^{1/2}$, single resource type
Chen, Jasin, and Duenyas (2019):	$k^{1/2}$, restrictive demand curvatures

Numerical Results

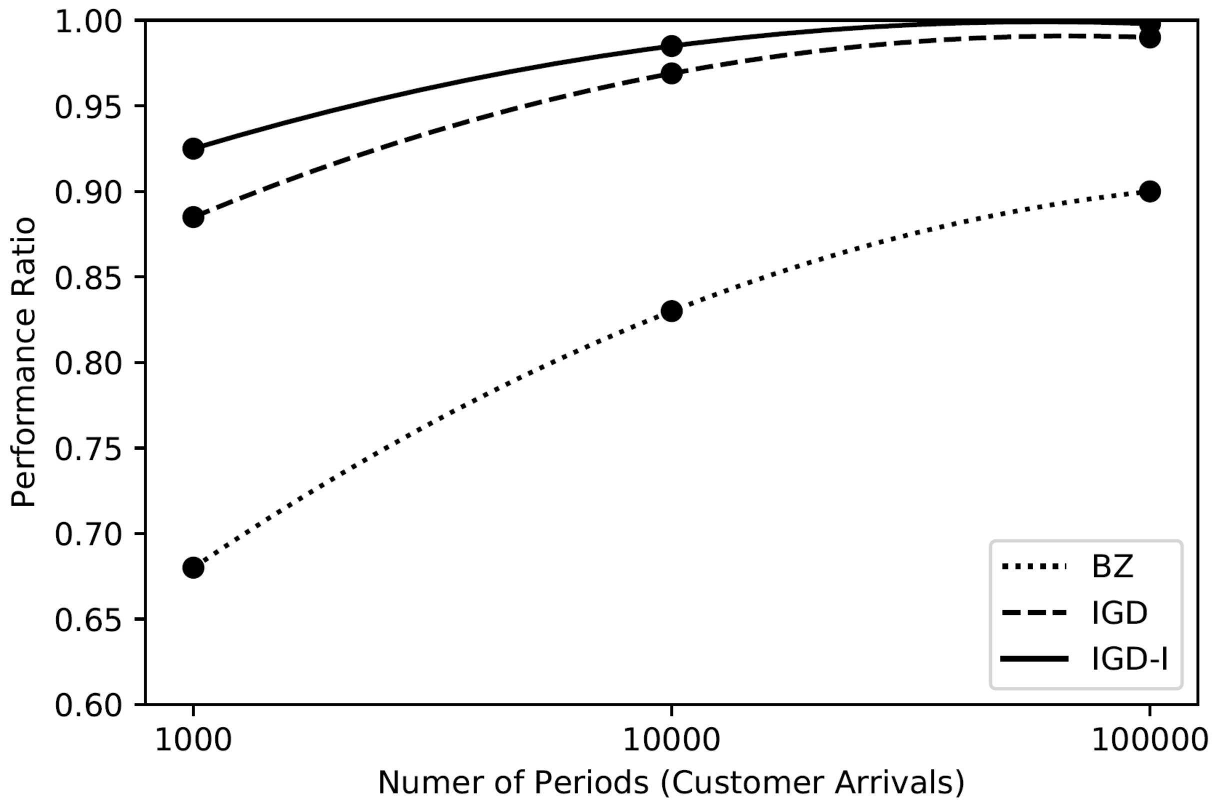
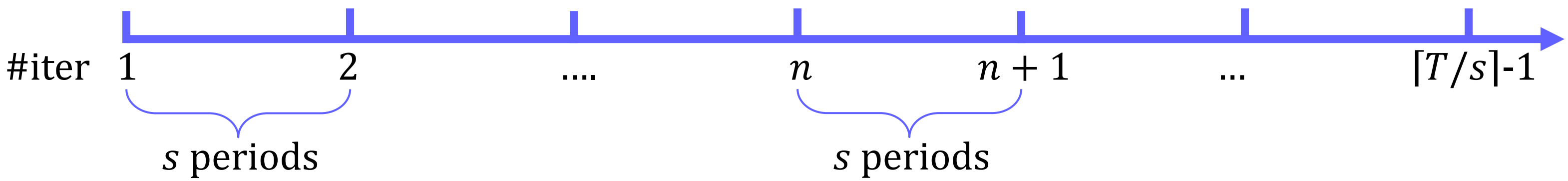


Figure 2 **Performance Comparison of Dynamic Pricing Algorithms**

- Chen, Jasin, Duenyas (2019) has comparable performance to Besbes and Zeevi (2012)
- IGD outperforms "discretized" version of Thompson Sampling with inventories (Ferreira, Simchi-Levi, and Wang 2018) by a large margin

Sketch of Performance Analysis



$$\text{Regret}^{\pi}\left(k\boldsymbol{x}_0, kT\right) \quad \leq$$

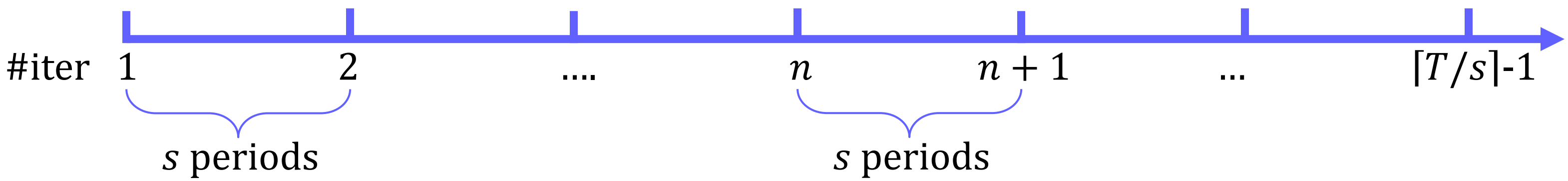
$$\sum_{n=1}^{\lceil T/s \rceil - 1} \left(\mathsf{E}\left[R\left(\boldsymbol{q}^*\left(0\right)\right) - R\left(\boldsymbol{q}_n\right)\right]\right) s \xrightarrow{O\left(k^{4/5}\right)} \boxed{\text{Diff. between } \boldsymbol{p}^* \text{ and } \boldsymbol{p}_n}$$

$$+ \sum_{n=1}^{\lceil T/s \rceil - 1} \sum_{j=1}^{J} \left(\mathsf{E}\left[R\left(\boldsymbol{Q}\left(\boldsymbol{p}_n\right)\right) - R\left(\boldsymbol{Q}\left(\boldsymbol{p}_n^{j,\pm}\right)\right)\right] \frac{s}{J}\right) \xrightarrow{o\left(k^{4/5}\right)} \boxed{\text{Diff. between } \boldsymbol{p}_n \text{ and } \boldsymbol{p}_n^{\pm}}$$

$$+ \left(T - \left(\left\lceil \frac{T}{s} \right\rceil - 1\right) s\right) R\left(\boldsymbol{q}^*\left(0\right)\right) \xrightarrow{o\left(k^{4/5}\right)} \boxed{\text{Loss of last incomplete batch}}$$

$$+ \; \bar{p}\mathsf{E}\left[\left|\left|\left(\boldsymbol{A}\bar{\boldsymbol{Y}} - \boldsymbol{x}_0\right)^+\right|\right|_1\right] \xrightarrow{O\left(k^{4/5}\right)} \boxed{\text{Lost-sales (oversold quantity)}}$$

Sketch of Performance Analysis



To show $O\left(k^{4/5}\right)$

$$\text{Regret}^{\pi}\left(k\boldsymbol{x}_0, kT\right) \leq \sum_{n=1}^{\lceil T/s \rceil - 1} \left(\mathsf{E}\left[R\left(\boldsymbol{q}^*\left(0\right)\right) - R\left(\boldsymbol{q}_n\right)\right]\right) s \;+\; o(k^{4/5})$$

Ensure feasibility w.h.p.

$$\sum_{n=1}^{\lceil T/s \rceil - 1} \left(\mathsf{E}\left[R\left(\boldsymbol{q}^*\left(2\epsilon\right)\right) - R\left(\boldsymbol{q}_n\right)\right]\right) s \;+\; o(k^{-1/5})$$

Bound a key quantity!

$\nabla_{\boldsymbol{q}} R\left(\boldsymbol{q}_n\right)$

$$M_n \triangleq \left|\left|\boldsymbol{q}^*\left(2\epsilon\right) - \boldsymbol{q}_{n+1}\right|\right|_2^2 - \left|\left|\boldsymbol{q}^*\left(2\epsilon\right) - \boldsymbol{q}_n - \eta \boxed{\left(\boldsymbol{p}_n + \boldsymbol{DQ}\left(\boldsymbol{p}_n\right)^{-1,T} \boldsymbol{q}_n\right)}\right|\right|_2^2$$

Loss from approximations – estimating $\boldsymbol{Q}$, $\boldsymbol{DQ}$, $\boldsymbol{DQ}^{-1}$, forward and inverse mapping

Sketch of Performance Analysis

$$\nabla_{\boldsymbol{q}} R(\boldsymbol{q}_n)$$

$$M_n \triangleq ||\boldsymbol{q}^*(2\epsilon) - \boldsymbol{q}_{n+1}||_2^2 - \left|\left|\boldsymbol{q}^*(2\epsilon) - \boldsymbol{q}_n - \eta \boxed{\left(\boldsymbol{p}_n + \boldsymbol{DQ}(\boldsymbol{p}_n)^{-1,T}\boldsymbol{q}_n\right)}\right|\right|_2^2$$

Loss from approximations – estimating $\boldsymbol{Q}$, $\boldsymbol{DQ}$, $\boldsymbol{DQ}^{-1}$, forward and inverse mapping

To show $O(k^{4/5})$

$$\text{Regret}^{\pi}(\boldsymbol{x}_0, T) \leq O\left(\frac{s}{\eta}\mathsf{E}[M_n]\right) + o(k^{4/5})$$

$$\mathsf{E}[M_n|\mathcal{G}^n]\,\mathsf{P}(\mathcal{G}^n) \qquad \mathsf{E}[M_n|\mathcal{G}^{n,c}]\,\boxed{\mathsf{P}(\mathcal{G}^{n,c})}$$

Concentration ineq. $O(k^{-3/5})$

Well-estimated events: estimated quantile suff. close to true quantile

$$\mathcal{G}^n \triangleq \left\{\max\left\{||\boldsymbol{q}_{ml}^{j,+} - \boldsymbol{Q}(\boldsymbol{p}_m^{j,+})||_1, ||\boldsymbol{q}_{ml}^{j,-} - \boldsymbol{Q}(\boldsymbol{p}_m^{j,-})||_1\right\} \leq \alpha L_1\left(p_{m,j}^{j,+} - p_{m,j}^{j,-}\right)\right\}$$

Sketch of Performance Analysis

It suffices to bound $\mathsf{E}\left[M_n | \mathcal{G}^n\right]$ conditional on the well-estimated events

Loss from approximations – estimating $\boldsymbol{DQ}$ and $\boldsymbol{DQ}^{-1}$, forward and inverse mapping

$$\left|\left|\boldsymbol{q}^*(2\epsilon) - \boldsymbol{q}_n - \eta \boxed{\left(\boldsymbol{p}_n + \boldsymbol{DQ}(\boldsymbol{p}_n)^{-1,T}\boldsymbol{q}_n\right)}\right|\right|_2^2$$

$$\nabla_{\boldsymbol{q}} R(\boldsymbol{q}_n)$$

Loss from approx. gradient $\nabla_{\boldsymbol{q}}\boldsymbol{R}$

$$\left|\left|\boldsymbol{q}^*(2\epsilon) - \boldsymbol{q}_n - \eta\widehat{\nabla R_n}\right|\right|_2^2$$

Loss from projection in quantile space

$$\left|\left|\boldsymbol{q}^*(2\epsilon) - \boldsymbol{q}_n - \left(\bar{\boldsymbol{q}}_{n+1} - \hat{\boldsymbol{q}}_n\right)\right|\right|_2^2$$

Loss from approx. mar. quantile $\boldsymbol{DQ}$

$$\left|\left|\boldsymbol{q}^*(2\epsilon) - \boldsymbol{q}_n - \boldsymbol{DQ}(\boldsymbol{p}_n)\widehat{\boldsymbol{DQ}}_n^{-1}\left(\bar{\boldsymbol{q}}_{n+1} - \hat{\boldsymbol{q}}_n\right)\right|\right|_2^2$$

Loss from approx. quantile $\boldsymbol{Q}$

$$\left|\left|\boldsymbol{q}^*(2\epsilon) - \boldsymbol{Q}\left(\boldsymbol{p}_n + \widehat{\boldsymbol{DQ}}_n^{-1}\left(\bar{\boldsymbol{q}}_{n+1} - \hat{\boldsymbol{q}}_n\right)\right)\right|\right|_2^2$$

Loss from projection in price space

$$\left|\left|\boldsymbol{q}^*(2\epsilon) - \boldsymbol{q}_{n+1}\right|\right|_2^2$$

Some Thoughts on the Rate

- Online convex optimization (e.g., Zinevich (2004))

$$O(k^{1/2})$$

- Online convex optimization in the bandit setting: gradient descent without a gradient (e.g., Flaxman et. al. (2005))

$$O(k^{3/4})$$

- Online convex optimization without a gradient and with forward and inverse mappings (our IGD approach)

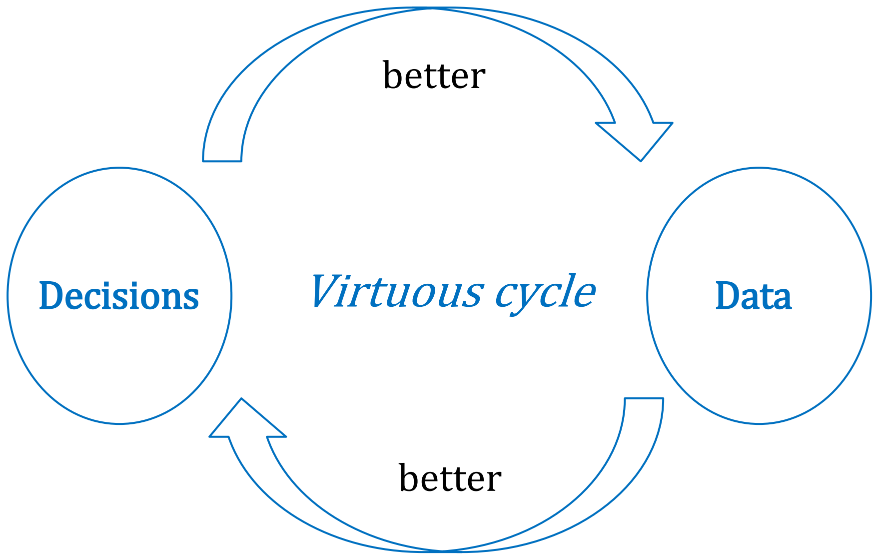
$$O(k^{4/5})$$

Computational rate is far better!

Future Directions

- Revenue management models with reusable resources?

- Other core models whose original objective function is not concave/convex but transformed one is concave/convex?



Thank you for your attention!

Marrying SGD with Bandits: Learning Algorithms for Inventory Systems with Fixed Costs

VG441 Summer 2020

Cong Shi
Industrial & Operations Engineering
University of Michigan

Motivation

Perhaps the most fundamental problem in inventory management

Labor

Machine

Transportation

- A firm makes a sequential ordering decision q_t under stochastic demand
- The ordering cost consists of variable cost c and fixed cost K

$$cq_t + K\mathbb{1}(q_t > 0)$$

- It is well-known that (s, S) policy is optimal (Scarf (1960))
  - Order up to S when the inventory x_t drops below s; do nothing o/w
  - Optimal for the backorder model with arbitrary lead times $L \geq 0$
  - Optimal for the lost sales model with zero lead times $L = 0$

Motivation

There is a long line of literature on inventory control with fixed costs

- Basic model: Scarf (1960), Veinott (1966), Iglehart (1963), Federgruen and Zipkin (1984), Zheng (1991) Generalized demand model: Sethi and Cheng (1997), Gallego and Ozer (2001) Capacitated model: Chen and Lambrecht (1996), Gallego and Scheller-Wolf (2000), Chen (2004) Joint pricing and inventory control: Chen and Simchi-Levi (2004a,b), Chen et al. (2006), Huh and Janakiraman (2008), Feng (2010), Pang et al. (2012), Hu et al. (2018) Quantity dependent fixed costs: Chao and Zipkin (2008), Caliskan-Demirag et al. (2012) Joint replenishment problems: Khouja and Goyal (2008), Cheung et al. (2016), Nagarajan and Shi (2016)

This list goes on and on…

- In practice, the firm **may not know** the demand distribution D *a priori*

$$D_1, D_2, \ldots, D_t, \ldots = D \text{ in distribution}$$

- There is usually **a censored demand phenomenon**
  - Cannot observe d_t but can only observe sales $\min(d_t, x_t + q_t)$
  - The lost-sales customers $(d_t - x_t - q_t)^+$ are not observed!

Goal: Design a nonparametric learning algorithm that uses the **sales** collected over time to minimize the cumulative expected **regret**

Fixed Cost Model under Censored Demand

- **Markov Decision Process:**
  - State: on-hand inventory x_t
  - Control: ordering quantity q_t

Censored demand: observe sales data $\min\{d_t, x_t + q_t\}$

↓

Review inventory level x_t → Decide ordering quantity q_t → Demand D_t realizes as d_t → Incur cost C_t

State transition: $x_{t+1} = (x_t + q_t - d_t)^+$

$$C_t(x_t, q_t, d_t) = \boxed{K\mathbb{1}(q_t > 0)} + \boxed{cq_t} + \boxed{h(x_t + q_t - d_t)^+} + \boxed{p(d_t - x_t - q_t)^+}$$

Fixed cost | Variable cost | Holding cost | Lost sales cost (censored!)

Clairvoyant Optimal Policy: By letting the inventory gap $\delta^* := S^* - s^*$,

$$y_t = \pi_t(x_t) = \begin{cases} S^* & \text{if } x_t \le S^* - \delta^* \\ x_t & \text{if } x_t > S^* \end{cases}$$

s^*

Our Main Result

Our performance benchmark is the notion of regret

Regret of a learning algorithm π for any planning horizon $T \geq 1$:

$$\mathcal{R}_T := \mathbb{E}\sum_{t=1}^{T} C_t^{\pi} - \mathbb{E}\sum_{t=1}^{T} C_t^{\pi^*}$$

Clairvoyant OPT $\pi^* = (\delta^*, S^*)$

Our Main Result:

- Propose the first learning algorithm (δ, S) that attains $\mathcal{R}_T \leq O(\log T\sqrt{T})$
- No algorithms could do better than $\Omega(\sqrt{T})$ even if $K = 0$

Our Methodological Contributions:

- Combined stochastic gradient descent (1^{st} order) with bandit controls (0^{th} order)
- Leveraged a simulation procedure to “exploit” one-side information
- Developed a high probability bound for sub-exponential SGD

Literature Review on Inventory Learning

There is an active and growing literature on nonparametric learning for inventory systems:

- Classical lost-sales model: Burnetas and Smith (2000), Huh and Rusmevichientong (2009), Besbes and Muharremoglu (2013), Huh et al. (2011)
- Lost-sales model with lead times: Huh et al. (2009), Zhang et al. (2019), Shipra and Randy (2019)
- Perishable inventory model: Zhang et al. (2018)
- Capacitated inventory model: Shi et al. (2016), Chen et al. (2018c)
- Joint pricing and inventory control: Chen et al. (2018a,b)

Other parametric learning approaches:

- Bayesian approaches: Scarf (1959), Iglehart (1964), Murray and Silver (1966), Azoury (1985), Lu et al. (2005, 2008), Chen and Plambeck (2008)
- Operational statistics: Liyanage and Shanthikumar (2005) and Chu et al. (2008)
- Cave adaptive estimation: Godfrey and Powell (2001), Powell et al. (2004)

- **The learning problem with fixed costs: opened for quite some time!**
- **The major difficulty lies in that the objective function is not jointly convex in**

$$(s, S) \text{ or } (\delta, S)$$

Cost Transformation and Partial Convexity

$$C_t = K\mathbb{1}(q_t > 0) + cq_t + h(x_t + q_t - d_t)^+ + \boxed{p(d_t - x_t - q_t)^+}$$

Lost sales cost (censored!)

$$\tilde{C}_t = K\mathbb{1}(q_t > 0) + cq_t + h(x_t + q_t - d_t)^+ + \boxed{\cancel{pd_t}} - \boxed{p\min(x_t + q_t, d_t)}$$

Indep. of decisions

Observable!

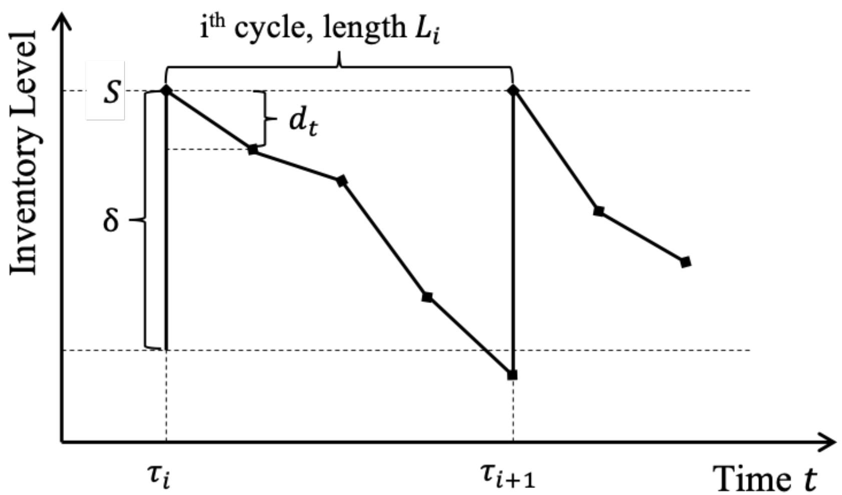
- Given fixed (δ, S), the cycle pseudo cost is observable and also convex in S!

Cycle pseudo cost

$$\lim_{T\to\infty} \frac{1}{T}\mathbb{E}\sum_{t=1}^{T} \tilde{C}_t^{(\delta,S)} = \frac{\mathbb{E}G(\delta, S)}{\mathbb{E}L(\delta, S)}$$

Cycle length

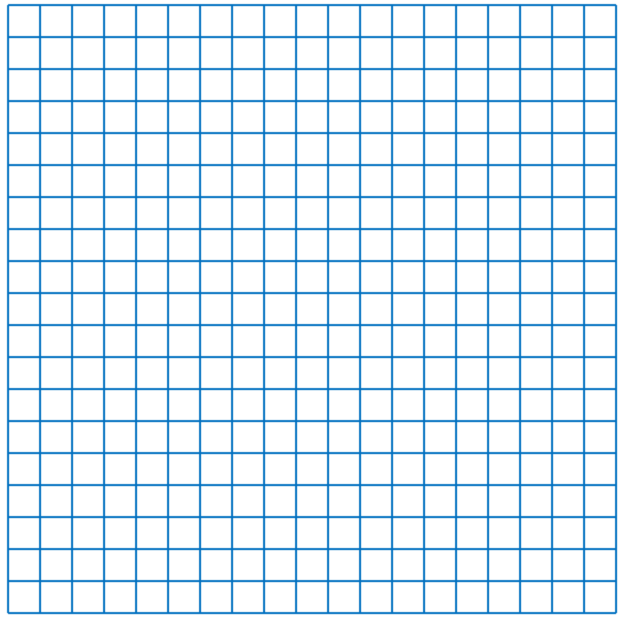
$$\pi^* = (\delta^*, S^*) \in \arg\min_{(\delta,S)} \frac{\mathbb{E}G(\delta, S)}{\mathbb{E}L(\delta, S)}$$



First Line of Thought

- Our problem can be thought of a 2-dimensional continuum-armed bandit problem (treating **cycle cost as “period” cost** and **ignoring inventory carryover**)

Discretize S



Discretize δ

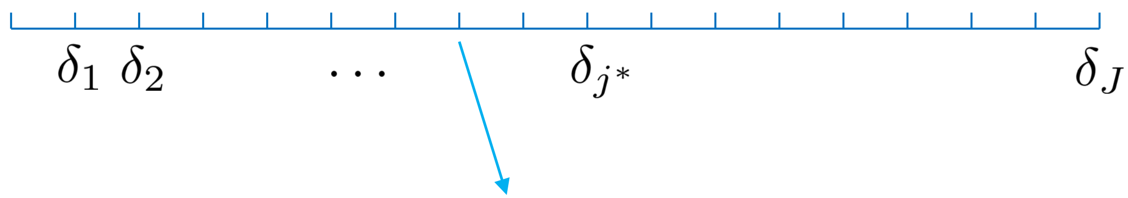
- Apply Kleinberg et al. (2008) or Bubeck et al. (2011)

$$\mathcal{R}_T \leq O\left(T^{\frac{d+1}{d+2}}\right) \quad \text{where } d = 2$$

Not really satisfactory given the lower bound $\Omega(\sqrt{T})$

How about Integrating Bandits with SGD?

- What if we run a bandit control on δ (via discretized arms)?



For each selected δ, we run SGD on the cycle cost!

- Question 1: How do we select which δ to run in each iteration?

  Ans: Use confidence interval type rules to narrow down!

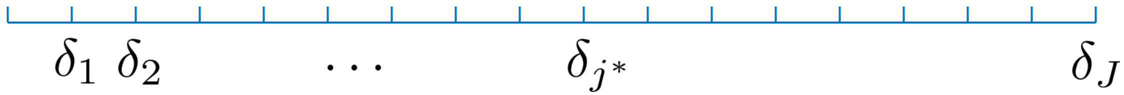
- Question 2: Is this fast enough to get $O(\sqrt{T})$?

  Ans: If done right, can only get to $O(T^{2/3})$!

Question: Can we exploit more structure of this problem?

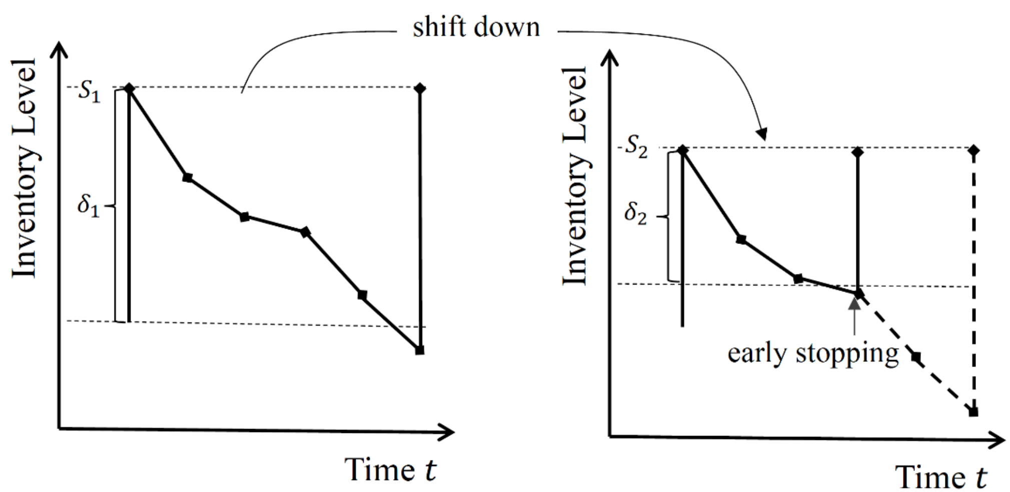
Exploit One-Side Information

- What if we run a bandit control on δ?
- For each δ, we run SGD on the cycle cost!



- Yes, for inventory problems, we can exploit one-side information!

Given 2 policies (δ_1, S_1) and (δ_2, S_2) with $\delta_1 \geq \delta_2$ and $S_1 \geq S_2$ the demands collected from (δ_1, S_1) can simulate (δ_2, S_2)



Simulation of All Active Policies

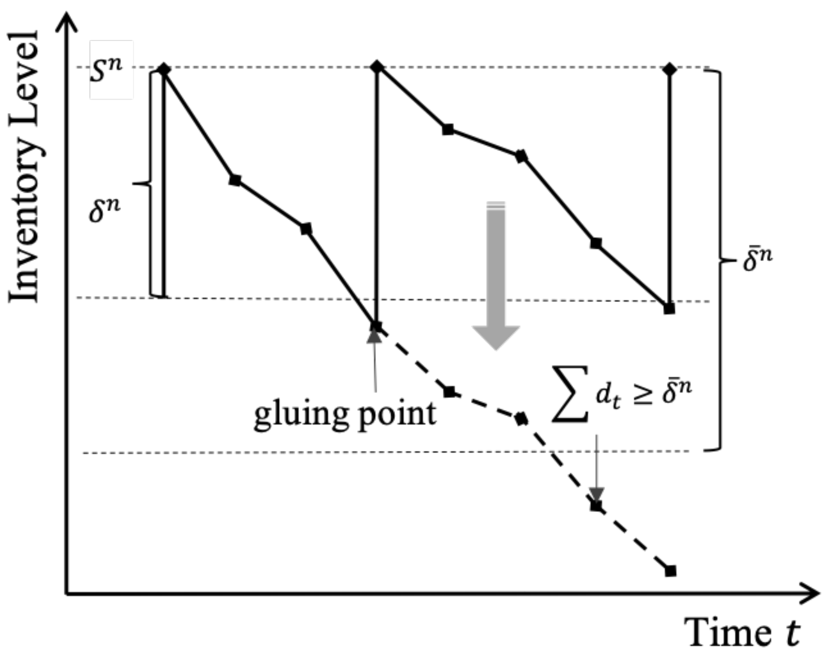
- Define the active set A^n = all "favorable" policies thus far

$\mathcal{A}^n =$ δ_2 δ_4 δ_9 δ_{15}

$S^n =$ 10 20 30 15

Ideally

- Cannot always run $\left(\max_{j \in \mathcal{A}^n} \delta_j, \max_{j \in \mathcal{A}^n} S_j^n\right) := (\bar{\delta}^n, S_{j^n}^n)$ → Not feasible
- Can only run $(\delta_{j^n}, S_{j^n}^n)$



Run $(\delta_{j^n}, S_{j^n}^n)$

Until $\sum_{d \in \mathcal{D}^n} d \geq \bar{\delta}^n$

Our Learning Algorithm

- Initialize the active set $\mathcal{A}^1 = \{1, \ldots, J\}$
- For each iteration $n = 1, 2, \ldots$

  Run $(\delta_{j^n}, S_{j^n}^n := \max_{j \in \mathcal{A}^n} S_j^n)$ until $\sum_{d \in \mathcal{D}^n} d \geq \bar{\delta}^n := \max_{j \in \mathcal{A}^n} \delta_j$

  Information-maximizing action

  To the point that all active policies can be simulated

- **(Simulation + 1st order)** Use the demands collected to simulate active policies in $j \in \mathcal{A}^n$, compute the cycle cost G_j^n, length L_j^n and gradient $\tilde{\nabla}_j^n$

$$S_j^{n+1} = \mathbf{Proj}_{[\delta_j, \beta]}\left(S_j^n - \eta_n \tilde{\nabla}_j^n\right), \quad \hat{G}_j^n = \hat{G}_j^{n-1} + G_j^n, \quad \hat{L}_j^n = \hat{L}_j^{n-1} + L_j^n.$$

- **(0th order)** Update and prune the active set

$$\mathcal{A}^{n+1} = \left\{ j \in \mathcal{A}^n : \frac{\hat{G}_j^n}{\hat{L}_j^n} - \min_{j' \in \mathcal{A}^n} \frac{\hat{G}_{j'}^n}{\hat{L}_{j'}^n} \leq \Delta^n \right\}.$$

Confidence Size

Pruning procedure with high probability (narrowing down):
Remove policy *j* if its lower confidence bound is higher than the upper confidence bound of the currently best policy

The Learning Algorithm: An Example

Iter #1

$\mathcal{A}^1 =$ 1 2 3 4 5 6 7 8

$S^1 =$ 10 15 30 40 38 36 30 28

- Play information-maximizing action (4, 40) until cumulative demand crosses 8
- Simulate all active policies (1,2,3,4,5,6,7,8)
- Perform SGD on all active polices
- Prune the active set (according to confidence interval type rule)

Iter #2

$\mathcal{A}^2 =$ 2 4 5 6

$S^2 =$ 25 35 37 32

- Play information-maximizing action (5, 37) until cumulative demand crosses 6
- Simulate all active policies (2,4,5,6)
- Perform SGD on all active polices
- Prune the active set (according to confidence interval type rule)

Iter #3

$\mathcal{A}^3 =$ 4

$S^3 =$ 33

Exploration-Exploitation Trade-Off

More exploration $\longrightarrow$ More exploitation

T

Beginning Iterations of the Algorithms:

- Larger active sets $\mathcal{A}^n$ (exploring more favorable policies)
- Larger order-up-to levels (with longer cycle time)
- Ensuring sufficient demand information to simulate all active policies

Later Iteartions of the Algorithms:

- Smaller active sets $\mathcal{A}^n$ (exploitating more empirically sound policies)
- Near-optimal order-up-to levels (with near-optimal frequency)

Formal Statements of Results

Main Result 1 (given the number of cycles):

$$J = \lfloor \sqrt{N} \rfloor, \ \eta_n = \frac{\beta}{\xi\sqrt{n}}, \text{ and } \Delta^n = \frac{2\theta \log(8N^2)}{\sqrt{n}}$$

$$\text{We have regret } \mathcal{R}_N \leq O\left(\log N \sqrt{N}\right).$$

Main Result 2 (anytime algorithm using doubling trick):

$$\text{We have regret } \mathcal{R}_N \leq O\left(\log N \sqrt{N}\right).$$

(a) Partition epochs into groups of exponentially increasing lengths
{1}, {2,3}, {4,5,6,7}, {8,9,10,11,12,13,14,15},...
(b) Apply our algorithm for each group with parameters chosen according to the group length.

Numerical Results

distribution	fixed cost K	optimal policy δ^*	S^*	optimal average cost	relative regret r_T (%) 125	250	500	1000
uniform	50	225.73	340.11	1015.76	6.38	5.67	4.97	4.92
	100	489.62	547.43	1030.49	4.66	4.25	3.61	3.52
	150	917.26	981.60	1036.74	5.17	3.88	2.95	2.86
gamma ($\alpha = 3$)	50	384.17	478.08	1015.05	5.08	5.05	4.98	4.65
	100	578.73	667.05	1031.77	4.56	3.79	3.64	3.53
	150	644.77	734.10	1038.75	4.50	3.74	3.65	3.50
gamma ($\alpha = 5$)	50	390.64	472.82	1017.20	5.30	5.20	4.85	4.65
	100	338.99	466.06	1029.54	5.80	4.37	3.80	3.22
	150	687.52	771.72	1041.12	4.11	3.37	3.28	3.06
gamma ($\alpha = 7$)	50	489.09	548.55	1018.23	4.84	4.00	3.99	3.72
	100	453.56	554.17	1030.30	5.11	3.87	3.28	2.76
	150	730.55	782.93	1043.23	3.85	3.01	2.83	2.81
exponential	50	599.53	648.39	1006.59	8.32	7.18	6.66	6.55
	100	703.03	751.85	1011.24	8.51	7.37	7.09	6.49
	150	756.28	851.91	1022.77	7.43	6.61	5.92	5.85
lognormal ($\sigma = 0.1$)	50	245.59	310.61	1025.98	4.25	3.07	2.55	2.32
	100	347.24	406.68	1040.84	3.14	2.01	1.75	1.65
	150	425.54	526.58	1049.70	3.38	2.05	1.43	1.21

Table 1 Performance of the (δ, S) algorithm with varying fixed costs K.

Analysis Sketch

Establishing 3 bridging policies (that successively relax the problem)

- Our algorithm: $(\delta_{j^n}, \max(x^n, S^n_{j^n}))$
- Ignore inventory "overshoot": $(\delta_{j^n}, S^n_{j^n})$
- Optimal-S oracle: $(\delta_{j^n}, S^*_{j^n})$

  where $S^*_{j^n} = \arg\min_S V(\delta_{j^n}, S)$
- Optimal on grid: $(\delta_{j^*}, S^*_{j^*})$

  where $j^* = \arg\min_{j\in\{1,\ldots,J\}} V^*_j$
- The optimal: (δ^*, S^*)

Inventory carry-over loss

- Queueing theory argument $O(\sqrt{N})$

Synchronized SGD loss

- Online convex optimization $O(\sqrt{N})$

Pruning loss

- Sub-exp SGD + bandit $O(\log(N)\sqrt{N})$

Discretization loss

- Lipschitz $O(\sqrt{N})$

Analysis Sketch

Establishing 3 bridging policies (that successively relax the problem)

- Our algorithm: $(\delta_{j^n}, \max(x^n, S^n_{j^n}))$
- Ignore inventory "overshoot": $(\delta_{j^n}, S^n_{j^n})$
- Optimal-S oracle: $(\delta_{j^n}, S^*_{j^n})$
  where $S^*_{j^n} = \arg\min_S V(\delta_{j^n}, S)$
- Optimal on grid: $(\delta_{j^*}, S^*_{j^*})$
  where $j^* = \arg\min_{j\in\{1,\ldots,J\}} V^*_j$
- The optimal: (δ^*, S^*)

Inventory carry-over loss
- Queueing theory argument $O(\sqrt{N})$

Synchronized SGD loss
- Online convex optimization $O(\sqrt{N})$

Pruning loss
- Sub-exp SGD + bandit $O(\log(N)\sqrt{N})$

Discretization loss
- Lipschitz $O(\sqrt{N})$

Proof Sketches:
- Prove a technical result on Lipschitz continuity of an ascending RW
- Prove the optimal average cycle cost $V^*(\delta)$ is Lipschitz in δ
- Pick the grid size $J = \sqrt{N}$

Analysis Sketch

Establishing 3 bridging policies (that successively relax the problem)

- Our algorithm: $(\delta_{j^n}, \max(x^n, S^n_{j^n}))$
- Ignore inventory "overshoot": $(\delta_{j^n}, S^n_{j^n})$
- Optimal-S oracle: $(\delta_{j^n}, S^*_{j^n})$
  where $S^*_{j^n} = \arg\min_S V(\delta_{j^n}, S)$
- Optimal on grid: $(\delta_{j^*}, S^*_{j^*})$
  where $j^* = \arg\min_{j\in\{1,\dots,J\}} V^*_j$
- The optimal: (δ^*, S^*)

Inventory carry-over loss
- Queueing theory argument $O(\sqrt{N})$

Synchronized SGD loss
- Online convex optimization $O(\sqrt{N})$

Pruning loss
- Sub-exp SGD + bandit $O(\log(N)\sqrt{N})$

Discretization loss
- Lipschitz $O(\sqrt{N})$

Pruning Loss

- Optimal-S oracle: $(\delta_{j^n}, S^*_{j^n})$
  where $S^*_{j^n} = \arg\min_S V(\delta_{j^n}, S)$
- Optimal on grid: $(\delta_{j^*}, S^*_{j^*})$
  where $j^* = \arg\min_{j\in\{1,\dots,J\}} V^*_j$

Pruning loss

- Sub-exp SGD + bandit $O(\log(N)\sqrt{N})$

Developed a new sub-exponential SGD high probability regret bound:

Set the step size $\eta_i = \frac{\beta}{\xi\sqrt{i}}$, then with probability at least $1-\delta$,

$$\frac{1}{n}\sum_{i=1}^{n}\left[f(z_i) - f(z^*)\right] \leq \max\left\{\sqrt{\frac{2\beta^2\nu^2\log(1/\delta)}{n}}, \frac{2b\log(1/\delta)}{n}\right\} + \frac{3\beta\xi}{2\sqrt{n}},$$

where $z^* = \arg\min_{z\in\mathcal{K}} f(z)$.

Then can be used to show

$$\mathbf{P}\left(\left|\hat{V}^n_j - V^*_j\right| > \frac{\theta\log(8N^2)}{\sqrt{n}}\right) \leq \frac{1}{N^2} \longrightarrow$$

Estimation error bound

Pruning Loss (continued)

Estimation error bound $\mathbf{P}\left(\left|\hat{V}_j^n - V_j^*\right| > \frac{\theta \log(8N^2)}{\sqrt{n}}\right) \leq \frac{1}{N^2}$

Let $\Delta^n = \frac{2\theta \log(8N^2)}{\sqrt{n}}$

Well-estimated events: $A = \left\{ \text{for all } j \in [J], n \in [N] \text{ we have } \left|\hat{V}_j^n - V_j^*\right| < \Delta^n/2 \right\}.$

Condition on A:
$$V_{j^n}^* - V_{j^*}^* = \left(V_{j^n}^* - \hat{V}_{j^n}^{n-1}\right) + \left(\hat{V}_{j^n}^{n-1} - \hat{V}_{j^*}^{n-1}\right) + \left(\hat{V}_{j^*}^{n-1} - V_{j^*}^*\right)$$
$$\leq \frac{1}{2}\Delta^{n-1} + \Delta^{n-1} + \frac{1}{2}\Delta^{n-1} = 2\Delta^{n-1}$$

Use estimation error bound

$$\mathbb{E}\left[\sum_{n=1}^{N}\left(V_{j^n}^* - V_{j^*}^*\right)\right] = \mathbb{E}\left[\sum_{n=1}^{N}\left(V_{j^n}^* - V_{j^*}^*\right)|A\right]\mathbb{P}[A] + \mathbb{E}\left[\sum_{n=1}^{N}\left(V_{j^n}^* - V_{j^*}^*\right)|A^c\right]\mathbb{P}[A^c]$$
$$\leq \mathbb{E}\left[\sum_{n=1}^{N}\left(V_{j^n}^* - V_{j^*}^*\right)|A\right] + \gamma\sqrt{N} \leq \sum_{n=1}^{N} 2\Delta^{n-1} + \gamma\sqrt{N} = O(\log N\sqrt{N})$$

Key: to show that sub-exponential SGD error and pruning error are **additive**!

Analysis Sketch

Establishing 3 bridging policies (that successively relax the problem)

- Our algorithm: $(\delta^n, \max(x^n, S^n_{j^n}))$
- Ignore inventory "overshoot": $(\delta^n, S^n_{j^n})$
- Optimal-S oracle: $(\delta^n, S^*(\delta^n))$
  where $S^*(\delta^n) = \arg\min_S V(\delta^n, S)$
- Optimal on grid: $(\delta_{j^*}, S^*(\delta_{j^*}))$
  where $j^* = \arg\min_{j \in \{1,\ldots,J\}} V^*_j$
- The optimal: (δ^*, S^*)

Inventory carry-over loss
- Queueing theory argument $O(\sqrt{N})$

Synchronized SGD loss
- Online convex optimization $O(\sqrt{N})$

Pruning loss
- Sub-exp SGD + bandit $O(\log(N)\sqrt{N})$

Discretization loss
- Lipschitz $O(\sqrt{N})$

Slightly Modified Online Convex Optimization Analysis for Synchronized SGD Loss

Analysis Sketch

Establishing 3 bridging policies (that successively relax the problem)

- Our algorithm: $(\delta_{j^n}, \max(x^n, S^n_{j^n}))$
- Ignore inventory "overshoot": $(\delta_{j^n}, S^n_{j^n})$
- Optimal-S oracle: $(\delta_{j^n}, S^*_{j^n})$
  where $S^*_{j^n} = \arg\min_S V(\delta_{j^n}, S)$
- Optimal on grid: $(\delta_{j^*}, S^*_{j^*})$
  where $j^* = \arg\min_{j\in\{1,\ldots,J\}} V^*_j$
- The optimal: (δ^*, S^*)

Inventory carry-over loss
- Queueing theory argument $O(\sqrt{N})$

Synchronized SGD loss
- Online convex optimization $O(\sqrt{N})$

Pruning loss
- Sub-exp SGD + bandit $O(\log(N)\sqrt{N})$

Discretization loss
- Lipschitz $O(\sqrt{N})$

Inventory Carryover Loss

Establishing 3 bridging policies (that successively relax the problem)

- Our algorithm: $(\delta_{j^n}, \max(x^n, S^n_{j^n}))$
- Ignore inventory "overshoot": $(\delta_{j^n}, S^n_{j^n})$
- Optimal-S oracle: $(\delta_{j^n}, S^*_{j^n})$

  where $S^*_{j^n} = \arg\min_S V(\delta_{j^n}, S)$
- Optimal on grid: $(\delta_{j^*}, S^*_{j^*})$

  where $j^* = \arg\min_{j\in\{1,\dots,J\}} V^*_j$
- The optimal: (δ^*, S^*)

Inventory carry-over loss
- Queueing theory argument $O(\sqrt{N})$

Synchronized SGD loss
- Online convex optimization $O(\sqrt{N})$

Pruning loss
- Sub-exp SGD + bandit $O(\log(N)\sqrt{N})$

Discretization loss
- Lipschitz $O(\sqrt{N})$

Proof Sketches: Relate to a GI/GI/1 queue

$$\left(S_{n+1} - S^{j_{n+1}}_{n+1}\right) \le \Big[\left(S_n - S^{j_n}_n\right) + \underbrace{c_1\eta L^n}_{\text{SGD}} + \underbrace{c_2\mathbf{1}_{\{j_n\notin\mathbf{A}_{n+1}\}}}_{\text{Active set pruning}} - D_n\Big]^+$$

Lemma: Queueing system $Z_{n+1} = \mathrm{Proj}_{[0,\beta]}(Z_n + A_n - D_n)$

A_n independent of D_n and $\sum_{n=1}^N A_n = O(\sqrt{N})$ Then $\mathbf{E}\left[\sum_{n=1}^N Z_{n+1}\right] = O(\sqrt{N})$

Analysis Sketch

Establishing 3 bridging policies (that successively relax the problem)

- Our algorithm: $(\delta_{j^n}, \max(x^n, S^n_{j^n}))$
- Ignore inventory "overshoot": $(\delta_{j^n}, S^n_{j^n})$
- Optimal-S oracle: $(\delta_{j^n}, S^*_{j^n})$
  where $S^*_{j^n} = \arg\min_S V(\delta_{j^n}, S)$
- Optimal on grid: $(\delta_{j^*}, S^*_{j^*})$
  where $j^* = \arg\min_{j\in\{1,\ldots,J\}} V^*_j$
- The optimal: (δ^*, S^*)

Inventory carry-over loss
- Queueing theory argument $O(\sqrt{N})$

Synchronized SGD loss
- Online convex optimization $O(\sqrt{N})$

Pruning loss
- Sub-exp SGD + bandit $O(\log(N)\sqrt{N})$

Discretization loss
- Lipschitz $O(\sqrt{N})$

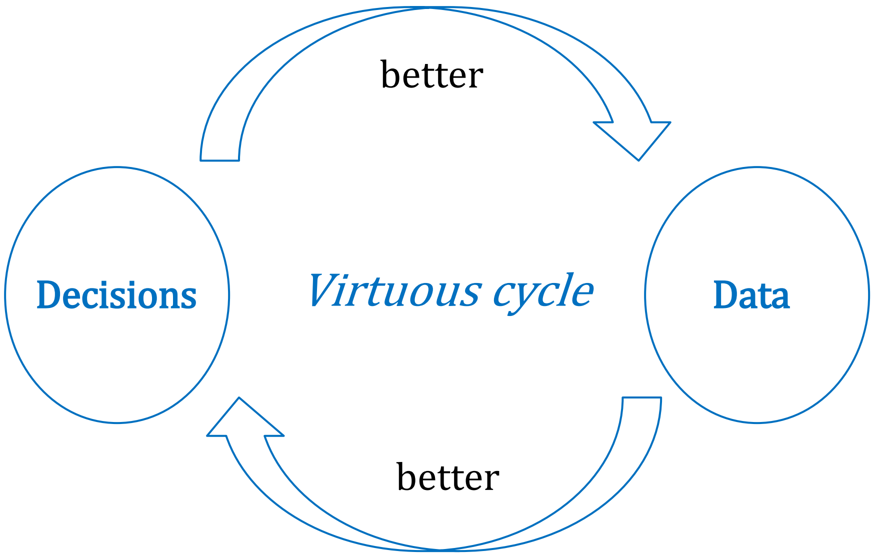
Future Directions

- Other models with fixed costs

- Other models with partial convexity or concavity properties
  e.g., dual-sourcing inventory systems

- More generally…

  - Demand censoring
  - Lasting impact of decisions on costs
  - Complex state transfer
  - Physical inventory constraints



Thank you for your attention!