FEOR – Module 6 – Optimization

Optimization under uncertainty

Agnès Gorge – Rachid Ellaia

Nouvelle proposition

FEOR – Module 6 – Optimization for supply-chain

Jour 1 – Vendredi 21/01		
9h - 11h	Combinatorial optimization	*Rachid Ellaia*
11h15 - 12h45	TP1 : MILP with PuLP (knapsack problem)	*Agnès Gorge*
14h – 15h30	Dynamic Programming	*Rachid Ellaia*
15h45 – 16h45	TP2 : Dynamic Programming (knapsack problem)	*Agnès Gorge*
17h – 18h	Introduction to scheduling and project presentation (Mine Scheduling)	*Agnès Gorge*

Jour 2 – Samedi 22/01		
9h - 11h	Meta-heuristics	*Rachid Ellaia*
11h15 - 12h45	TP3 : Knapsack problem with meta-heuristics	*Agnès Gorge*

Later on	
30/10 18h	Quizz
02/02 14h-17h	Project defense

Jour 3 – Samedi 29/01		
9h – 10h30	Multi-objective Optimization	*Rachid Ellaia*
10h45 – 12h15	TP4 : Multi-objective	*Rachid Ellaia*
13h15 - 14h15	Stochastic optimization	*Agnès Gorge*
14h15 - 15h45	TP5 : Newsvendor problem with LP	*Agnès Gorge*
16h00 - 17h30	TP6 : Dynamic (stochastic) programming for Inventory Management	*Agnès Gorge*

Newsvendor

TP5 : The newsvendor problem through LP

Problem presentation

Decision variable
- Purchase
- Sales[scenarios] : how much to sell
- Liq[scenarios] : how much to liquidate
- Revenue[scenarios] : the revenue of sales and liquidation

Uncertain data : demand following a discrete distribution (scenarios)
Known datas :
- SalesPrice
- LiqPrice
- UnitCost

Resolution through LP

1. **Formulate the problem as a linear program, for maximization of expected profit**
2. **What about considering the minimization of the variance in the objective function ?**
3. **Consider chance-constraint : Proba[sold out] <= 1%**
4. **Bonus : add a constraint over the Value at Risk (VaR) or Conditional Value at Risk (CVaR)**

Reminder : Chance-constraint, VaR, CVaR

Chance-constraint (or probability constraint)
$P [f(x) \leq a] \geq 1-\varepsilon$

Define a binary variable per scenario to « flag » whether or not the sold-out happen in this scenario. Let X[s] be this variable.

Define X[s] such that « sold-out on scenario s implies X[s] = 1 »

Then limit the number of X[s] = 1

Reminder : BigM tips

- X a continuous variable
- B a binary variable

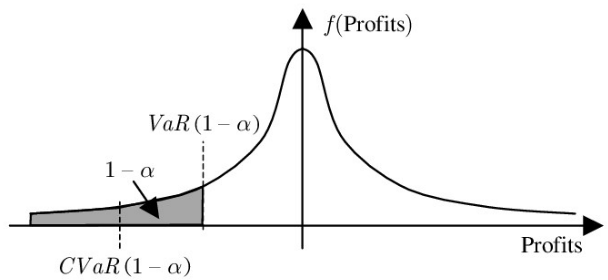
We want : X > 0 implies B = 1

Add a « bigM » constraint :

X <= M. B *with M big enough*

VaR and CVaR
Let X be the Profit & Loss variable

VaR	VaR = Value at Risk. VaR_p is the threshold such that : $P [X \leq VaR_p] \geq 1-p$ and $P [X \geq VaR_p] \leq p$
CVaR	CVaR : also called Expected Shortfall $CVaR_p$ = expected value of X strictly exceeding the VaR_p cutoff point



TP5 : The newsvendor problem

Problem presentation

Decision variable

- **x :** purchase

Uncertain data : demand to satisfy **D** of distribution F

Certain datas :

- SalesPrice p
- LiqPrice l (<p)
- UnitCost c

$$\text{Max} \quad E[p \min(\mathbf{x}, \mathbf{D}) + l \max(0, \mathbf{x} - \mathbf{D})] - c\,\mathbf{x}$$

Analytical solution

Solution : $\mathbf{x^* = F^{-1}\left(\frac{p-c}{p}\right)}$

→ **One of the rare example where a closed formula can be found !!**

An example where : Expected (optimal) <> Optimal (expected)

Inventory management

Inventory management

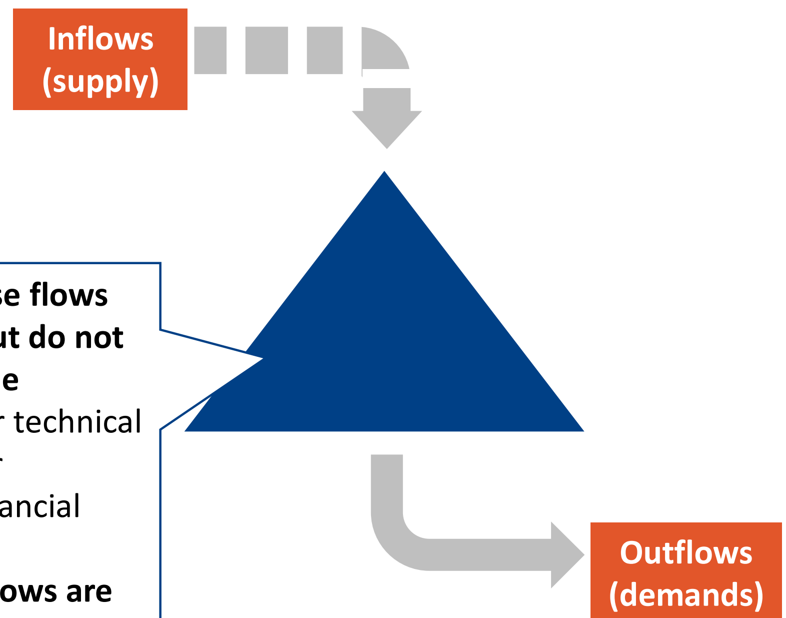
Loss of revenue

Customer’s insatisfaction
Make the company look bad

What do we mean by stock & inventory ?

Formal definition	**Inventory (American English) or stock (British English) refers to the goods and materials that a business holds available for future sale, production or utilization**
Key dynamics	**StockLevel[t] = StockLevel[t-1] + Inflow[t] – Outflow[t]**
Extended vision	• **Can be extended to services** : ex : a transportation capacity = a perishable stock of potential transportation • **Can be extended to a waiting line =** a stock of persons, vehicles, vessels

Stocks are fundamentally a way to decouple upstream and downstream flows



- **Either because flows are known but do not have the same dynamics** (for technical constraints or economic/financial choice)
- **Or because flows are uncertain**

Examples

	Known	Uncertain
Inflows	• Raw material ordering • Planned production	• Production breakdown • Stockout or delay at a supplier • Uncertain supply (agricultural product, rain, mininng, …)
Outflows	• Need of raw materials for production • Spare-parts for planned maintenance	• Uncertain demand

Inventory management is also a way to smooth predictable price fluctuations and to hedge against unpredictable price fluctuations

FlowIn - FlowOut (columns) / **Price - cost** (rows)

	Constant (= 0)	Time-varying but known	Uncertain
Constant	∅ No stock needed	**Tool stock / seasonal stock** *Ex : classical procurement*	**Safety stock** *Ex : newsvendor model*
Time-varying but known	**Price smoothing stock** *Ex : ???*	**Tool stock / seasonal stock + Price smoothing stock** *Ex : ???*	**Tool stock / seasonal stock + Safety stock** *Ex : ???*
Uncertain	**Price hedging stock** *Ex : commodities procurement*	**Tool stock / seasonal stock + Price hedging stock** *Ex : commodities procurement*	**Hedging stock** *Ex : trading commodities*

Uncertainty on production, demand, or transportation delays

Does not practically exist

Financial risk

Physical risk

Tool stock
=adaptation to supply dynamics

Seasonal stock
=adaptation to demand dynamics

No optimisation

Deterministic optimisation

Optimization under uncertainty

Optimization under uncertainty

Inventory management is also a way to smooth predictable price fluctuations and to hedge against unpredictable price fluctuations

FlowIn - FlowOut

Price - cost	**Constant (= 0)**	**Time-varying but known**	**Uncertain**
Constant	∅ No stock needed	**Tool stock / seasonal stock** *Ex : classical procurement*	**Safety stock** *Ex : newsvendor model*
Uncertain	**Price hedging stock** *Ex : commodities procurement*	**Tool stock / seasonal stock + Price hedging stock** *Ex : commodities procurement*	**Hedging stock** *Ex : trading commodities*

Uncertainty on production, demand, or transportation delays

Financial risk

Physical risk

Tool stock
=adaptation to supply dynamics

Seasonal stock
=adaptation to demand dynamics

No optimisation

Deterministic optimisation

Optimization under uncertainty

Optimization under uncertainty

Inventory management aims at finding the best trade-off between costs and value-creation

Ordering cost

- Order placing
- Transportation
- Quality control

Carrying costs

- Storage + handling
- Deterioration & obsolescence
- Cost of capital

Avoid stock-outs

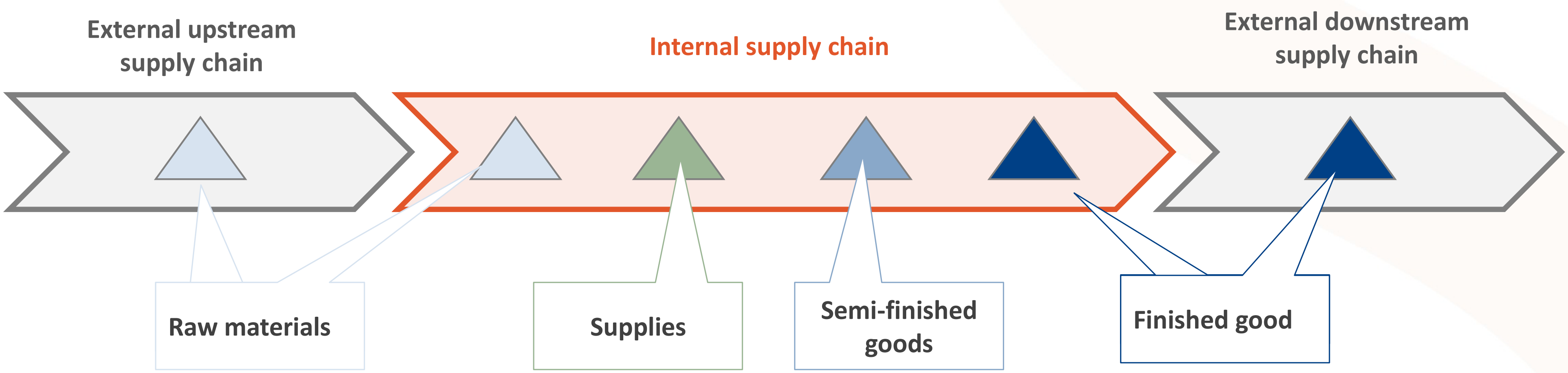
- Meet variations in customer demand
- Ensure uninterrupted production
- Reliability image

Perform financially

- Hedge against price increases
- Strengthen your negociating position

Costs | **Value creation**

Different types de stocks all along the value chain



Different paradigms of inventory management

Inventory management

Deterministic

Uncertain

	Decision process	Outcomes
Static	I take a decision, or a set of decision, right now And I will start again tomorrow (rolling process)	A scalar or a vector d
Dynamic	I wait to make a part of my decisions in order to have additional information. But I decide right now on the rule (or policy) to make them	A function of uncertain parameters ξ and of the state of my system s : d = f(ξ , s)

Decisions to make related to inventory management

Strategic → **Tactical** → **Operational**

Inventory design

New warehouses (sizing, location, …)

Strategic inventories

Management rules

What the rules or policy to apply to each inventory ?

Implementations

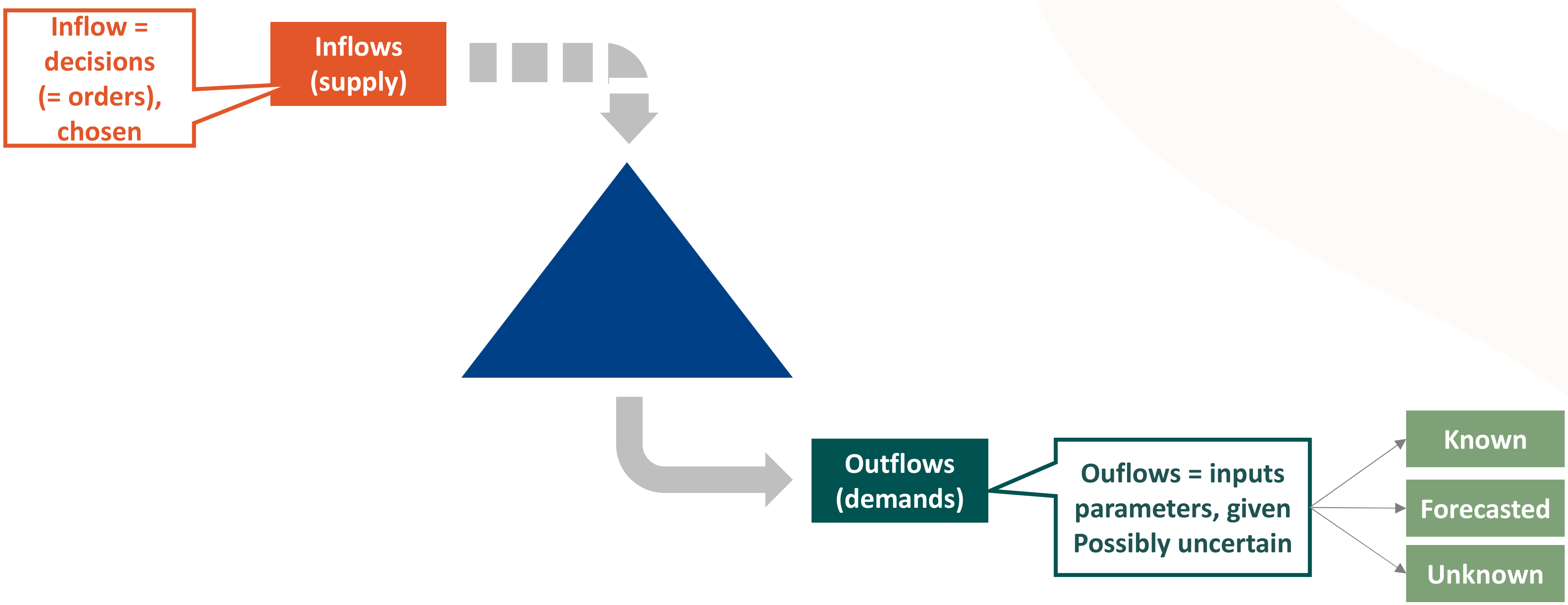
How much do I buy or produce, for which product and from whom?

What do we mean by policy ?

Definition	**A policy is a rule allowing to make decisions**
Stationary policy	**A policy is stationary if the action it chooses at time t only depends on the state of the process at time t.**

Basic tools of inventory management

Classical models for inventory management relies on the following (simplified) vision of inventories



Economic Order Quantity (Wilson formula)

Key question	**How much to purchase per order in order to minimize the handling costs ?**
Tradeoff	← **Small frequent orders**: **Less holding costs** / **Large infrequent orders** →: **Decreases the risk of stockouts**

Model & data

- Continuous time
- Deterministic and constant demand (D) per period
- Immediate refill (no lead-time)
- Costs :
  - Order cost C_o (per order)
  - Holding cost C_h
  - Purchase cost : C_p

Resolution

- Let X be the volume per order
- Total cost : $F(X) = C_o\, D/X + C_h/2\, X + DC_p$
- $dF/dX(X) = -C_o\, D/X^2 + C_h/2$
- $dF/dX(X) = 0 \rightarrow X^* = \sqrt{2\, C_o\, D / C_h}$

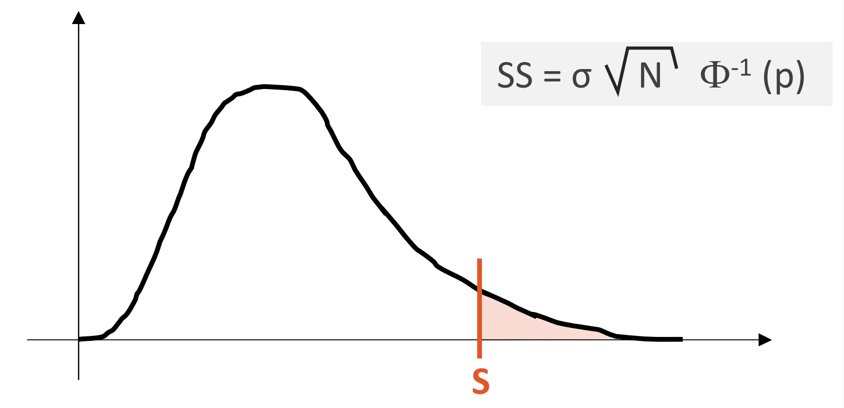
Safety stock with uncertain demand

Key question	**What is the minimum level to launch a new purchase so that the stockout risk during the lead-time be not more than a certain level p**
Remark	**Do not consider costs here, only service level**

Model & data

- The lead-time is known and has a duration of N time steps
- The demand **during one time-step** D follows a distribution F of mean μ and standard deviation σ
- p in [0,1] is the required service level
- S is the safety stock if :
  - SS = min X such that P[D ≤ E+X] ≥ p

Visualisation



Safety stock with Gaussian demand and uncertain lead time

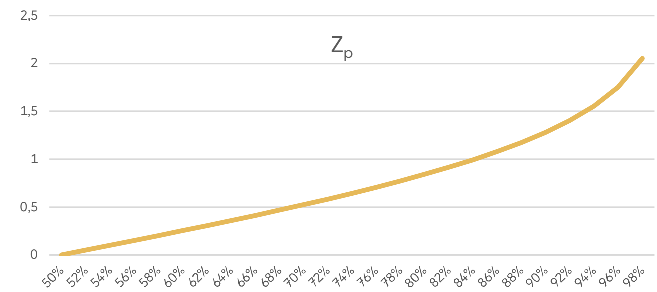
Key question	**What is the minimum level to launch a new purchase so that the stockout risk during the lead-time be not more than a certain level p**
Remark	**Do not consider costs here, only service level**

Model & data

- The lead-time L is uncertain and follows a Gaussian distribution F of mean μ_L and standard deviation σ_L
- The demand **during one time-step** D follows a **Gaussian** distribution F of mean μ_D and standard deviation σ_D
- p in [0,1] is the required service level
- S is the safety stock if :
  - SS = min X such that P[D ≥ E+X] ≤ p

Solution

$$SS = Z_p \sqrt{\mu_L^2 \sigma_D^2 + \mu_D^2 \sigma_L^2}$$



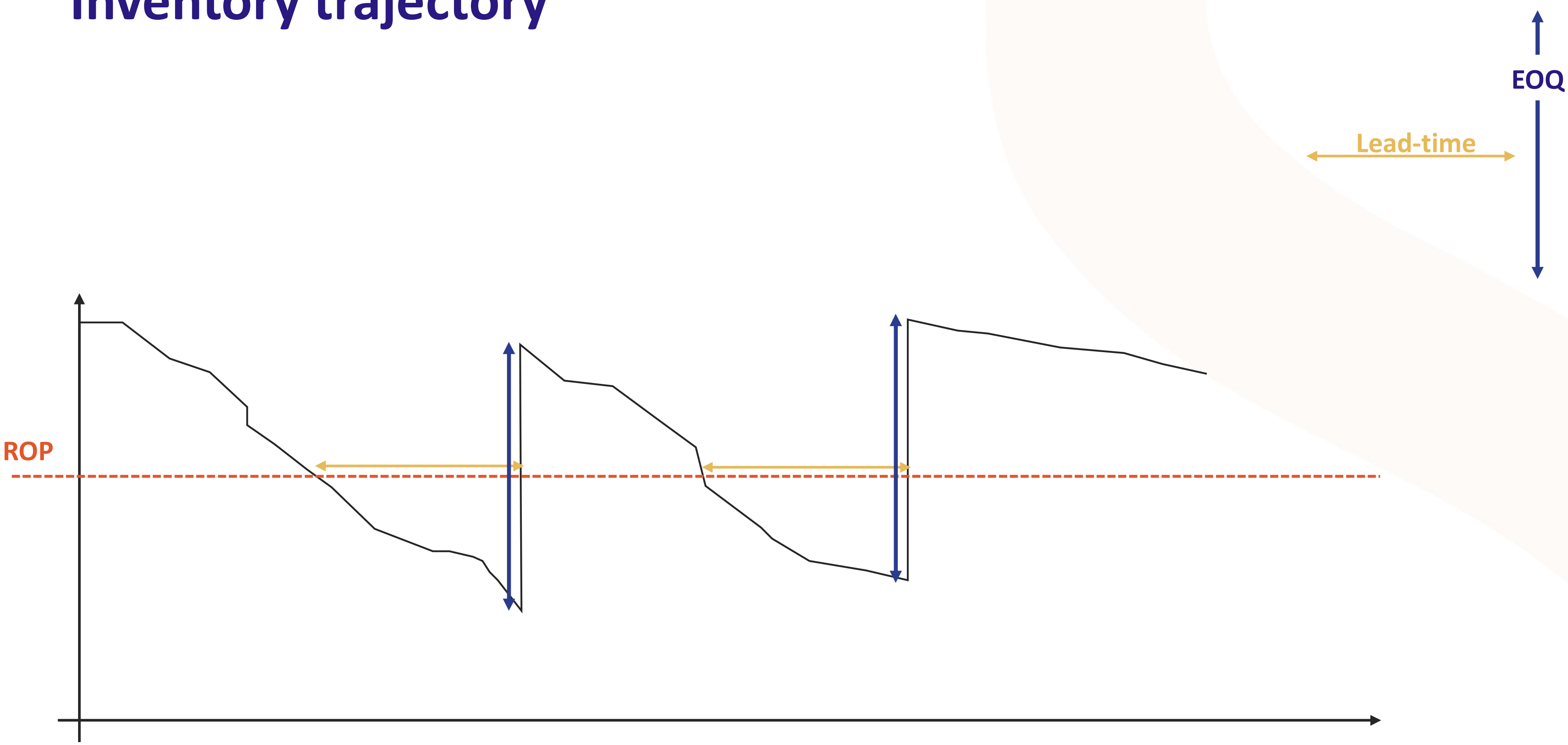
Re-order point Policy

= EOQ + LT

Definition	Re-order level L	**Level of the stock when an order is needed to be placed** for avoiding the risk of being out of stock
	Re-order quantity Q	**Quantity of the order** that is to be placed on the new purchase
Formula	Re-order level L	L = what you plan to use during LT + safety stock $L = \mu_L * \mu_D + SS$
	Re-order quantity Q	$Q = \sqrt{2\ C_o\ D\ /\ C_h}$ *#with D = what you plan to use during LT*

Policy : Wheneven S ≤ L, purchase Q

Inventory trajectory



Dynamic programming for inventory management

Introduction to dynamic programming

Which relationship between graph theory, inventory management and finance ?

Dynamic programming, a very powerful framework of algorithms :

- **Can be applied to problem with a « dynamic » structure (not necessarily linear)**
- **Can easily be extended to uncertainty consideration.**

What is a dynamic structure ?

- **Decisions are made in stages → *a sequence of decision x_t***
- **Discrete-time dynamic system → *a state variable that varies depending on the decisions***
- **Cost function is Markovian**

Captures a tradeoff between present and future costs by

- suming present cost and expected future cost
- assuming optimal decisions making for subsequent stages

***Example**:*

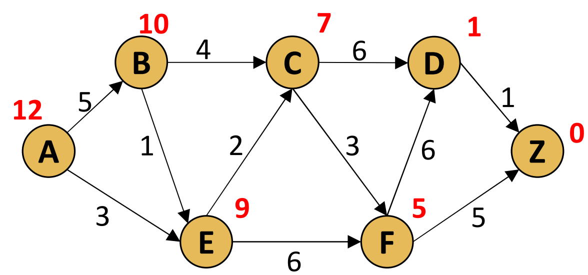
- *Which route shall I take ?*
- *How much shall I fill up / empty my inventory ?*
- *Do I have to sell my stock now or later ?*

Example of the shortest path in a graph

Illustration on the shortest path in a graph

Bellman's principle

"The subpolicy of an optimal policy is itself optimal"



Shortest path (AZ) is AECDZ if and only if

- the shortest path (AD) is AECD
- the shortest path (ED) is ECD
- … for each subpath of AECDZ

Suppose that I know that :

- The shortest path (CZ) is CDZ, which gives **7**
- The shortest path (FZ) is FZ, which gives **5**

In E, only two possibilities :

- Go to C, then do CDZ → 2+7 = **9**
- Go to F, then do FZ → 6+5 = 11

Then I know that the shortest path (EZ) is ECDZ that gives 9

<u>Key idea</u> :

From optimal solution and value of subsequent stages,

I can deduce the optimal solution and value of the current stage

→ Red values are called « Bellman values »

How works dynamic programming ?

Bellman's principle

"The subpolicy of an optimal policy is itself optimal"

Dynamic programming :

→ Divide the problem into one subproblem by stage

→ Solve subproblems in the recursive order

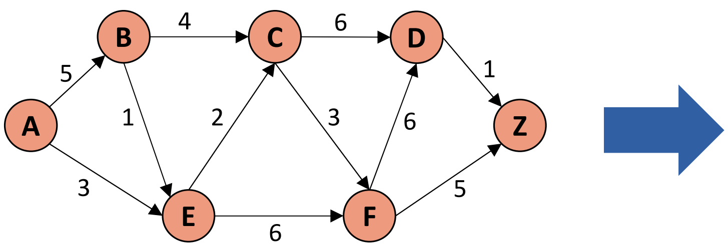
→ Prune partial decision sequences that cannot lead to the optimal solution

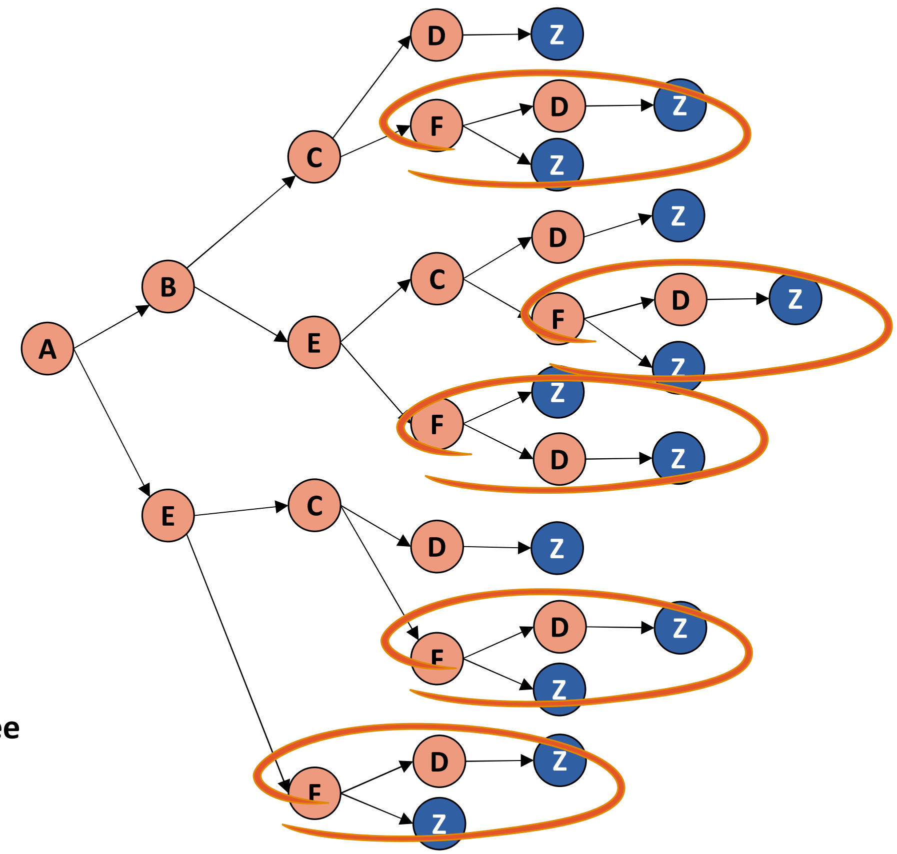
Dynamic programming is a smart way to efficiently explore a decision-tree

Problem that have

- **a dynamic structure**
- **a finite set of alternatives**

can be represented by a decision-tree :





Dynamic programming allows to explore efficiently a decision-tree
by pruning useless branches

Dynamic programming for inventory management
Let's practice

Problem description

Characteristics :

- N time step, with a demand $\mathbf{d_t}$ to satisfy
- At each time, decide how much to purchase, with a max = **orderMax**
- Minimize the cost, which is the sum of two components :
  - Purchasing cost $\mathbf{c^p_t}$ per unit
  - Stockout cost $\mathbf{c^s}$ per unit of unsatisfied demand
- Maximal inventory $\mathbf{S^{max}}$

Assumptions :

- Initial inventory = $\mathbf{S^0}$
- Final inventory is lost

Model description

Variables :

- Decision variables : $x_t \geq 0$
- State variables : $s_t \geq 0$

State equation :

- $s_{t+1} = s_t + x_t - d_t + \text{stockout}_t$

Cost function :

- $f_t(x_t, s_t) = c^p_t\, x_t + c^s\, \text{stockout}_t$

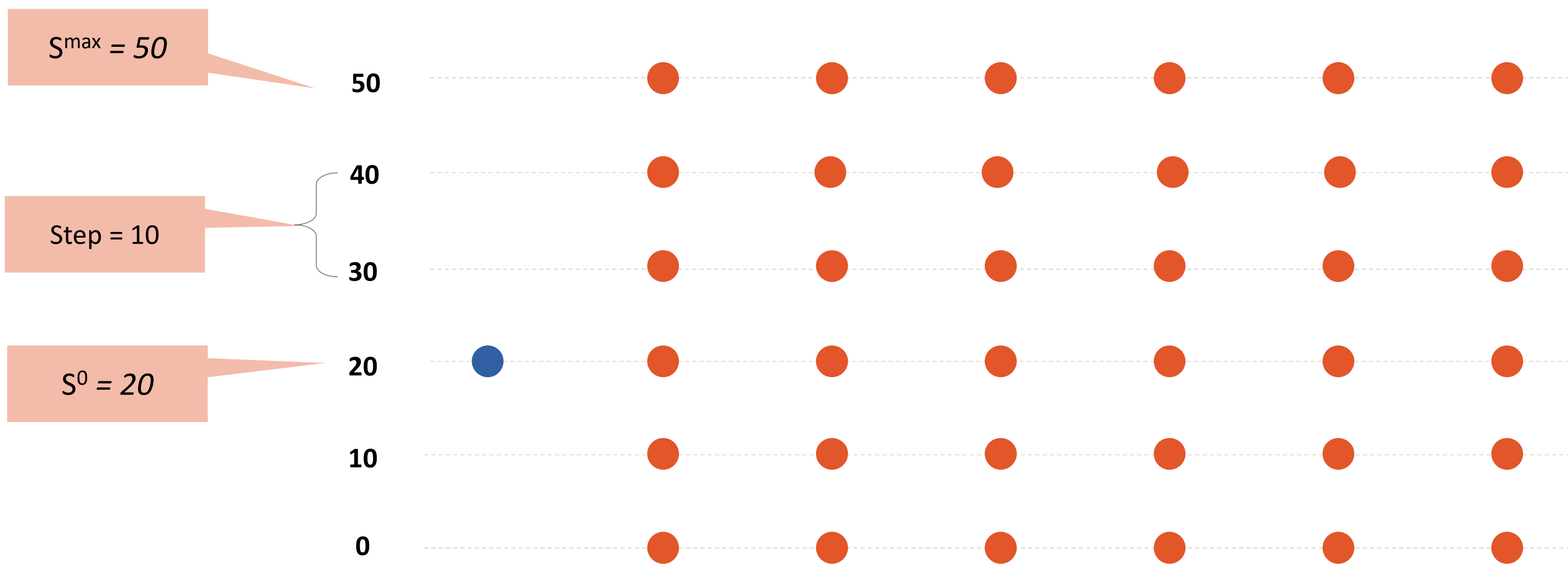
Recursive relationship :

- $B_{N+1}(s_t) = 0$, for all s_t
- $B_t(s_t) = \min\{x_t : s_{t+1} \geq 0\} \ \ (f_t(x_t, s_t) + B_{t+1}(s_{t+1}))$

Dynamic programming : let's practice (2)
First we have to discretize the possible states in order to build a grid

	0	1	2	3	4	5	6
d_t	*12*	*15*	*22*	*27*	*35*	*18*	*20*
c^p_t	*10*	*8*	*10*	*11*	*7*	*20*	*18*

Discretization of the state variable



At each stage, we compute the Bellman value of each state

At stage t

For all possible state s_{t+1}
Use $\mathbf{B_{t+1}(s_{t+1})}$

For all possible state s_t

$\cancel{x_t = s_{t+1} - s_t + d_t}$ $\cancel{F(s_{t+1}, s_t) = B_{t+1}(s_{t+1}) + f_t(x_t, s_t)}$	$\cancel{x_t = s_{t+1} - s_t + d_t}$ $\cancel{F(s_{t+1}, s_t) = B_{t+1}(s_{t+1}) + f_t(x_t, s_t)}$	$x_t = s_{t+1} - s_t + d_t$ $F(s_{t+1}, s_t) = B_{t+1}(s_{t+1}) + f_t(x_t, s_t)$
$\cancel{x_t = s_{t+1} - s_t + d_t}$ $\cancel{F(s_{t+1}, s_t) = B_{t+1}(s_{t+1}) + f_t(x_t, s_t)}$	$x_t = s_{t+1} - s_t + d_t$ $F(s_{t+1}, s_t) = B_{t+1}(s_{t+1}) + f_t(x_t, s_t)$	$x_t = s_{t+1} - s_t + d_t$ $F(s_{t+1}, s_t) = B_{t+1}(s_{t+1}) + f_t(x_t, s_t)$
$x_t = s_{t+1} - s_t + d_t$ $F(s_{t+1}, s_t) = B_{t+1}(s_{t+1}) + f_t(x_t, s_t)$	$x_t = s_{t+1} - s_t + d_t$ $F(s_{t+1}, s_t) = B_{t+1}(s_{t+1}) + f_t(x_t, s_t)$	$x_t = s_{t+1} - s_t + d_t$ $F(s_{t+1}, s_t) = B_{t+1}(s_{t+1}) + f_t(x_t, s_t)$

→ $\mathbf{B_t(s_t)}$ = *the minimum value of* $F(s_{t+1}, s_t)$ *such that* x_t *is feasible*

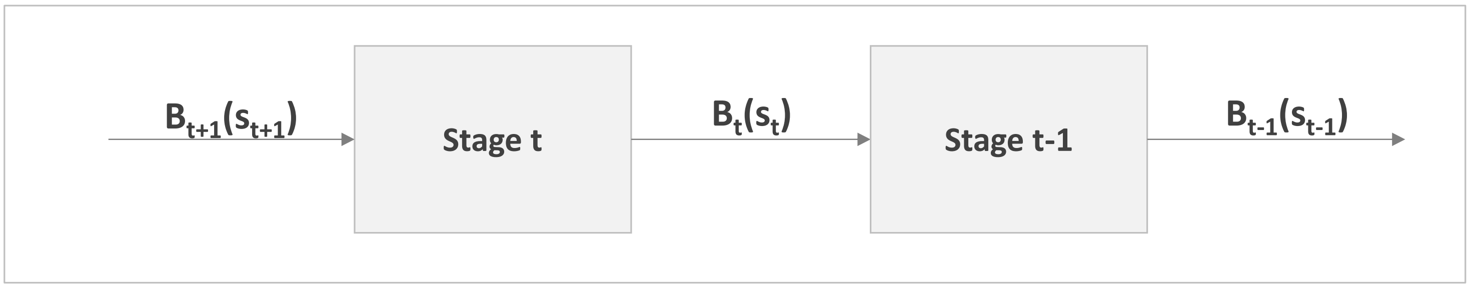
→ $\mathbf{(s^*_{t+1}, x^*_t)}$ = *the associated value of* (s_{t+1}, x_t)

Resolution of an optimisation problem for each possible state $\mathbf{s_t}$

The algorithm is based on the Bellman recursive relationship

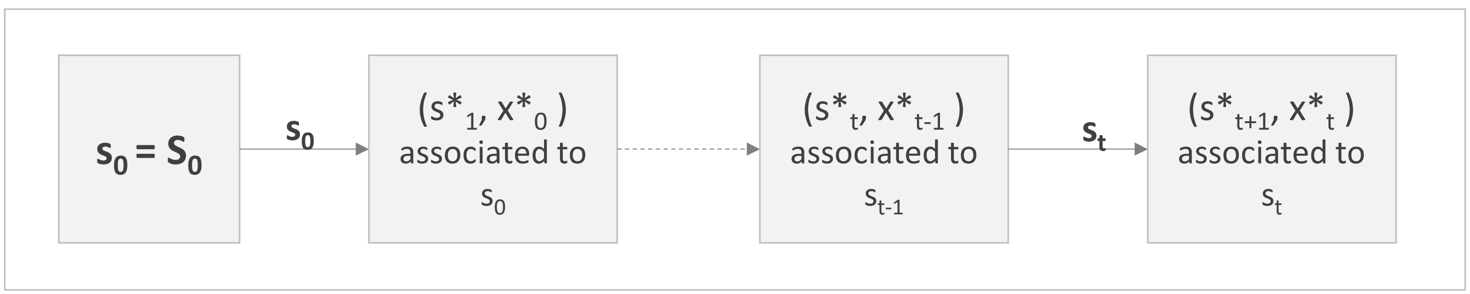
1

Recursion to compute Bellman values



2

Deduce the optimal states and decisions



Dynamic programming : let’s practice (3)

Determine the values xxx

	0	1	2	3	4	5	6
d_t	*12*	*15*	*22*	*27*	*35*	*18*	*20*
c^p_t	*10*	*8*	*10*	*11*	*7*	*20*	*18*
50		*912*	*782*	*612*	*365*	*250*	*50 = 0 x 18 + 1 x 50*
40		*982*	*822*	*662*	***425***	*230*	*40 = 0 x 18 + 1 x 40*
30		*1052*	*912*	*712*	*485*	*210*	*30 = 0 x 18 + 1 x 30*
20	***XXX***	***XXX***	*1002*	*762*	*545*	*370*	*20 = 0 x 18 + 1 x 20*
10		*1192*	*1092*	*862*	*605*	*530*	*190 = 10 x 18 + 1 x 10*
0		*1262*	*1182*	*962*	*665*	*720*	*360 = 20 x 18*

Dynamic programming : let’s practice (4)

The final grid

	0	1	2	3	4	5	6
d_t	12	15	22	27	35	18	20
c^p_t	10	8	10	11	7	20	18
50		912	782	612	365	250	50 = 0 x 18 + 1 x 50
40		982	822	662	**425**	230	40 = 0 x 18 + 1 x 40
30		1052	912	712	485	210	30 = 0 x 18 + 1 x 30
20	1232	1122	1002	762	545	370	20 = 0 x 18 + 1 x 20
10		1192	1092	862	605	530	190 = 10 x 18 + 1 x 10
0		1262	1182	962	665	**720**	360 = 20 x 18

Bellman values can be used to make your inventory decisions in a « value-driven » way

Optimal policy

Optimal trajectory = main outcome

Bellman values = a valuable by-product :

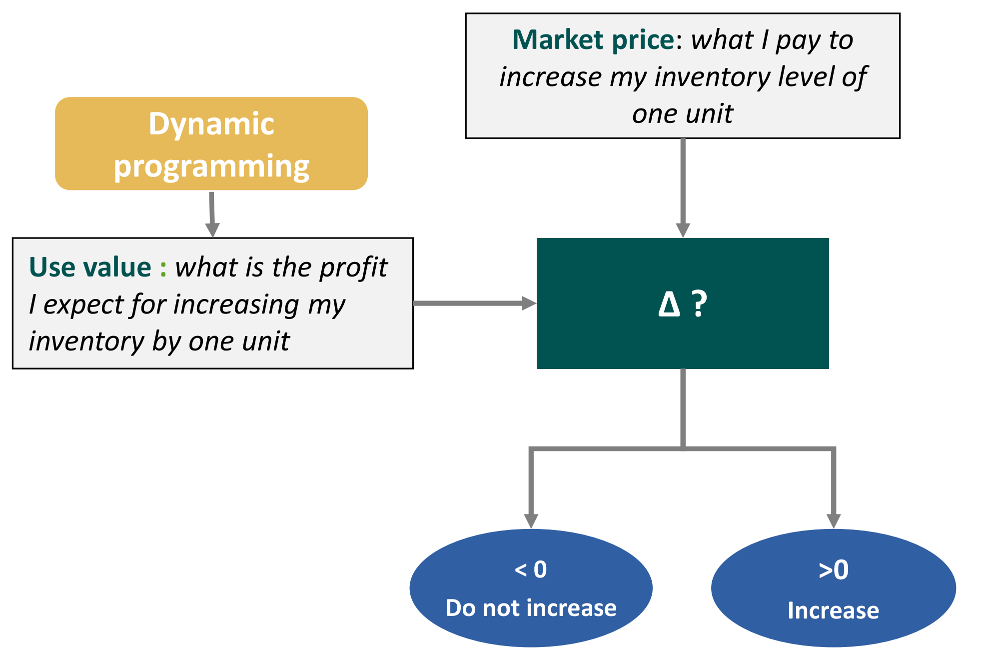
→ *For each point of the grid : we know how to carry on being optimal*

→ *Notion of optimal policy*

A optimal policy is a set of rules enabling to adapt the decisions to a dynamic context

Static decisions ⇔ **Dynamic decisions through optimal policy**

In practice



Dynamic programming can be extended to consider uncertainty

Example of uncertainty :

- *Sourcing cost*
- *Demand*

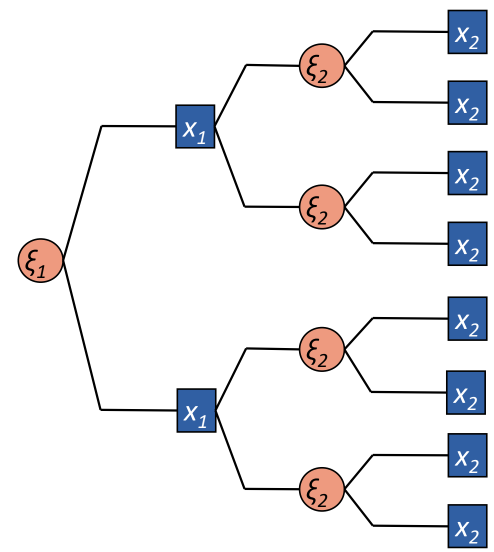
<u>At each stage t :</u>

→ Determine the Bellman value for each possible value of the uncertain data :

$B_t(\xi_t, s_t) = \min\{x_t : s_{t+1} \geq 0\} \left(f_t(x_t, s_t, \xi_t) + B_{t+1}(s_{t+1}) \right)$

→ Take its expected value : $B_t(s_t) = E[B_t(\xi_t, s_t)]$

Exploration of a stochastic decision-tree



Dynamic programming reveals all its power when considering uncertainty

Dynamic programming for inventory management
Let's practice with a stochastic demand

Problem description

Characteristics :

- N time step, with a random demand $\mathbf{d_t}$ to satisfy
- At each time, decide how much to purchase, with a max = **orderMax**
- Minimize the cost, which is the sum of two components :
  - Purchasing cost $\mathbf{c^p_t}$ per unit
  - Stockout cost $\mathbf{c^s}$ per unit of unsatisfied demand
- Maximal inventory $\mathbf{S^{max}}$

Assumptions :

- Initial inventory = $\mathbf{S^0}$
- Final inventory is lost

Model description

Variables :

- Decision variables : $x_t \geq 0$
- State variables : $s_t \geq 0$

State equation :

- $s_{t+1} = s_t + x_t - d_t + \text{stockout}_t$

Cost function :

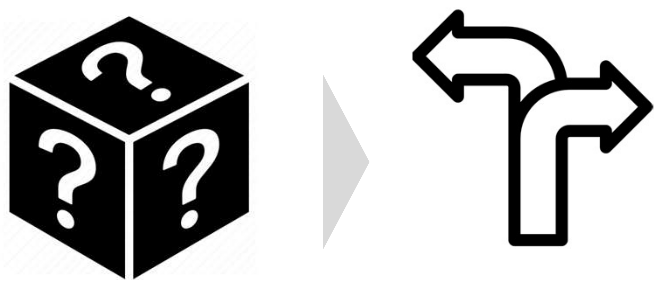
- $f_t(x_t, s_t) = c^p_t\, x_t + c^s\, \text{stockout}_t$

Recursive relationship :

- $B_{N+1}(s_t) = 0$, for all s_t
- $B_t(s_t) = \min\{x_t : s_{t+1} \geq 0\} \;\; (f_t(x_t, s_t) + B_{t+1}(s_{t+1}))$

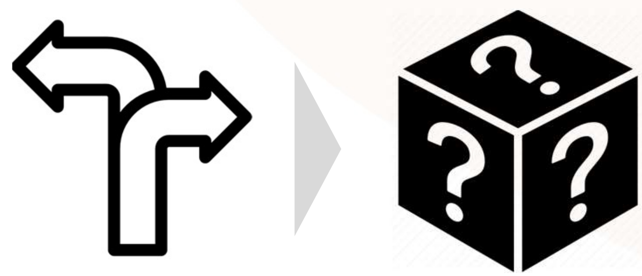
Stochastic programming : 2 possible schemes

Cas 1 : hazard-decision



$B_t (s_t) = E\ [\min\{x_t \text{ feasible}\}\ (\ f_t(x_t, s_t, \xi_t))+ B_{t+1}(s_t+1)\ )$

Cas 2 : decision-hazard



$B_t (s_t) = \min\{x_t \text{ feasible}\}\ E\ [\ f_t(x_t, s_t, \xi_t))+ B_{t+1}(s_t+1)\ ]$

But what does « x_t feasible” mean in this case ?

Bellman values can be used to dynamically optimize your inventory decisions and even more…

Optimal policy

Optimal trajectory = main outcome in deterministics

Bellman values = a valuable by-product :

→ *For each point of the grid : we know how to carry on being optimal*

→ ***Notion of optimal policy***

A optimal policy is a set of rules enabling to adapt the decisions to a dynamic context

Static decisions ⟺ **Dynamic decisions through optimal policy**

In practice

Suppose you are a fertilizer producer that shall purchase raw material on the market…

When optimizing your production : which cost do you consider for raw materials ?

- If you have no (physical / virtual) storage : it's easy, you consider the Spot price
- Suppose you have some physical storage ? Is Spot price still relevant ?

Bellman value(s) : the (expected) cost to attain s.

Bellman value derivative(s) = Use Value : the (expected) cost to get one more unit of inventory, if we are at level s.

→ ***the most appropriate cost to use for optimization***

→ ***valid provided that the raw material procurement follows the optimal policy***

Pros & Cons of Dynamic Programming

Pros	Cons
1. Efficient (elegant)	1. Requires a specific structure of the problem
2. Natural extension for considering uncertainty	2. Difficulty to consider some multi-period constraints such as : • Delay in delivery • Constraints relating successive filling / emptying values
3. No linearity restriction	3.
4. Provides an optimal policy	4. “ Curse of dimensionality ”

As soon as you can use Dynamic Programming, just use it