



# **Competitive Analysis of Online Booking: Dynamic Policies**



*Michael Ball, Huina Gao, Yingjie Lan, Itir Karaesmen*

*R. H. Smith School of Business*

*University of Maryland*

and

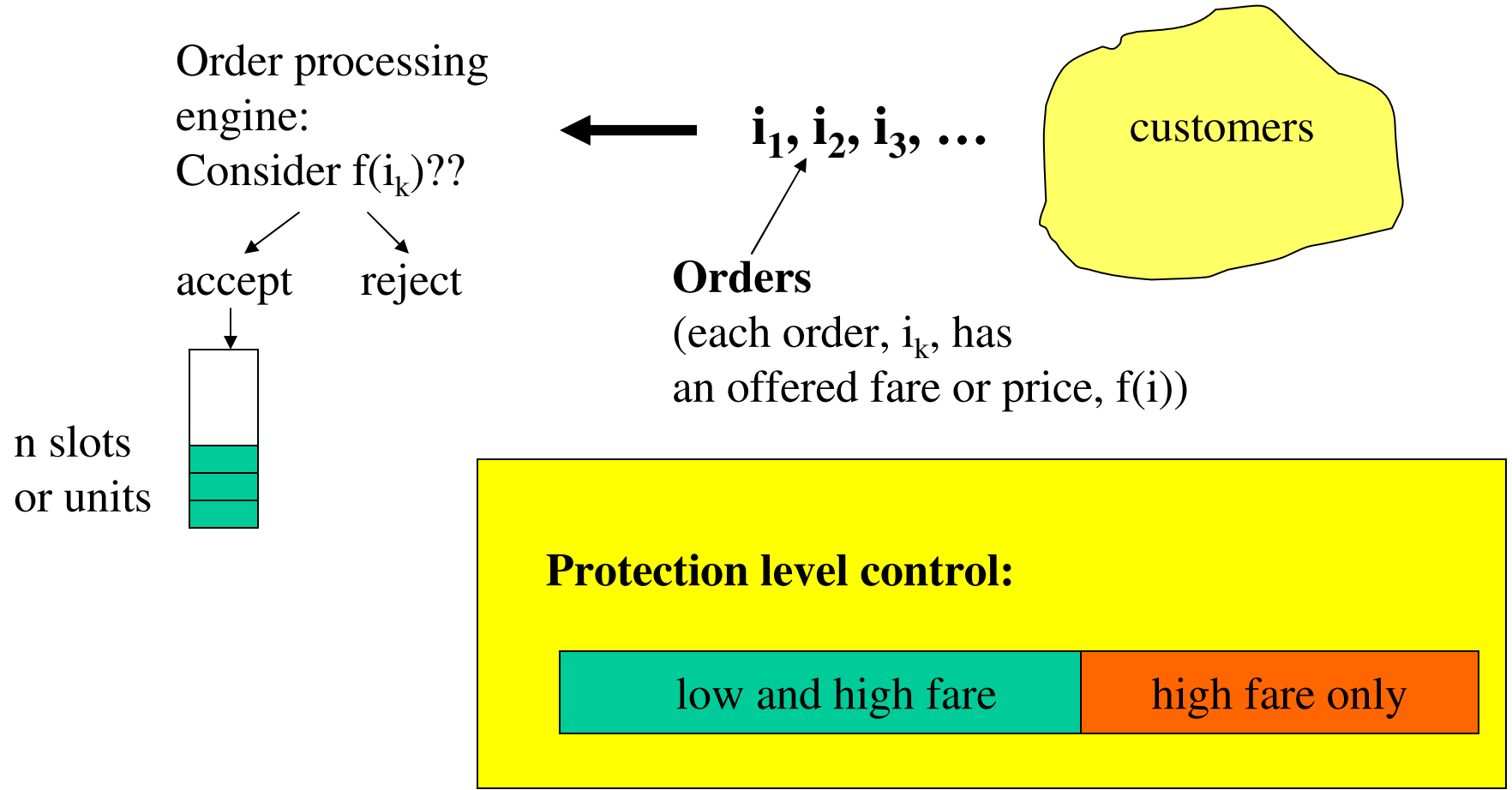
*Maurice Queyranne*

*Sauder School of Business*

*University of British Columbia*

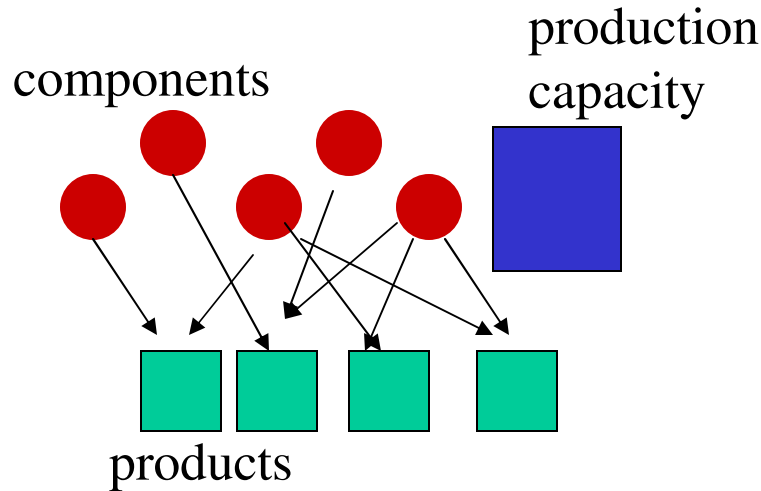


# Revenue Management Problem





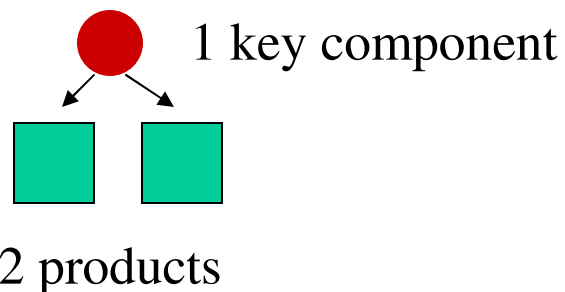
# Motivation: Assemble-to-Order Products



## *Some particular features:*

- Multiple capacity constraints
- Time dimension – production capacity and components replenish over time
- Time dimension – product differentiation based on delivery time
- No low before high

Very simple case:





# Competitive Analysis



Algorithm  
Designer



*algorithm*



$i_1, i_2, i_3, \dots$

*input  
stream*



Evil  
Adversary

**Competitive Ratio =**

$$\text{Min}_{\text{input streams}} \{(\text{alg performance})/(\text{best performance})\}$$

**“Traditional” revenue management analysis has assumed:**

- Demand can be forecast reasonably well
- Risk neutrality

**Are these valid??**



## Sample Result



- Flight has 95 available seats, three fare classes: \$1,000, \$750, \$500
- Policy that guarantees at least 63% of the max possible revenue:
  - Protect 15 high fare seats
  - Protect 35 seats for two higher fare classes (i.e. sell at most 60 lowest fare seats)



# Two-Fare Analysis



- $n$  = number of seats
- $f_1$  = higher fare;  $f_2$  = lower fare.
- $r = f_2/f_1$  = discount ratio.
- Key quantity:  $b(r) = 1 / (2 - r)$

Policy intuition:

Always best to accept any high fare (H) that comes along

→ Adversary will start off with low fare requests (L): Question – how many to accept before only H's are accepted??



# Basic Trade Off



*Note: must accept first order, o.w. adversary will stop after submitting one order with algorithm performance = 0.*

LLLL|LLLLLLLLLLLLLLLL

Stop accepting L's

*Accept too few L's → adversary will only send additional L's*

available seats

LLLLLLLLLLLLLLLL|HHHHHHHH

Stop accepting L's

*Accept too many L's → adversary will only send additional H's*



# Best Two-Fare Policy



**Proposal: protect  $(1 - b(r)) n = (1 - 1/(2 - r)) n$   
... assume for the moment that  $b(r) n$  is integer ..**

LLLLLLLLLLLLLLLL|LLLLLLL

**All L's after stopping →**

$$\text{Performance} = (f_2 b(r) n) / (f_2 n) = b(r)$$

LLLLLLLLLLLLLLLL|HHHHH

**All H's after stopping →**

$$\text{Performance} = [f_2 b(r) n + f_1 (1 - b(r)) n] / (f_1 n) = b(r)$$



# What about the assumption that $b(r)$ was integer?

*Continuous booking problem* – can accept a fraction of an order

**Theorem:** *For the continuous two-fare booking problem, the booking policy with protection level  $(1 - b(r))n$  has competitive ratio  $b(r)$ . This is best possible among all online booking policies.*

*Will also treat case where each order must be fully accepted or rejected.*



# Examples



$r = 50\%$ ;  $n = 90$ :

- Protection level: 30; Comp ratio:  $2/3$ .

$r = 25\%$ ;  $n = 90$

- Protection level: 39; Comp ratio:  $4/7$

$r = 75\%$ ;  $n = 90$

- Protection level: 18; Comp ratio:  $4/5$



# Discrete Case



$$C(x) = \min \{ [n - (1 - r)x]/n, x/n \}$$

$$\theta(r,n) = \begin{cases} n - \lfloor b(r) n \rfloor, & \text{if } c(\lfloor b(r) n \rfloor) \geq c(\lceil b(r) n \rceil) \\ n - \lceil b(r) n \rceil, & \text{otherwise} \end{cases}$$

**Theorem:** *For the discrete two-fare booking problem, the booking policy with protection level  $\theta(r,n)$  has competitive ratio  $c(n - \theta(r,n))$ . This is best possible among all deterministic online booking policies.*



# Randomized Algorithms



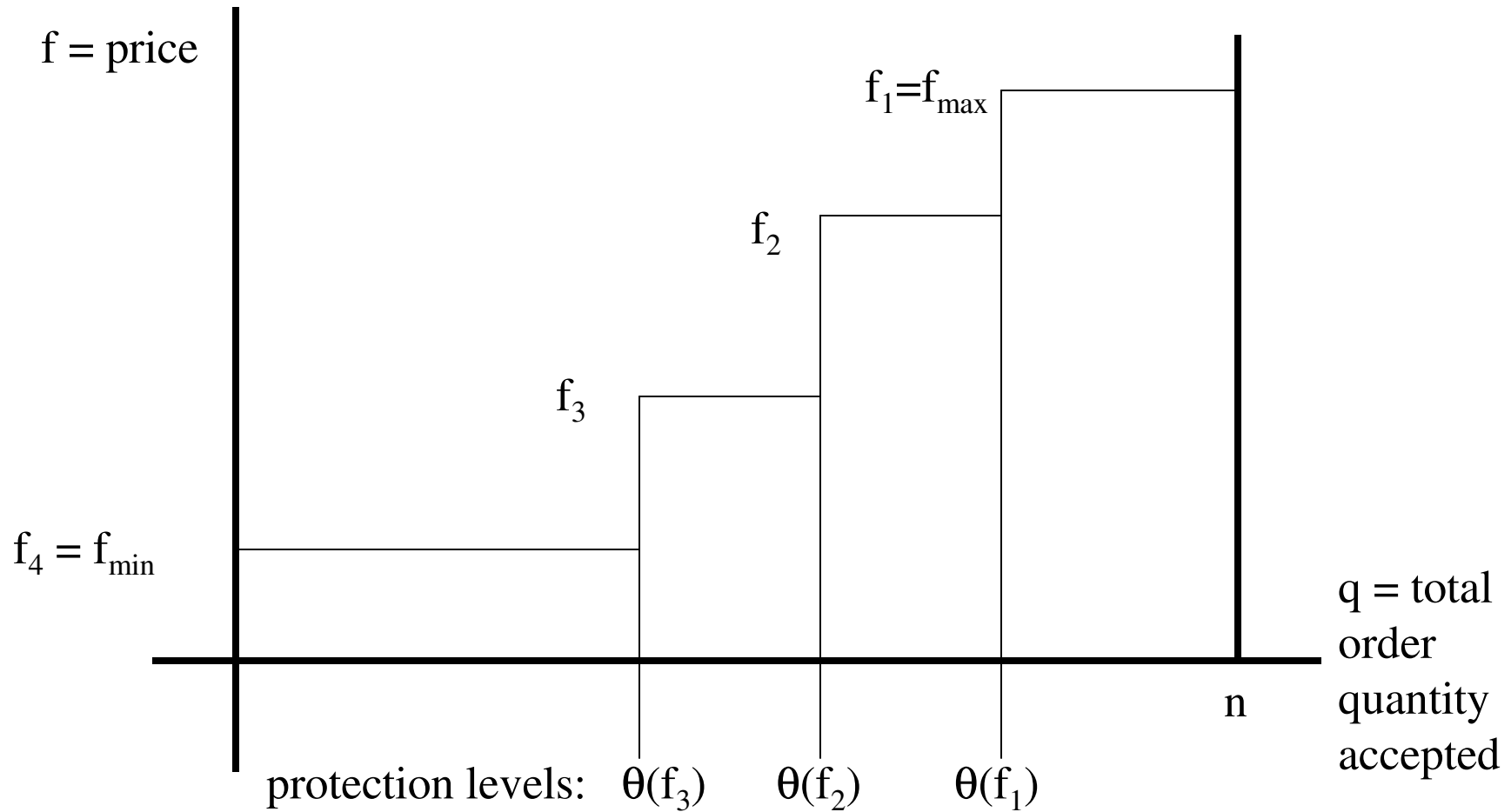
**Randomized algorithm can execute certain steps based on the outcome of a random experiment, e.g. flipping a coin – in some cases, randomized algorithms can achieve performance levels superior to the levels that can be achieved by deterministic algorithms.**

**Proposition:** *For the discrete two-fare booking problem, there exists a randomized booking policy with competitive ratio  $b(r)$ , and this is best possible.*

**Theorem:** *For the discrete two-fare booking problem, a randomized booking policy has optimal competitive ratio  $b(r)$ , if and only if the expected value of its protection level is  $(1 - b(r))n$ .*



# More General Cases





## m Fare Classes



**Define:**  $\Delta = m - \sum_{\{i=2,m\}} f_i / f_{i-1}$

**Theorem:** For the continuous m-fare problem, no booking policy, deterministic or random, has a competitive ratio larger than  $1 / \Delta$ .

**Define:**  $\theta_i = (n / \Delta) (i - \sum_{\{j=1,i\}} f_{j+1} / f_j)$

**Theorem:** For the continuous m-fare problem, the protection level policy using protection levels  $\theta_i$  achieves a competitive ratio of at least  $1 / \Delta$ .

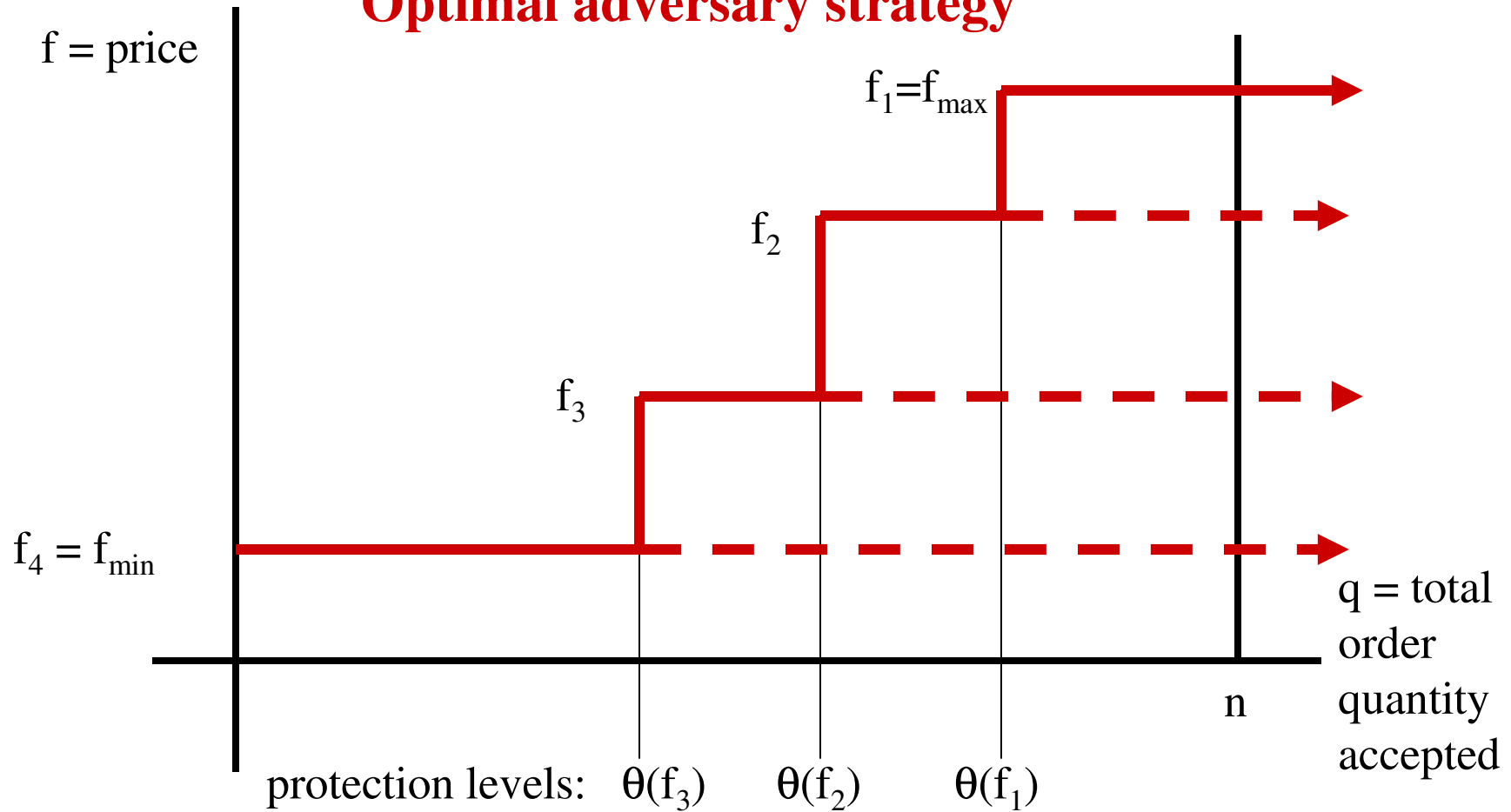
e.g.  $f_1 = 1000, f_2 = 750, f_3 = 500, 1 / \Delta \approx 63 \%$



# Approach to alternate type of policy

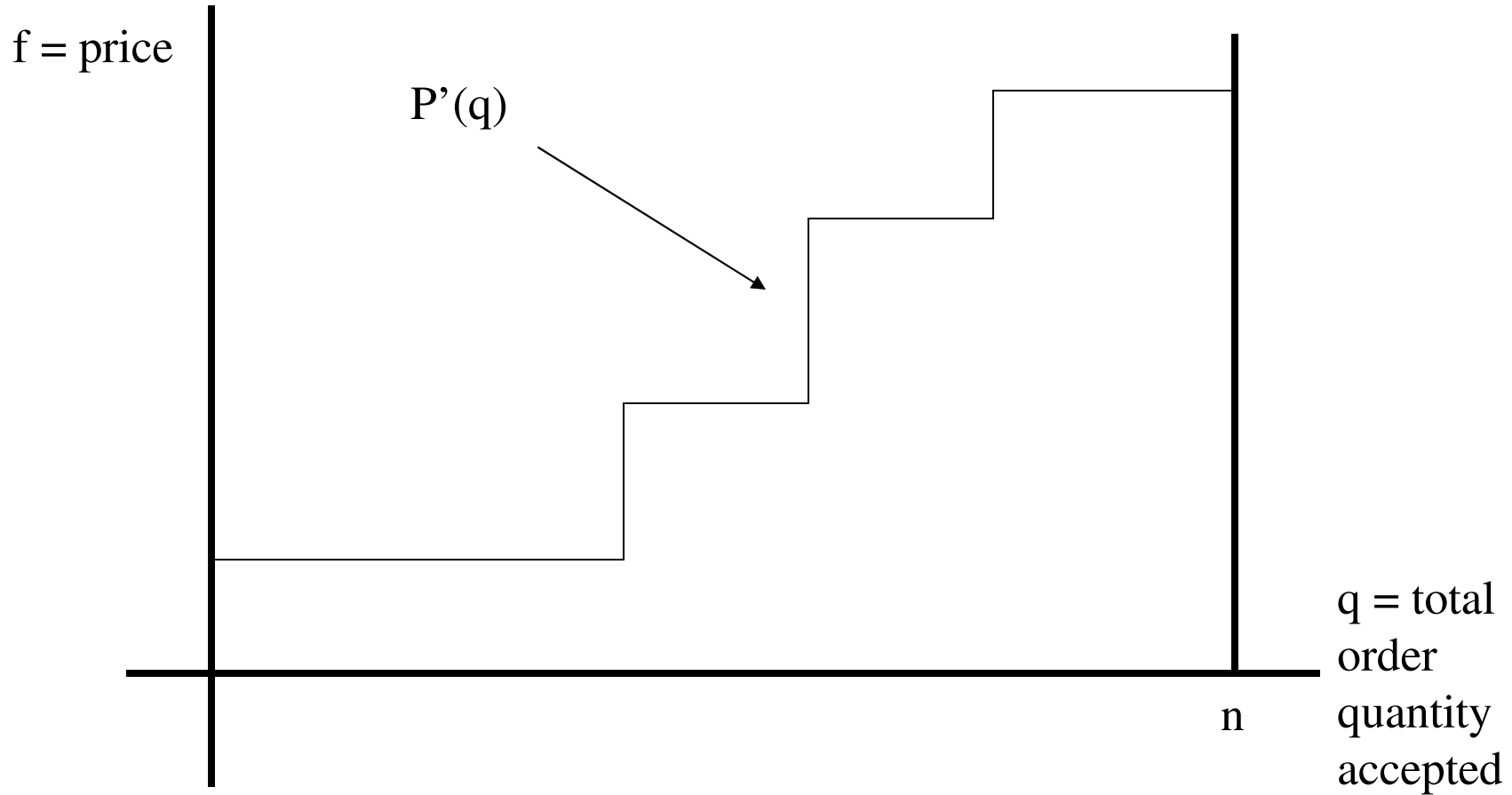


## Optimal adversary strategy





# Order Quantity Control Fcn



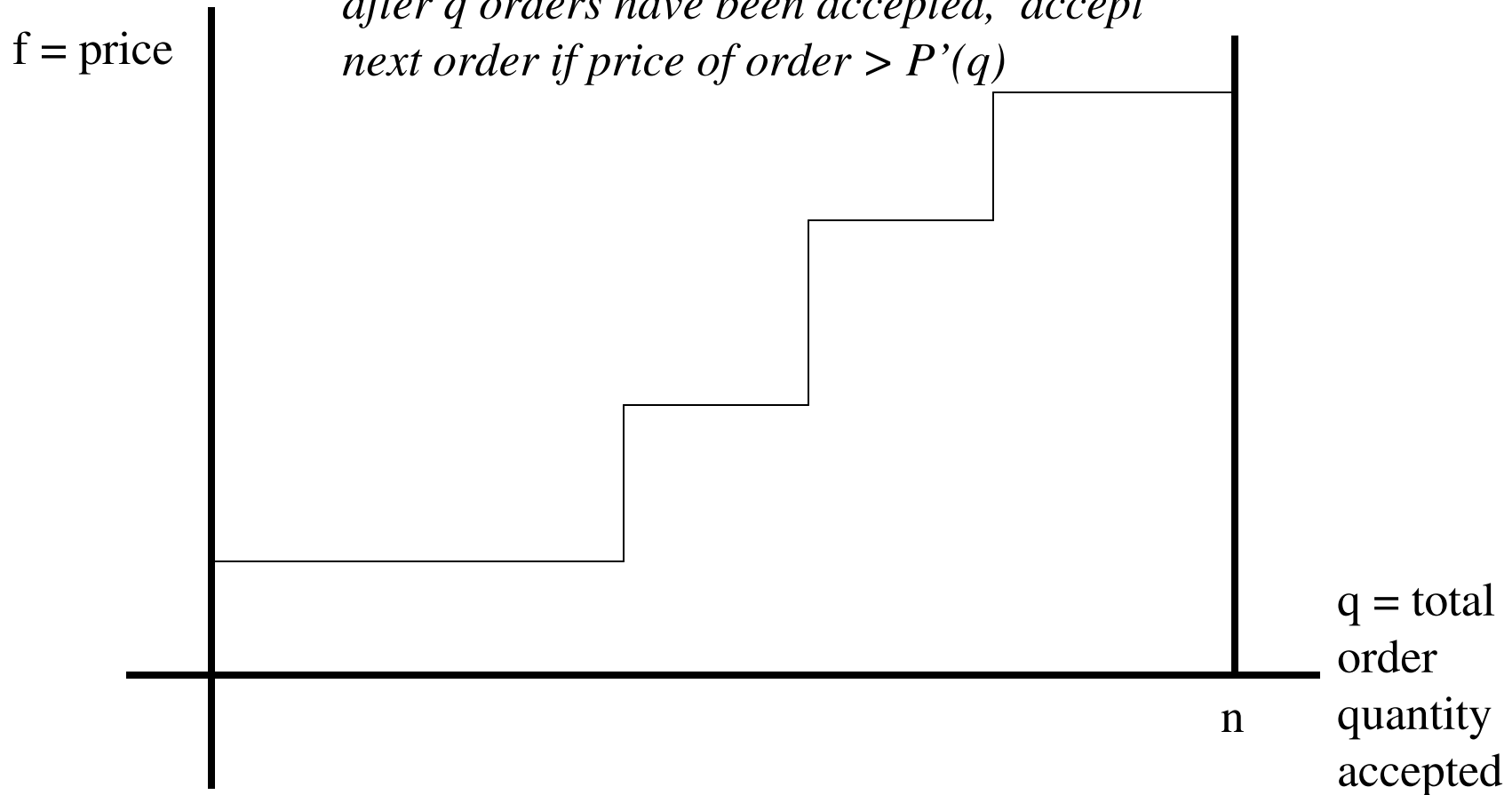


# Order Quantity Control Policy aka "Theft Nesting"



**Order quantity control policy:**

*after  $q$  orders have been accepted, accept  
next order if price of order  $> P'(q)$*





# Order Quantity Control Policy



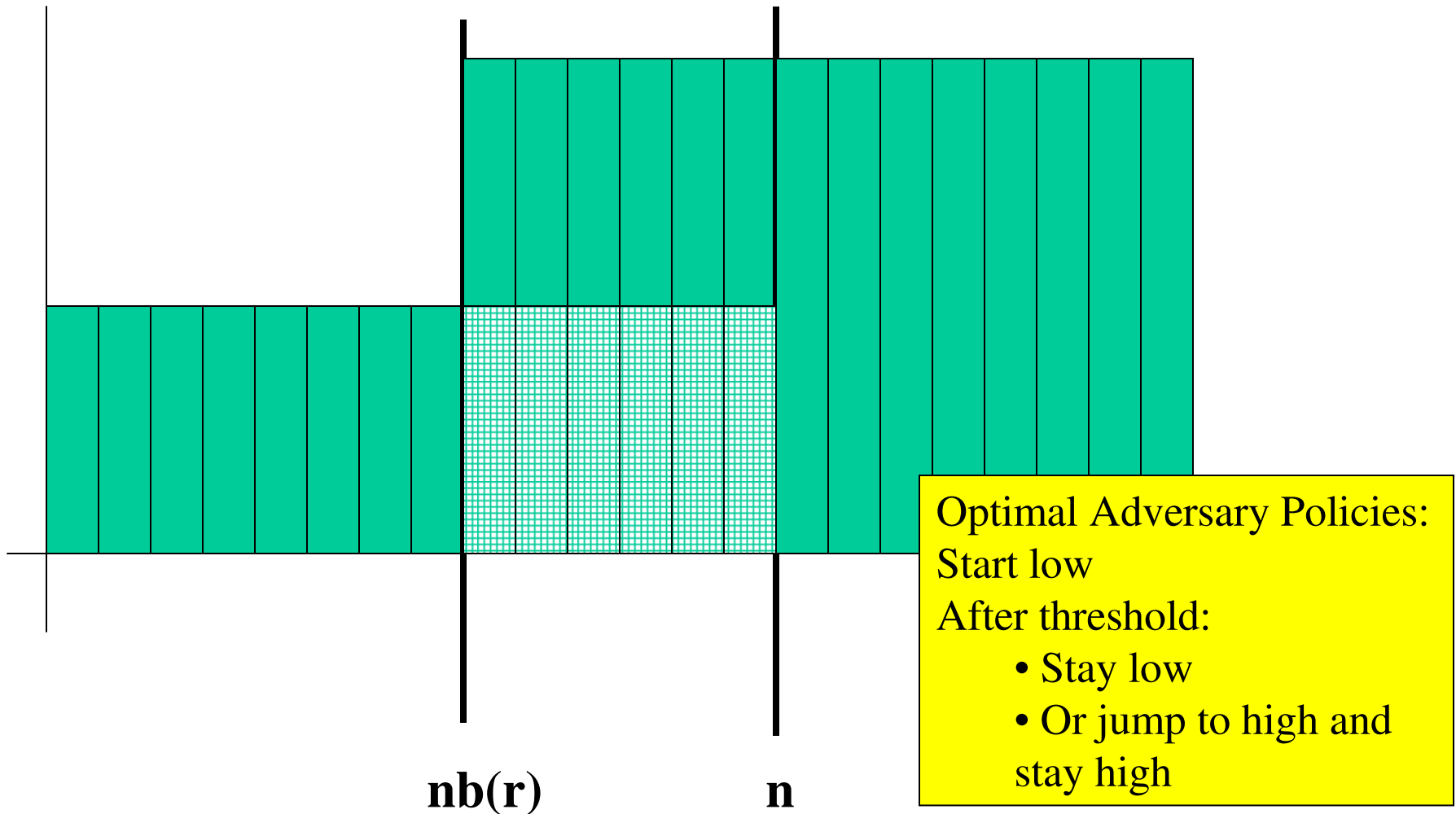
**Order quantity control policy:**

*after  $q$  orders have been accepted, accept next order if price of order  $> P'(q)$*



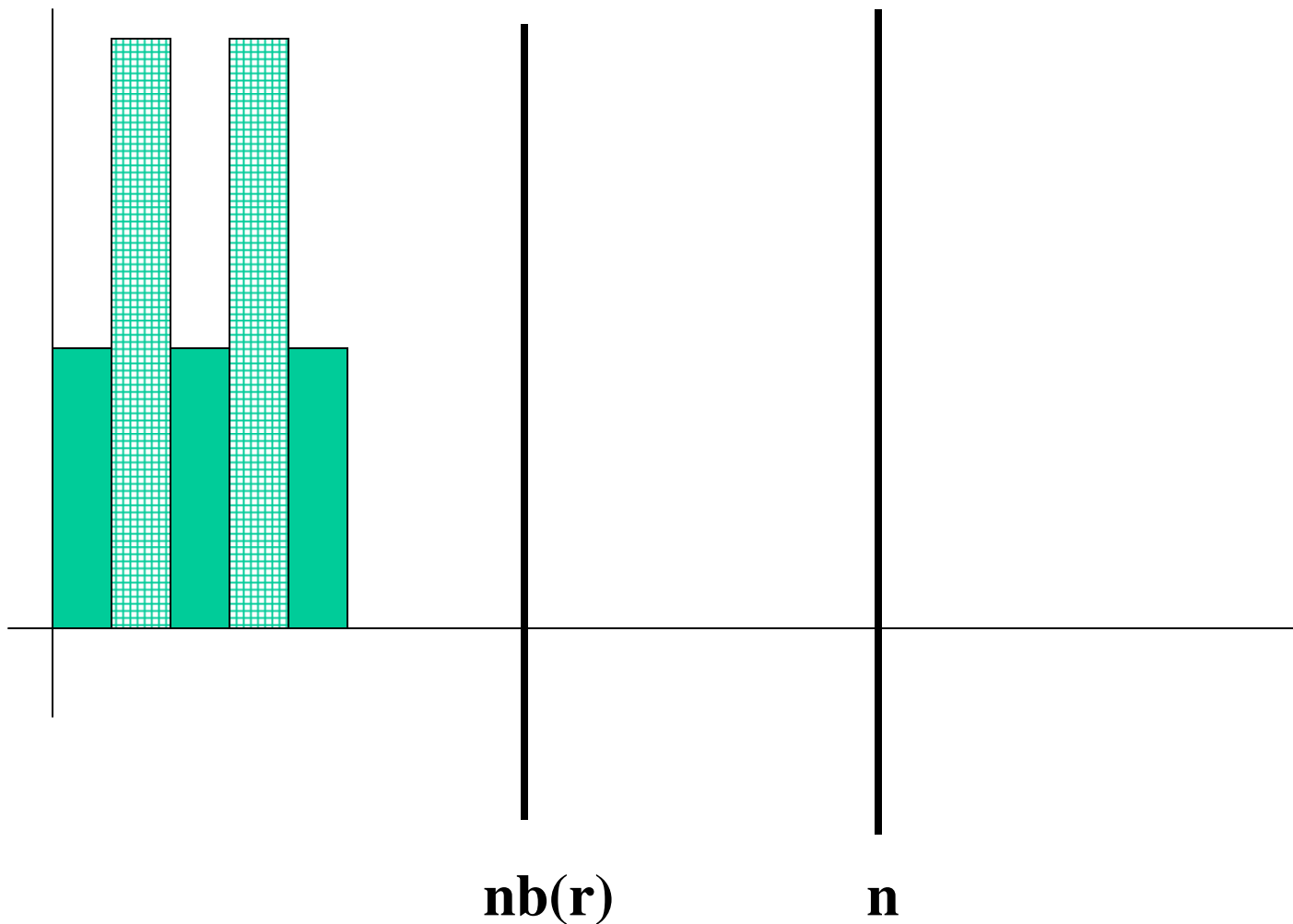


# Dynamic Policy Motivation: Taking advantage of an inferior adversary



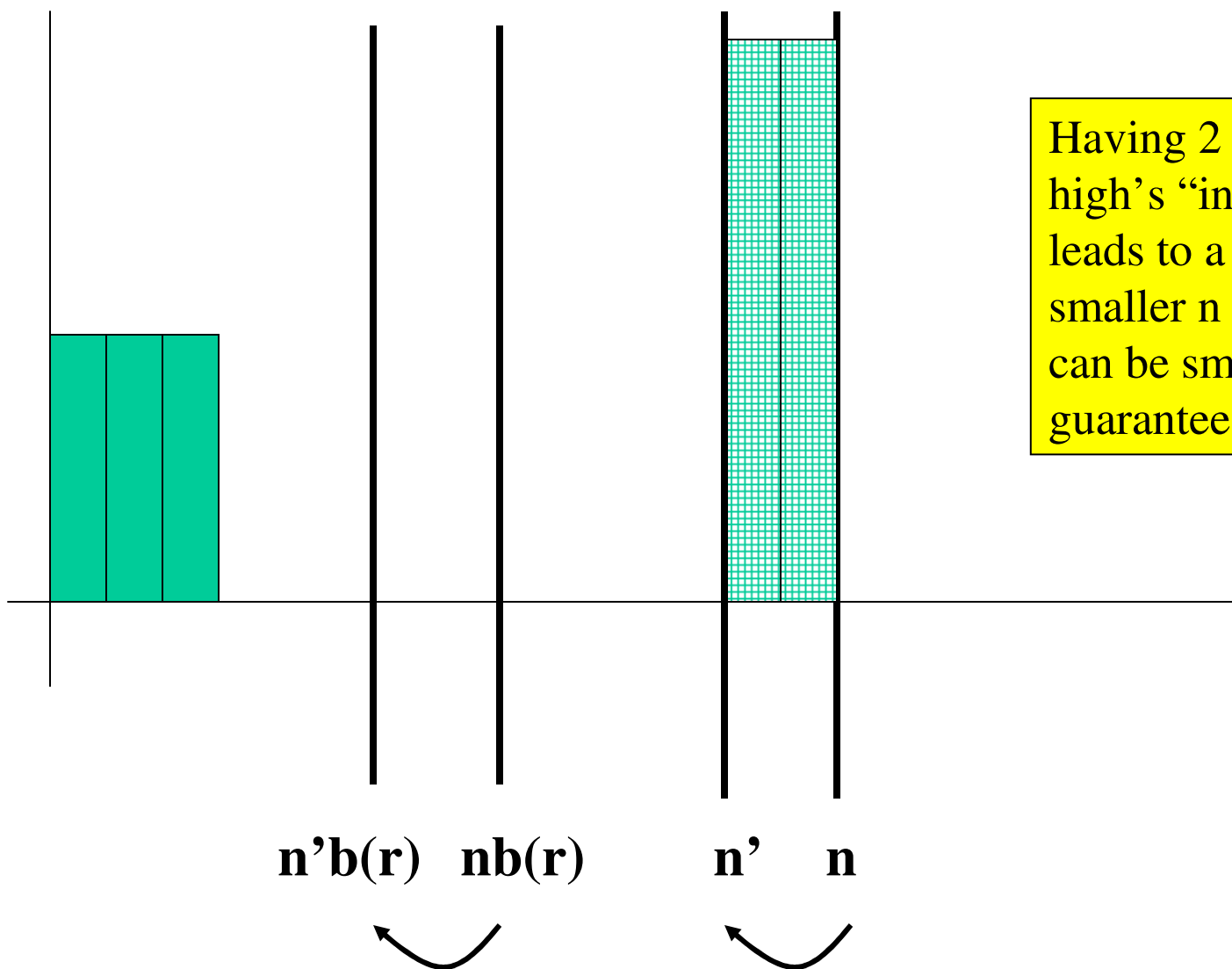


# Dynamic Policy Motivation: Taking advantage of an inferior adversary





# Dynamic Policy Motivation: Taking advantage of an inferior adversary



Having 2 (or more) high's "in the bank" leads to a problem on a smaller  $n \rightarrow$  threshold can be smaller & overall guarantee is improved.



# Dynamically Revising Threshold



orig threshold:  $n b(r) = n / (2-r)$

let:  $h' = \#$  high fare request accepted so far

$$\gamma = h'/n \quad \alpha = (\gamma f_1 + (1 - \gamma) f_2) / f_2$$

revised threshold:

$$n (1 + (1 - r) \gamma/r) / (1 + \alpha (1 - r))$$



# Dynamic Policy: numerical test

$n = 100$ ;  $r = .5$

| $h'$ | $l''$ | dynamic guarantee | $l'$  | static guarantee |
|------|-------|-------------------|-------|------------------|
| 1    | 65.78 | .67               | 66.67 | .67              |
| 5    | 62.30 | .69               | 66.67 | .67              |
| 10   | 58.60 | .71               | 66.67 | .67              |
| 20   | 50.00 | .75               | 66.67 | .67              |
| 30   | 42.42 | .79               | 66.67 | .67              |
| 40   | 35.29 | .82               | 66.67 | .67              |
| 50   | 28.57 | .86               | 66.67 | .67              |



# Numerical Experiments



- Order sequencing: Low-before-high (L-b-H) vs Random (uniformly distributed offer prices).
- Alternatives considered:
  - first-come-first-served
  - off-line optimal
  - **Robust -- static**
  - **Robust – dynamic**
  - EMSR – optimal strategy for L-b-H if demand (Poisson order arrivals) distribution parameters are known.
  - McGill-Van Ryzin: for L-b-H case with unknown parameters -- employs stochastic optimization methods.
  - Markov Decision Process (MDP) – optimal strategy for random when parameters are known
  - MDP with parameter forecasting – for random case where parameters are unknown.



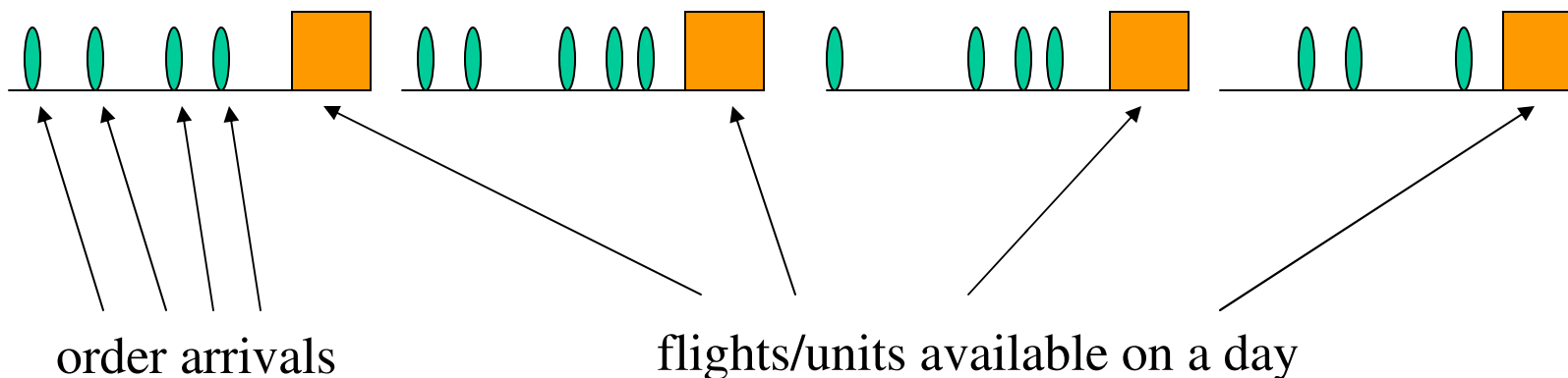
# Order Arrival Process



- **Stationary case:** exponential order inter-arrival times.
- **Key parameters:**  $L'$  = expected number of low orders;  $H'$  = expected number of high orders; number of seats/units = 100; interesting problems have  $L' + H'$  close to 100, e.g. 120.  $f_1 = 2000$ ,  $f_2 = 1000$ ; also 1500 & 1000.
- **Non-stationary case:**  $(H', L')$  jumps among set of possible values, e.g. (60,60), (20,100), (100,20)

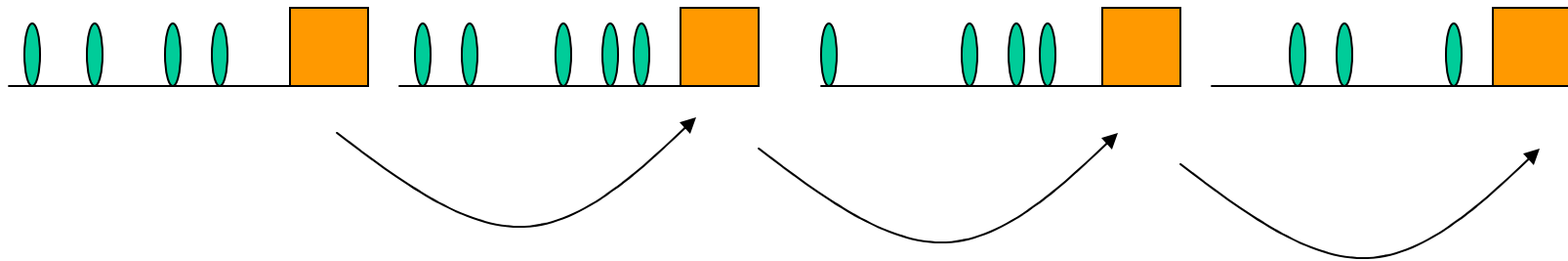


# Non-Stationarity and Parameter Update Process





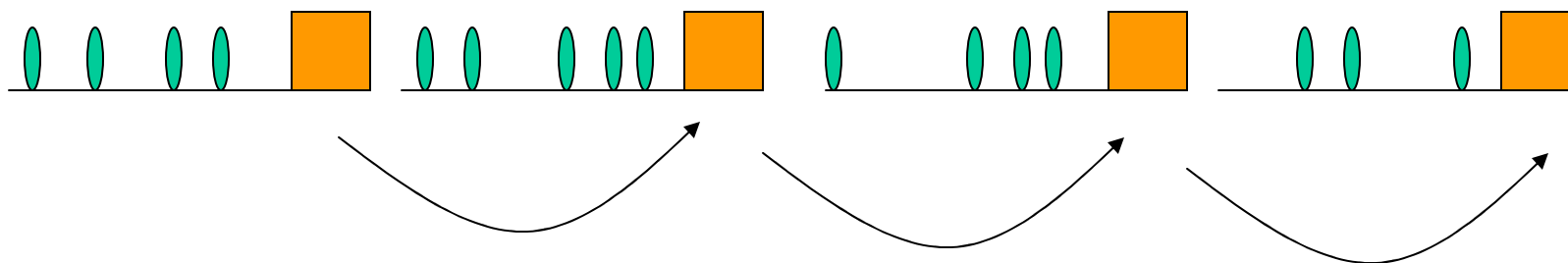
# Non-Stationarity and Parameter Update Process



## McGill-Van Ryzin Parameter Update Process



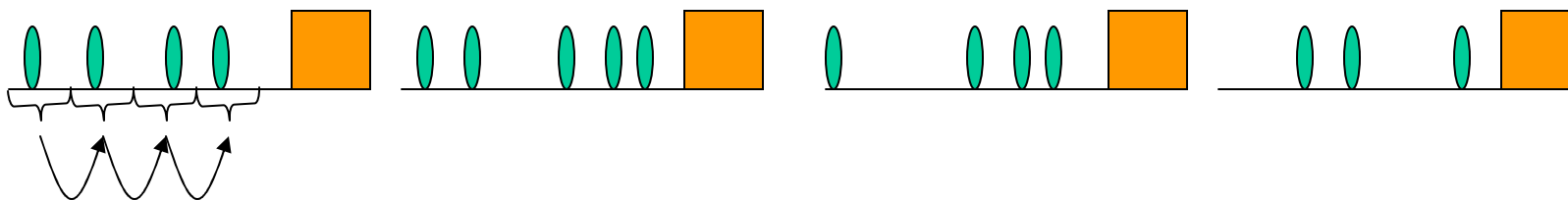
# Non-Stationarity and Parameter Update Process



**Flight-to-flight parameter jump process**



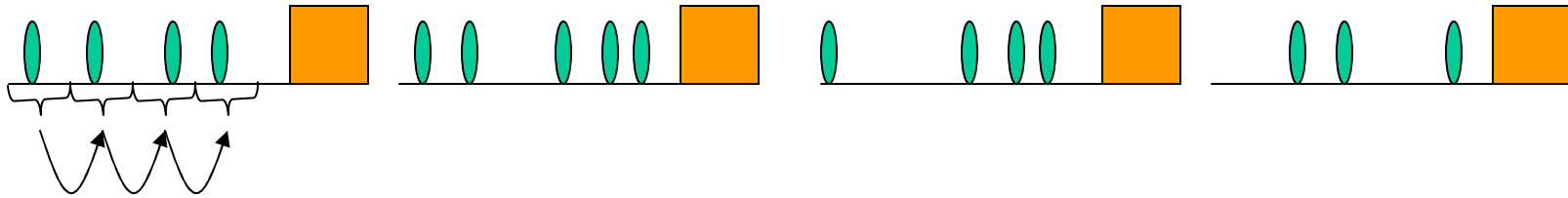
# Non-Stationarity and Parameter Update Process



## Dynamic MDP Parameter Estimation/Update Process



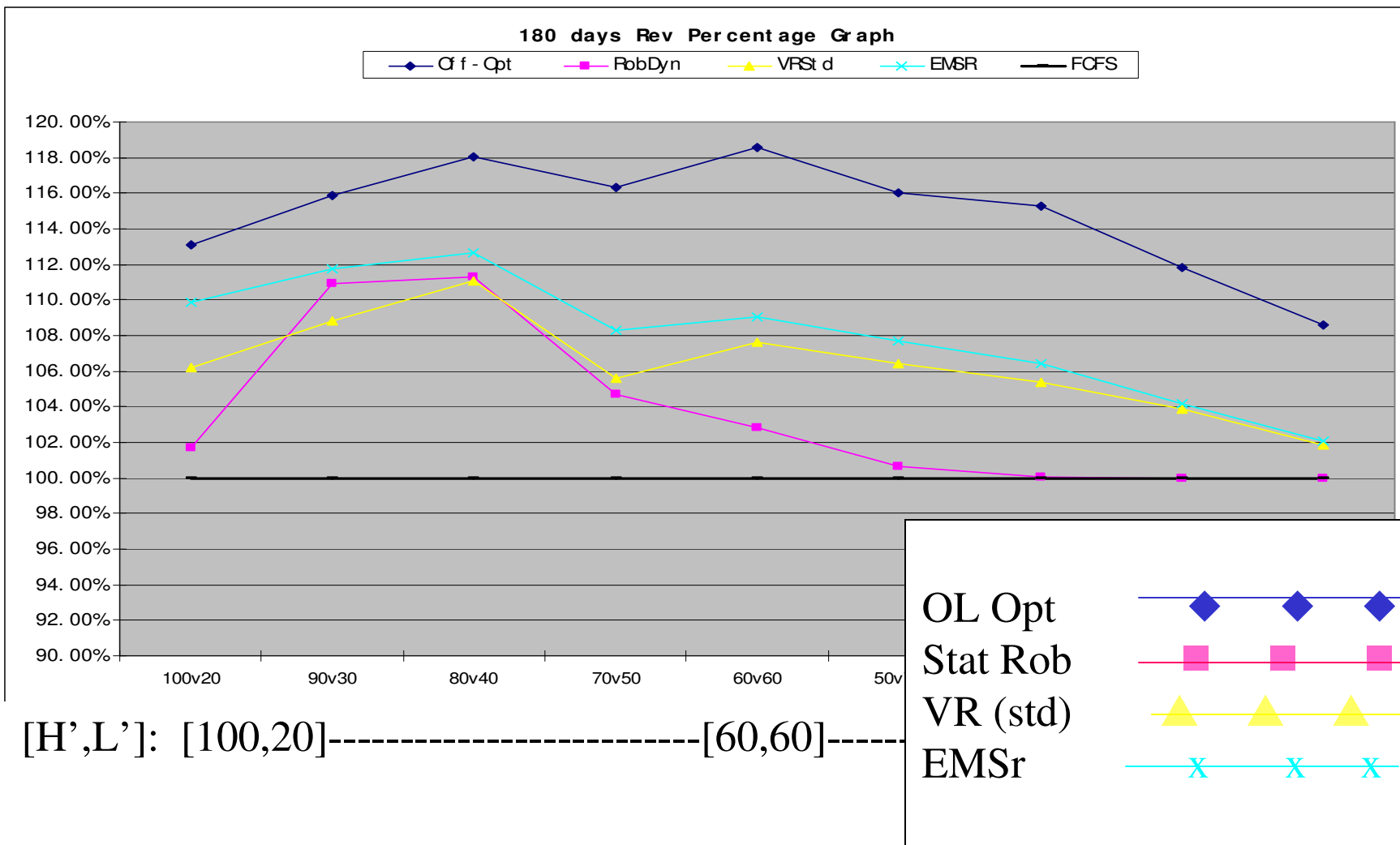
# Non-Stationarity and Parameter Update Process



**Within flight order jump process**

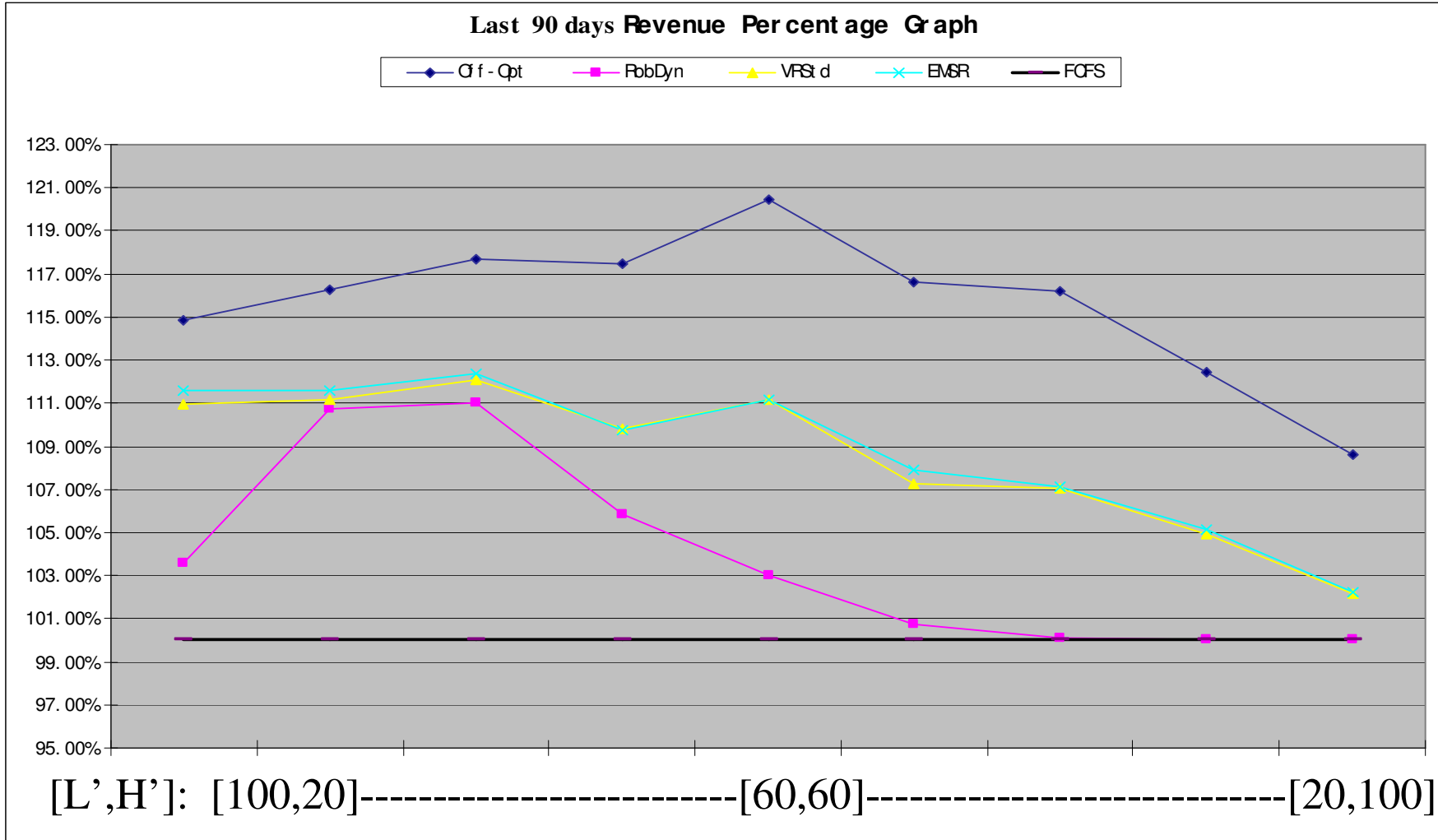


# L-b-H Stationary Demand (entire test period)



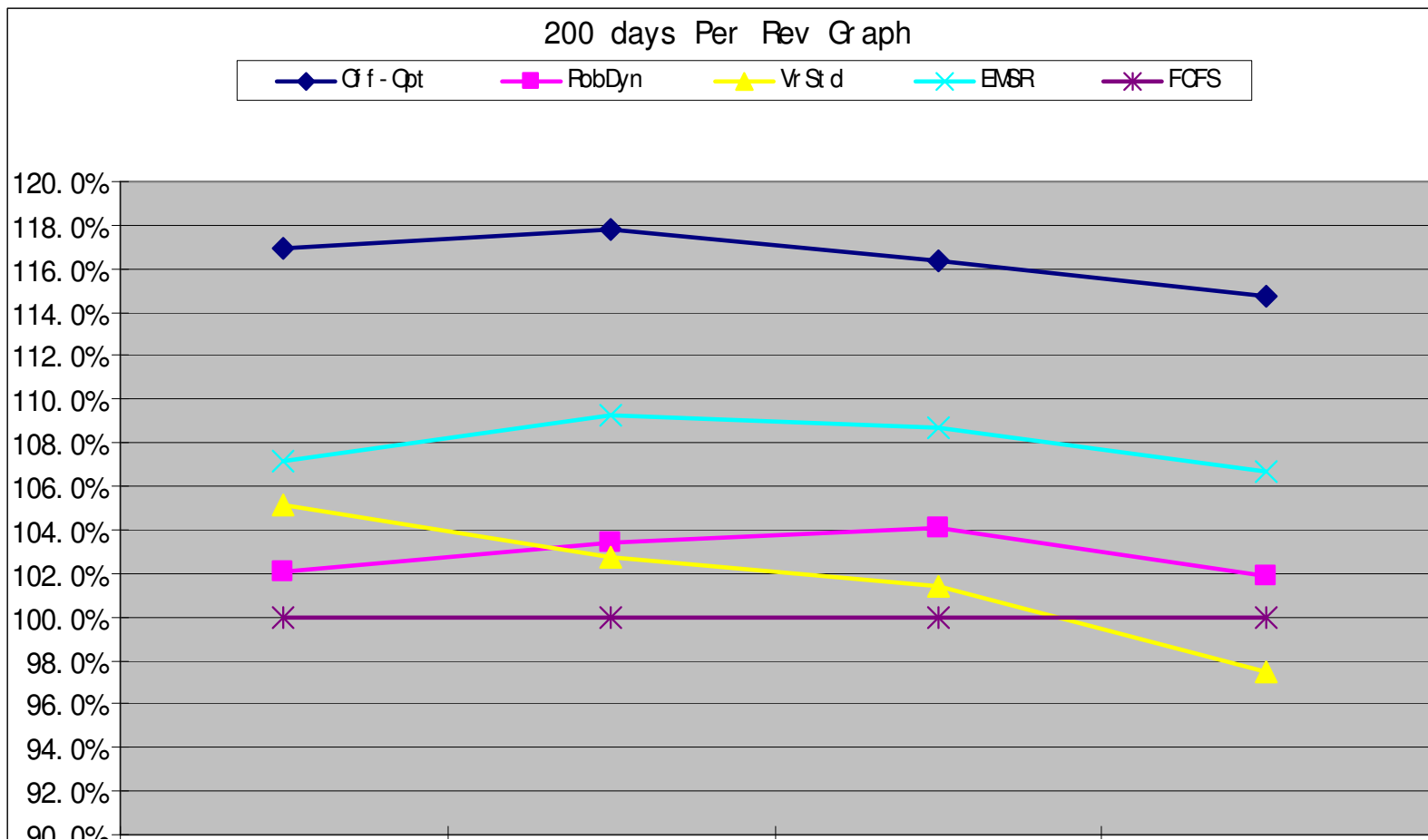


# L-b-H Stationary Demand (last 90 out of 180 day)





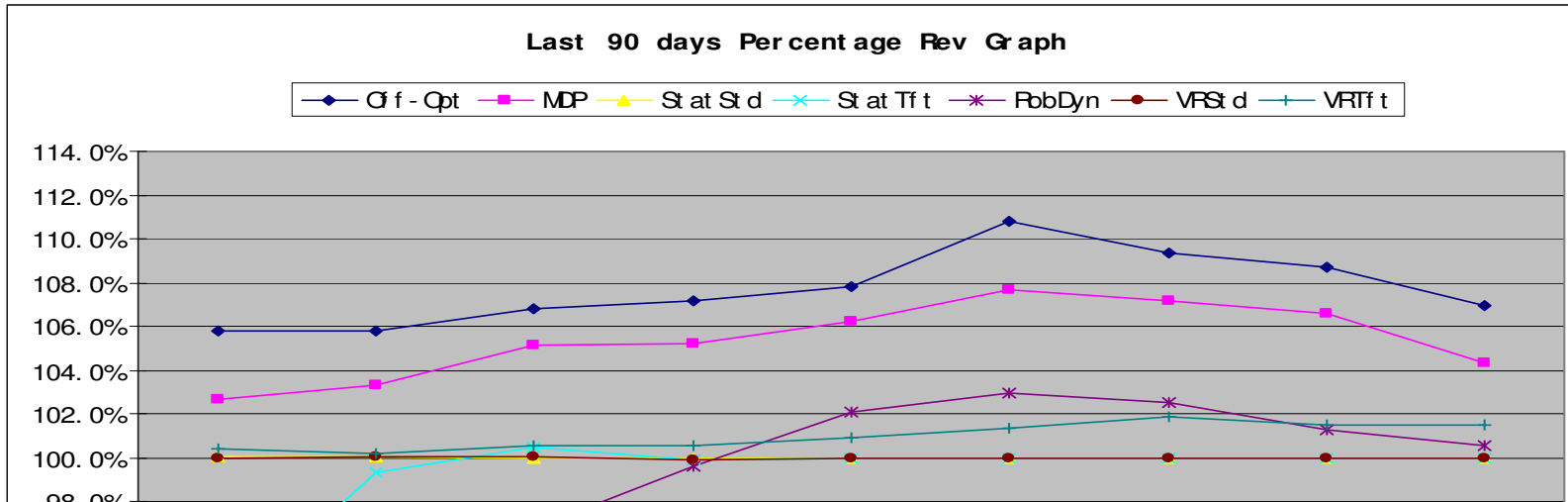
# L-b-H -- Non-stationary demand



[(60,60),(50,70),(70,50)] [(60,60),(40,80),(80,40)] [(60,60),(30,90),(90,30)] [(60,60),(20,100),(100,20)]



# Random orders – stationary demand – last 90 days

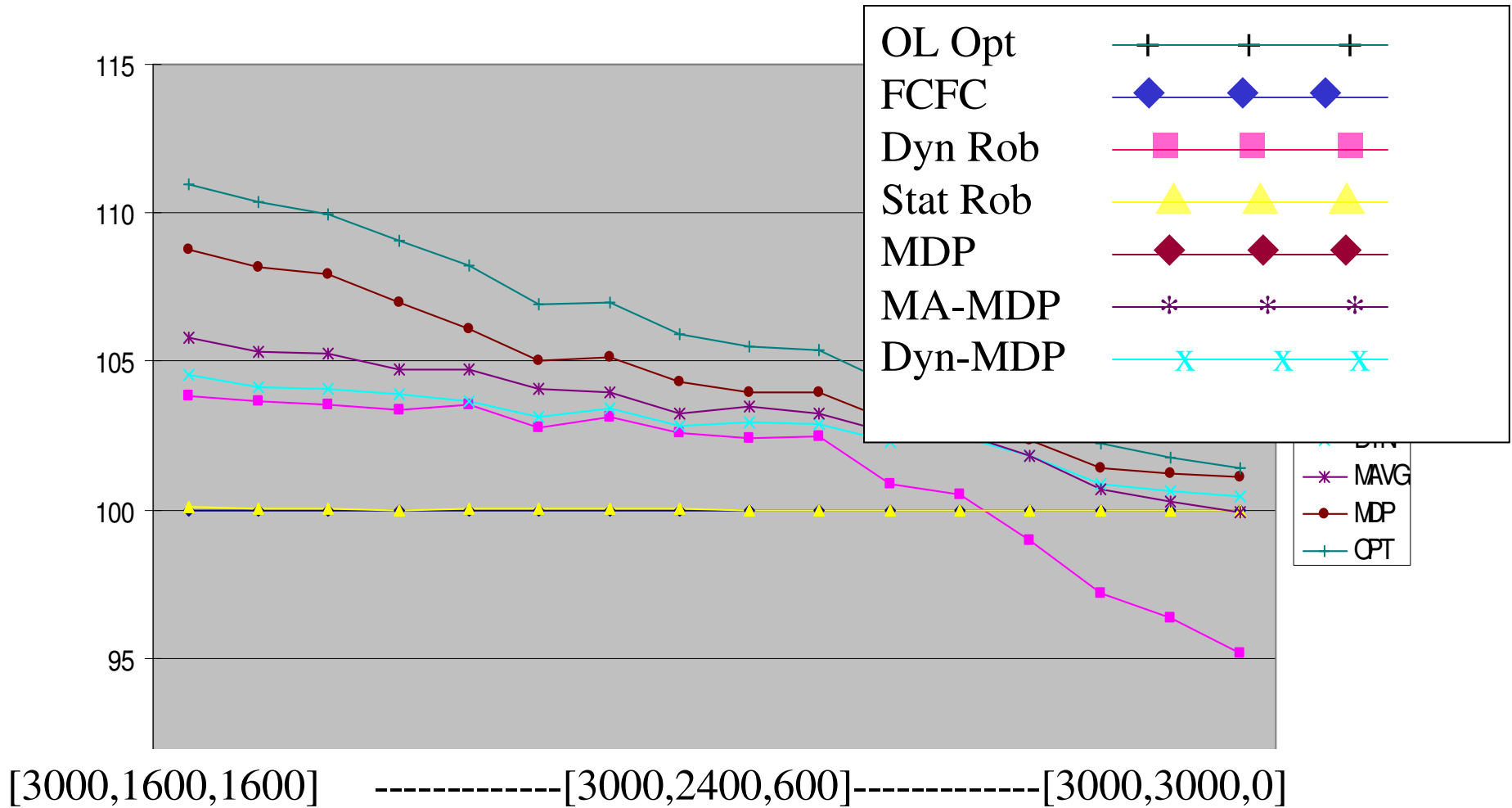


[L',H']: [100,20]-----[60,60]-----

|                |       |
|----------------|-------|
| M-VR (tft)     | + + + |
| OL Opt         | ◆ ◆ ◆ |
| MDP            | ■ ■ ■ |
| Stat Rob (std) | ▲ ▲ ▲ |
| M-VR std       | ◆ ◆ ◆ |
| Dyn Rob        | * * * |
| Stat Rob (tft) | x x x |

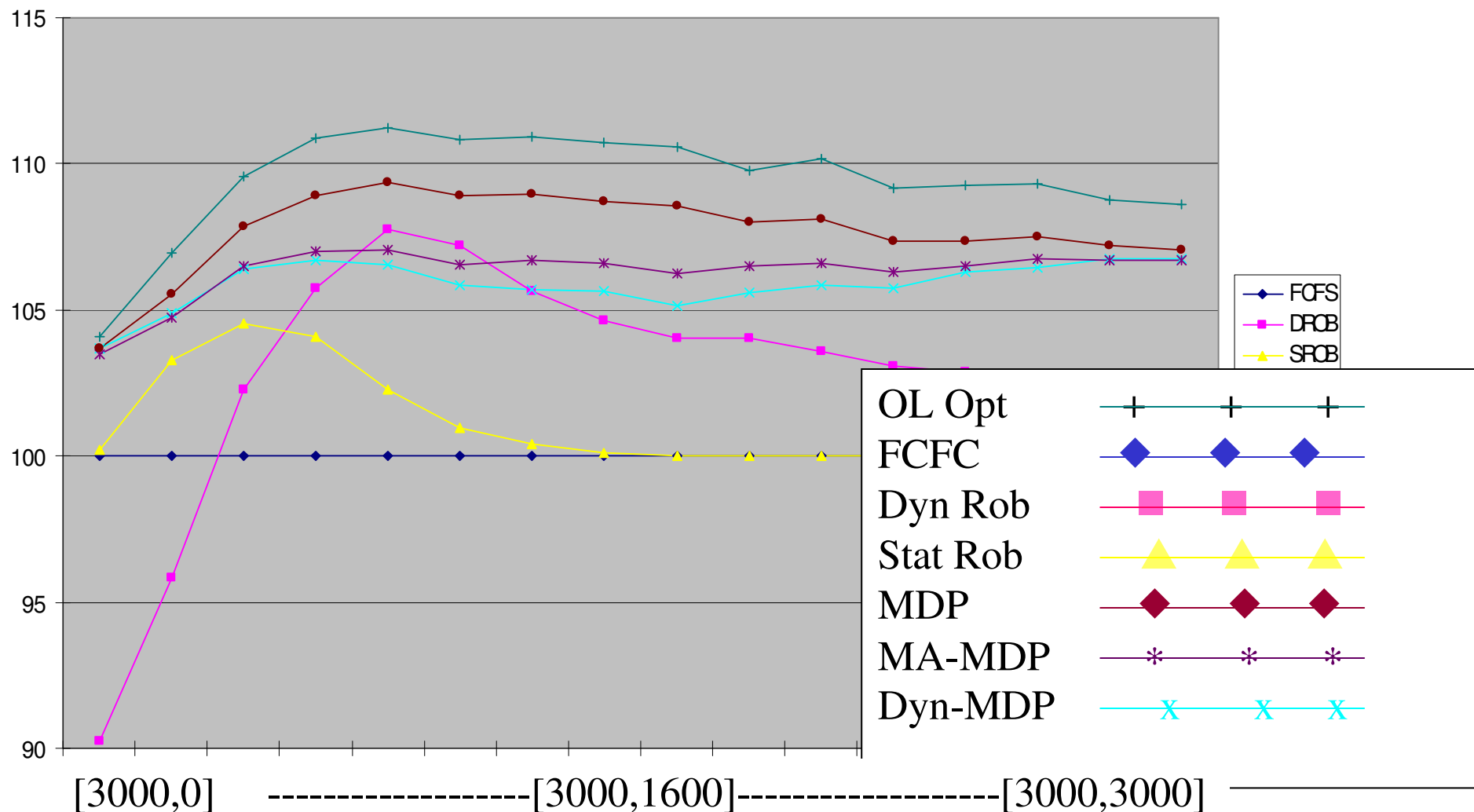


# Non-Stationary/Random: (L',H'): (25,95) → (50,50) → (95,25)





# Non-Stationary/Random: (H',L'): (30,90) → (90,30)





# Final Thoughts



- Many implicit and explicit assumptions in airline revenue management problems – relaxing these can lead to novel problems.
- Online algorithm approach shows significant promise:
  - No risk neutrality assumption.
  - Demand information not required.
  - Policies are practical; work well across range of demand distributions; worst case at extremes, esp very few highs.