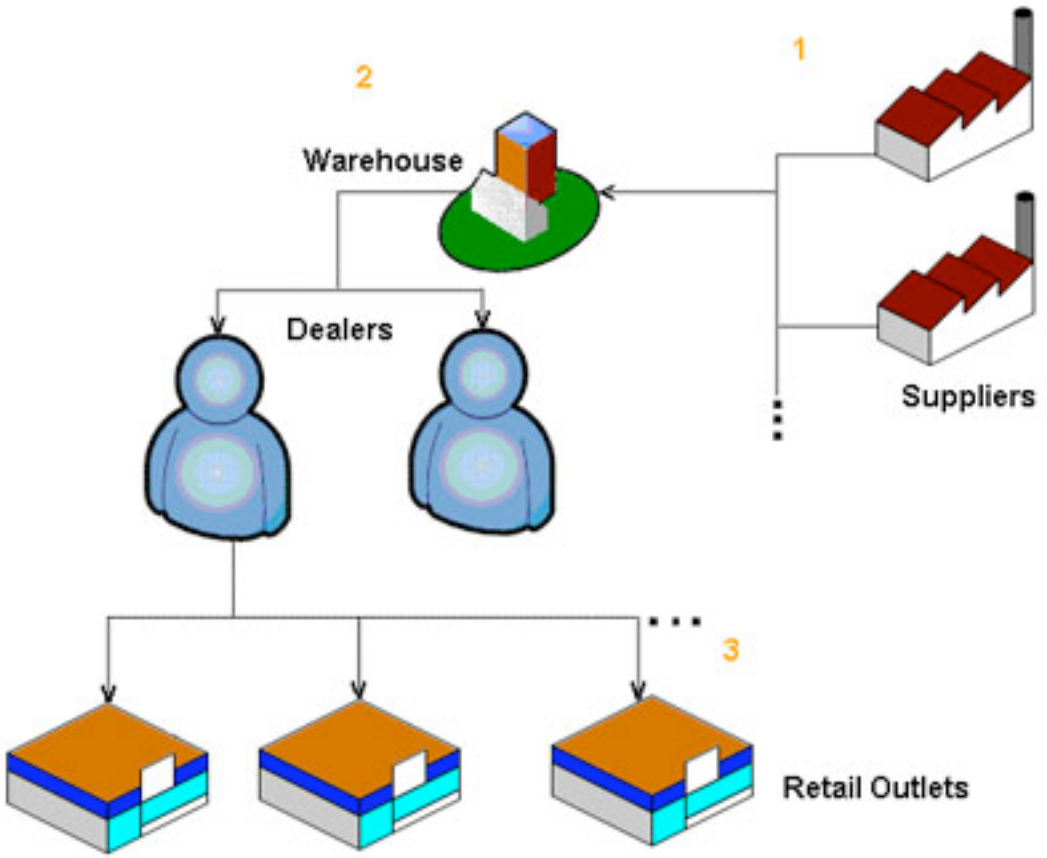
Un algorithme d'approximation pour le problème One-Warehouse Multi-Retailers

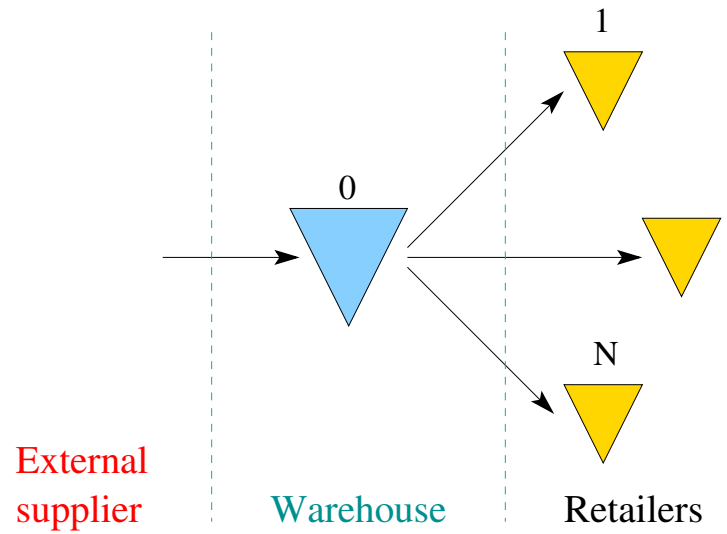
Jean-Philippe Gayon, Guillaume Massonnet, Christophe Rapine, Gautier Stauffer

JFRO - mars 2013

Réseau de distribution multi-échelon

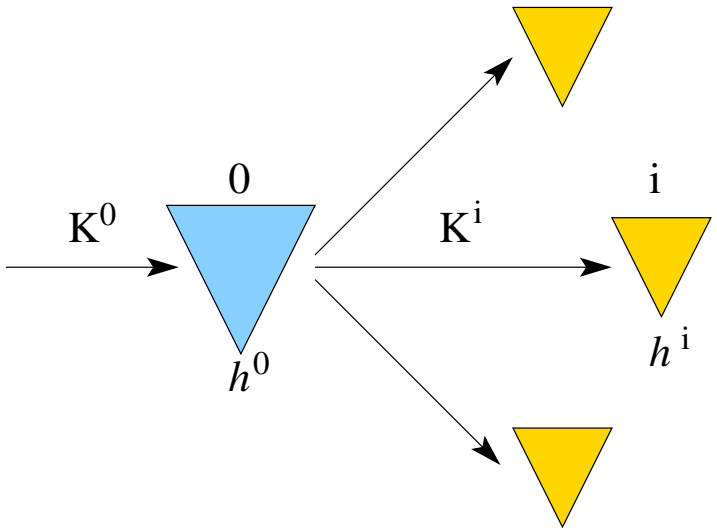


Le problème OWMR



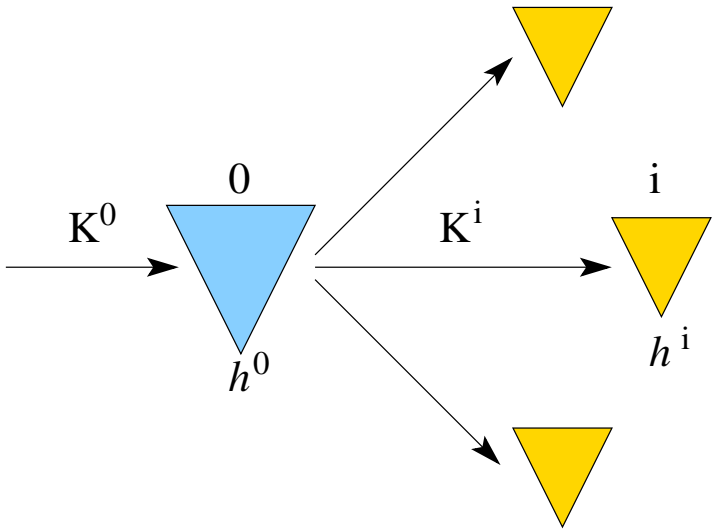
- **One Warehouse :** un entrepôt
- **Multiple Retailers :** N détaillants, chaque détaillant i devant satisfaire une demande d_{it} à la période t,
- Un horizon de temps discrétisé en T périodes.
- Chaque détaillant s'approvisionne auprès de l'entrepôt,
- L'entrepôt s'approvisionne auprès d'un fournisseur externe.

Coût d'une politique



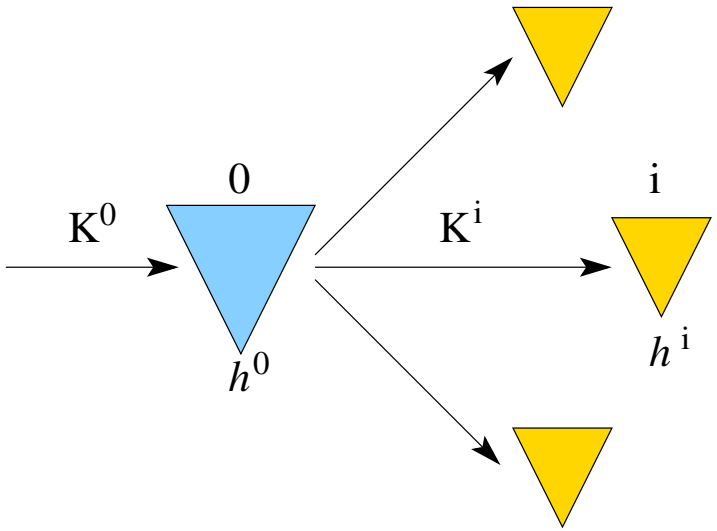
- Coût fixe de commande K^i payé si le site i commande
- Coût de possession $h_t^i(x)$ payé pour conserver une quantité x en stock de la période t à la période $t+1$ au site i

Coût d'une politique



- Coût fixe de commande K^i payé si le site i commande
- Coût de possession $h_t^i(x)$ payé pour conserver une quantité x en stock de la période t à la période $t+1$ au site i

Coût d'une politique



- Coût fixe de commande K^i payé si le site i commande
- Coût de possession $h_t^i(x)$ payé pour conserver une quantité x en stock de la période t à la période $t+1$ au site i

$\Rightarrow$ minimiser le coût total du système sur l'horizon de temps

Joint Replenishment Problem (JRP)

Problème d'approvisionnement conjoint

- N produits différents à approvisionner sur un horizon T
- Chaque produit est caractérisé par un coût de stockage h_i et des demandes d_{it} à satisfaire à chaque période.
- Chaque commande occasionne un coût fixe (majeur) de K €
- Chaque commande comportant la pièce i occasionne un coût fixe (mineur) de K_i €

Ce qui rend le problème difficile :

- Economie d'échelle possible sur les coûts majeurs de commande, en commandant plusieurs produits simultanément

Exemple

On connait les côuts logistiques pour chaque produit :

coûts	commande	stockage
pièce 1	40€	3€/unité
pièce 2	20€	1€/unité
pièce 3	30€	2€/unité

Coût de livraison de 50 €

Exemple

On connait les côuts logistiques pour chaque produit :

coûts	commande	stockage
pièce 1	40€	3€/unité
pièce 2	20€	1€/unité
pièce 3	30€	2€/unité

Coût de livraison de 50 €

On connaît les demandes de chaque produit par période sur un horizon T :

périodes	1	2	3	4	5	6
pièce 1	7	5	2	4	7	3
pièce 2	5	3	3	6	2	5
pièce 3	4	4	7	2	5	3

Dans la littérature

Theorème [Arkin *et al* 1989]

Le problème OWMR est $\mathcal{NP}$-difficile même pour des coûts de possession linéaires.

Malgré l'importance de OWMR en gestion de stock multi-échelon, peu de résultats d'approximation :

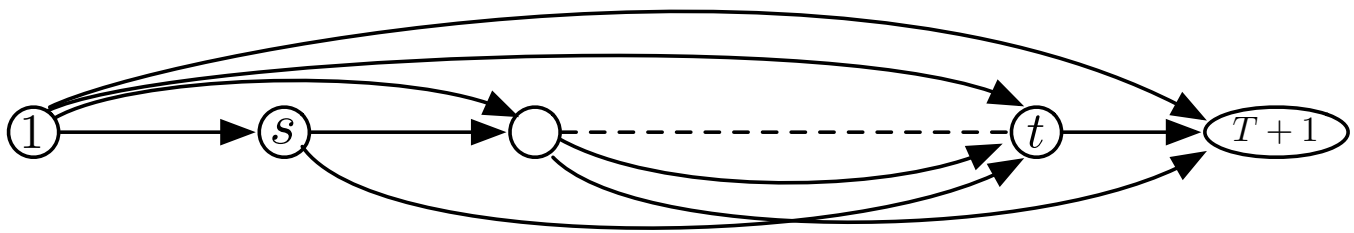
- Pour les politiques à temps continu (avec des taux de demande constants), Roundy [Management Science 1985] a proposé les politiques *power-of-two*, garanties à 2%.
- Pour les politiques à temps discret (notre problème), la meilleure approximation est due à Levi, Roundy, Shmoys & Sviridenko [Management Science 2008]. Leur approche basée sur la programmation linéaire est garantie à 1.8.

Notre résultat : une 2-approximation très rapide, basée sur des idées très simples.

Le cas d'un seul stock

Le problème avec un seul stock a été extrêmement étudié :

- Problème de dimensionnement de lot sans capacité (ULSP)
- Wagner & Within [Management Science 1958] ont proposé un algorithme exact en $\mathcal{O}(T^2)$ par programmation dynamique.



- Pour des coûts de possession linéaires, Aggarwal & Park [Operations Research 1993] ont proposé un algorithme linéaire en $\mathcal{O}(T)$ exploitant les matrices de Monge.

Principe de l'algorithme UNCROSSING

1. Décomposer le problème en $N + 1$ problèmes indépendants mono-stock de dimensionnement de lots (ULSP) :
   - Un problème mono-item pour chaque détaillant i
   - Un problème multi-item pour l'entrepôt

2. Déterminer une politique optimale π_i^* pour chacun

   $\longrightarrow$ facile

3. Reconstruire une politique réalisable à partir des politiques π_i^* pour chaque problème de dimensionnement de lots

   $\longrightarrow$ comment faire ?

Hypothèses

Les détaillants peuvent être partitionnés en 2 ensembles :

1. W-détaillants : le coût de possession à l'entrepôt est moins cher que chez le détaillant, *pour toutes les périodes*.
2. J-détaillants : le coût de possession chez le détaillant est moins cher qu'à l'entrepôt, *pour toutes les périodes*.

Hypothèses

Les détaillants peuvent être partitionnés en 2 ensembles :

1. *W*-détaillants : le coût de possession à l'entrepôt est moins cher que chez le détaillant, *pour toutes les périodes*.
2. *J*-détaillants : le coût de possession chez le détaillant est moins cher qu'à l'entrepôt, *pour toutes les périodes*.

- Pour le JRP, tous les "détaillants" sont des *J*-détaillants.
- Pour un réseau de distribution usuel, tous les "détaillants" sont des *W*-détaillants.

Hypothèses

Les détaillants peuvent être partitionnés en 2 ensembles :

1. W-détaillants : le coût de possession à l'entrepôt est moins cher que chez le détaillant, *pour toutes les périodes*.
2. J-détaillants : le coût de possession chez le détaillant est moins cher qu'à l'entrepôt, *pour toutes les périodes*.

- Pour le JRP, tous les "détaillants" sont des J-détaillants.
- Pour un réseau de distribution usuel, tous les "détaillants" sont des W-détaillants.

Les coûts fixent sont stationnaires aux détaillants : $K_t^i = K^i$.

$\Rightarrow$ Sinon le problème OWMR n'est pas dans APX [Chan et al, Management Science 2000]

Décomposition en problèmes de dimensionnement de lot

(S_i) Le détaillant i est considéré comme un système mono-stock, avec sa propre demande d_{it} et ses propres coûts de commande et de possession

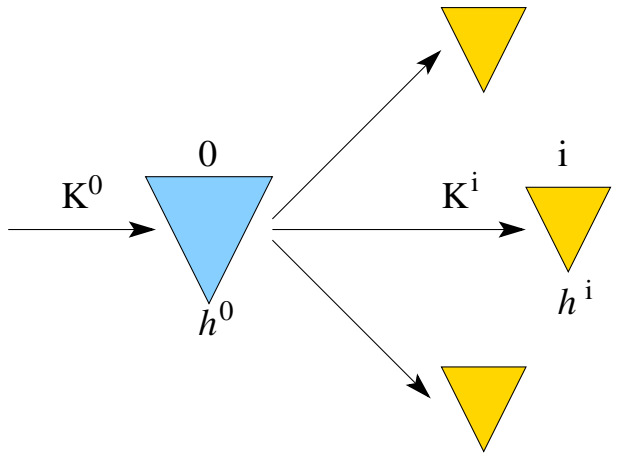
(S_0) L'entrepôt est considéré comme un système mono-stock multi-produit : chaque J-détaillant i est vu comme un produit i, les W-détaillants sont agrégés en un seul produit 0.

- Le coût de possession pour le produit $i \in J$ est $h^i(x)$
  $\longrightarrow$ coût au détaillant i
- Le coût de possession pour le produit 0 est $h^0(x)$
  $\longrightarrow$ coût à l'entrepot

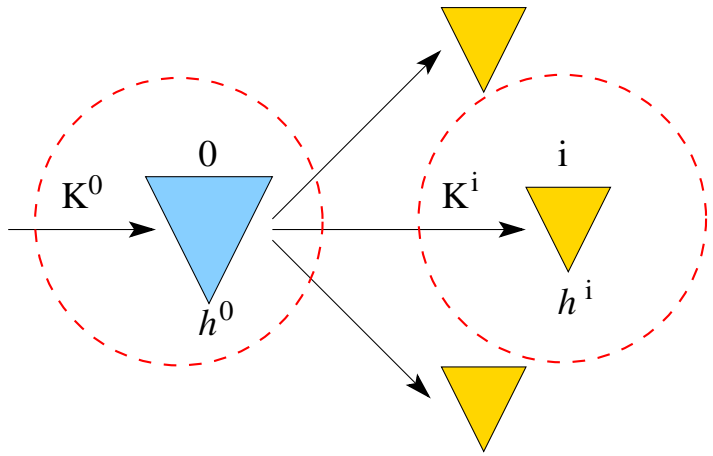
Le coût fixe de commande est K^0, le coût à l'entrepôt.

$\rightarrow$ (S_0) est un problème OWMR sans coût fixe de commande aux détaillants

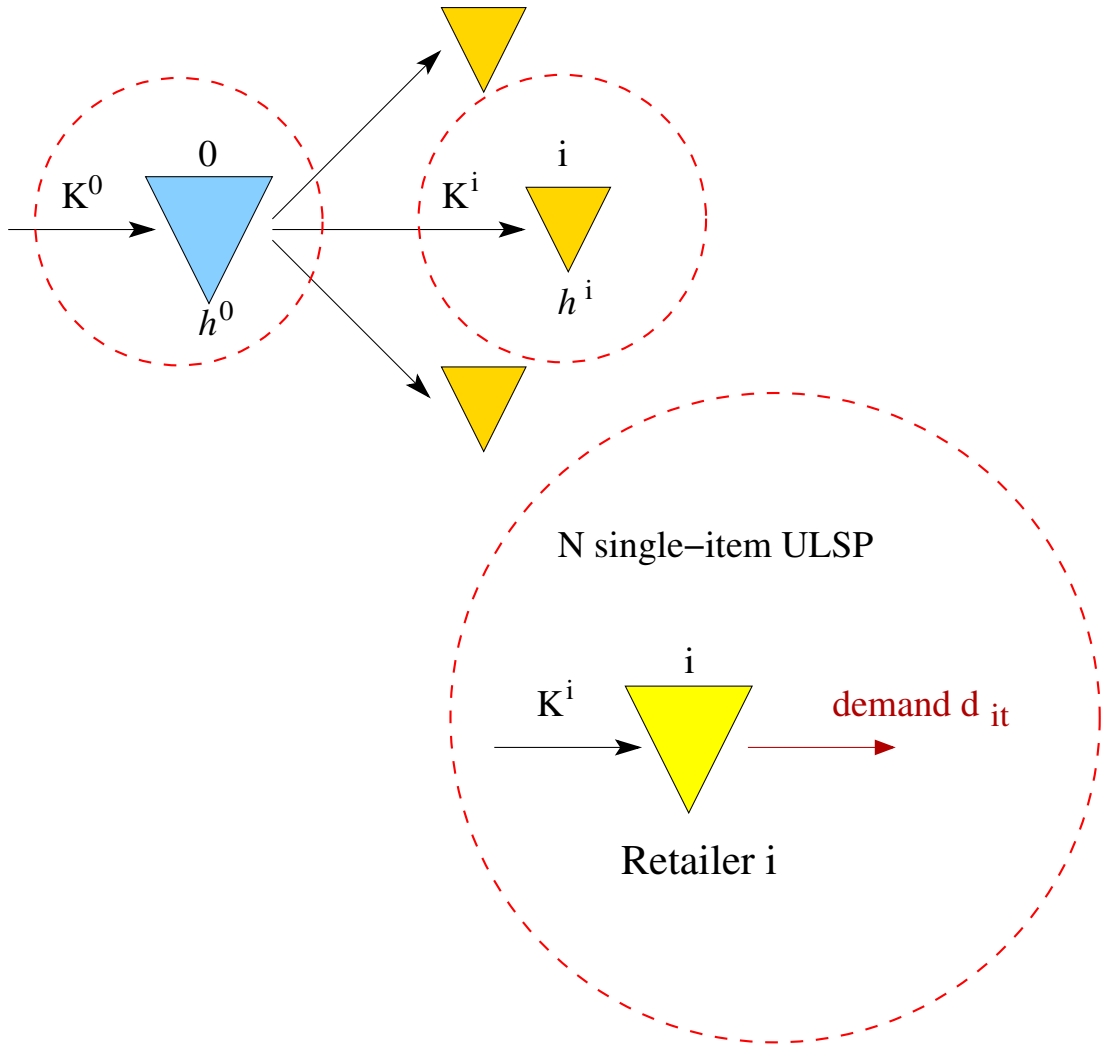
Décomposition en problèmes de dimensionnement de lot



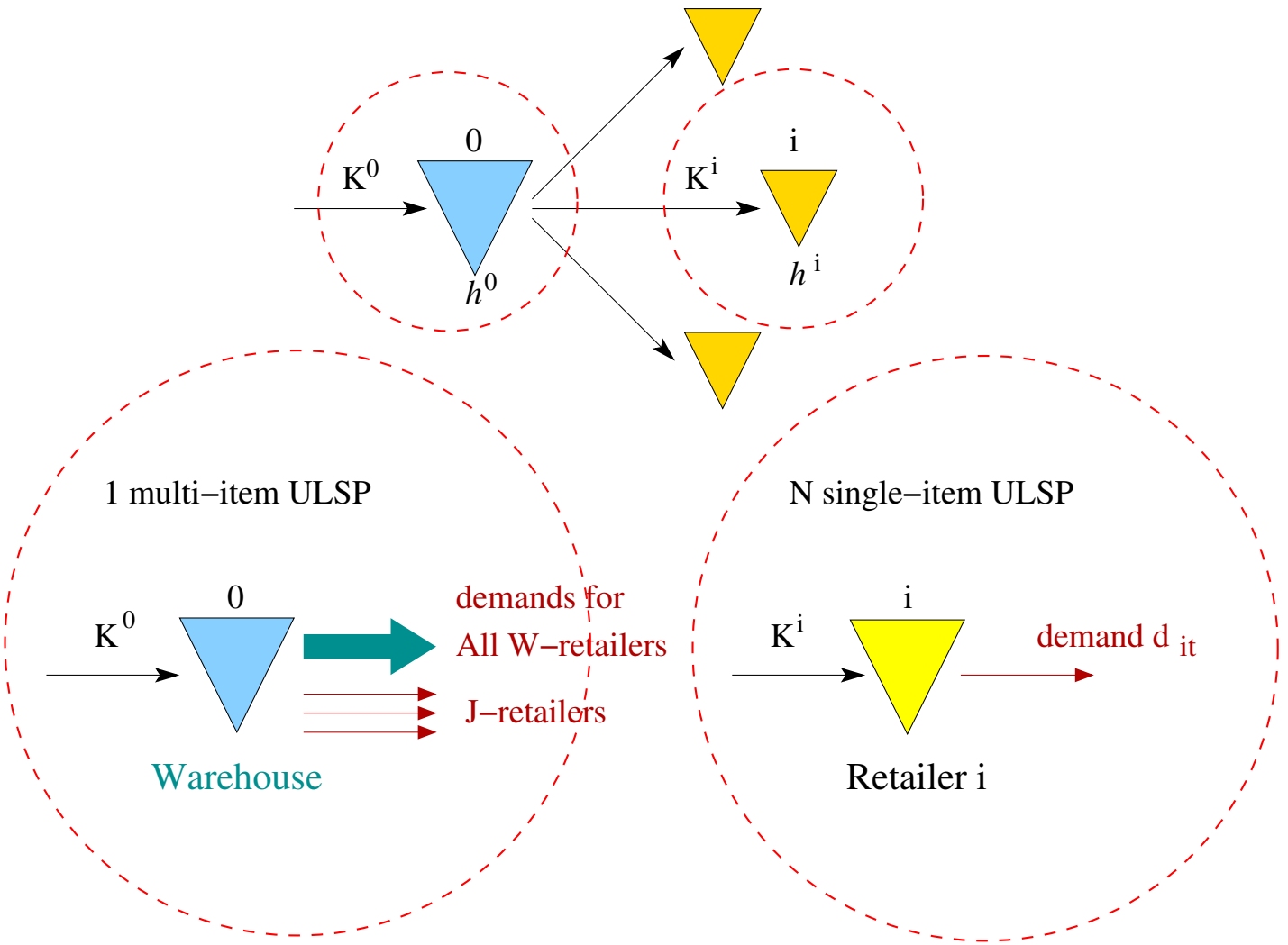
Décomposition en problèmes de dimensionnement de lot



Décomposition en problèmes de dimensionnement de lot



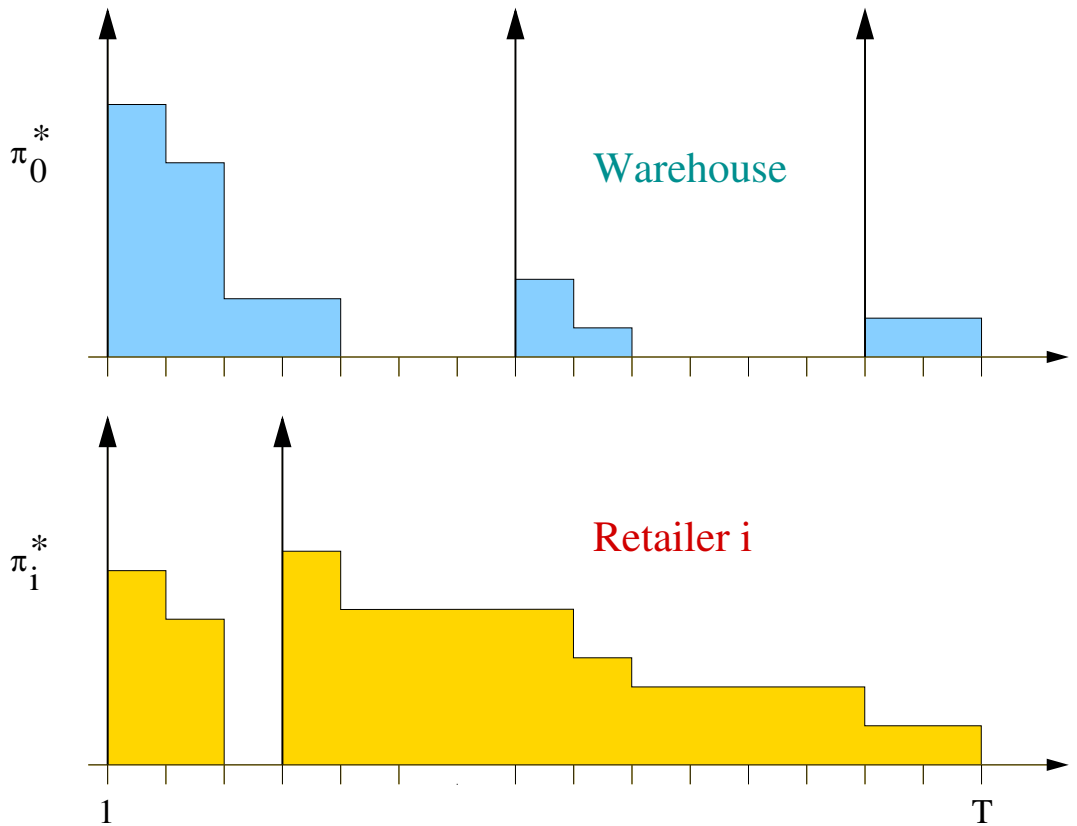
Décomposition en problèmes de dimensionnement de lot



Comment reconstruire une politique réalisable ?

- Soit π_i une politique réalisable pour (S_i) et $\mathcal{C}(\pi_i)$ son coût.
- Combinons les politiques mono-stock : $\pi = (\pi_0, \ldots, \pi_N)$.

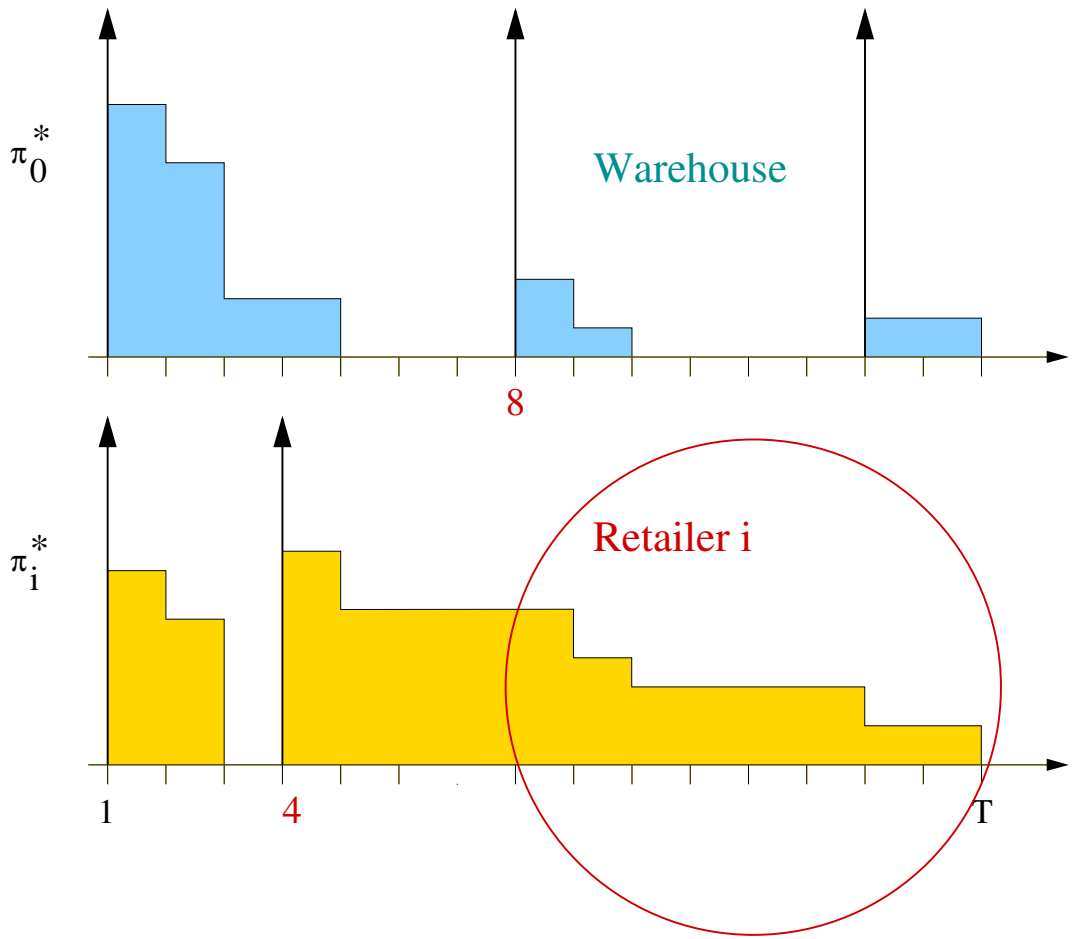
La politique π n'est pas réalisable en général pour OWMR



Comment reconstruire une politique réalisable ?

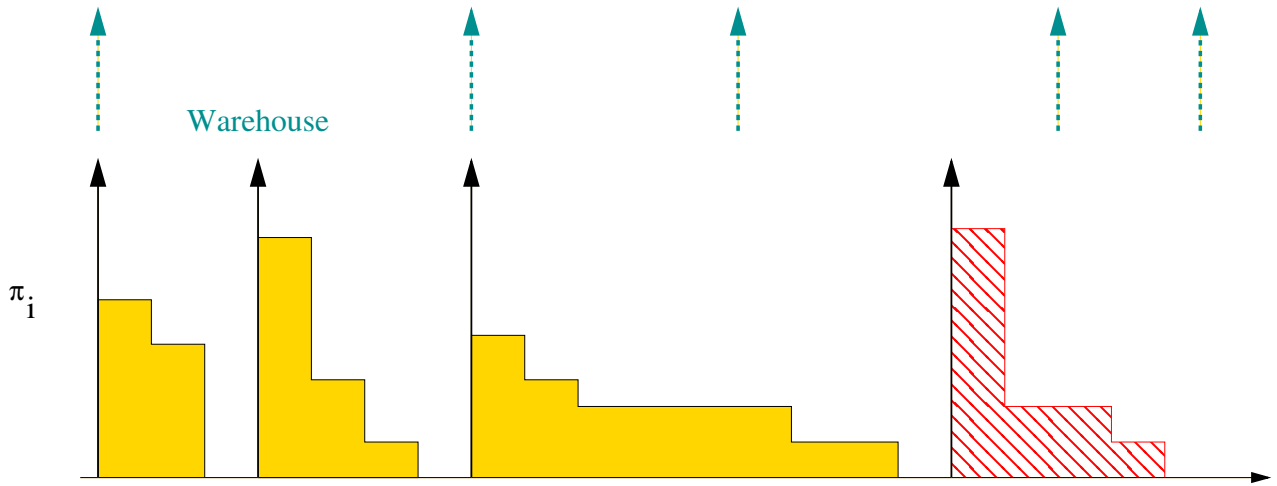
- Soit π_i une politique réalisable pour (S_i) et $\mathcal{C}(\pi_i)$ son coût.
- Combinons les politiques mono-stock : $\pi = (\pi_0, \ldots, \pi_N)$.

La politique π n'est pas réalisable en général pour OWMR



Politique non-croisée

Deux commandes consécutives s, s' à un détaillant i *croisent* les commandes de l'entrepôt si il existe 2 commandes consécutives r, r' telles que $r < s < r' < s'$.



Politique non-croisée

Une politique est *non-croisée* si aucune de ses commandes n'est croisée.

Politique non-croisée

Résultat

Si les politiques locales π_i sont non-croisées, il est possible de construire en temps linéaire $O(NT)$ un politique π^u réalisable de coût au plus $\mathcal{C}(\pi^u) \leq \sum_{i=0}^{N} \mathcal{C}(\pi_i)$.

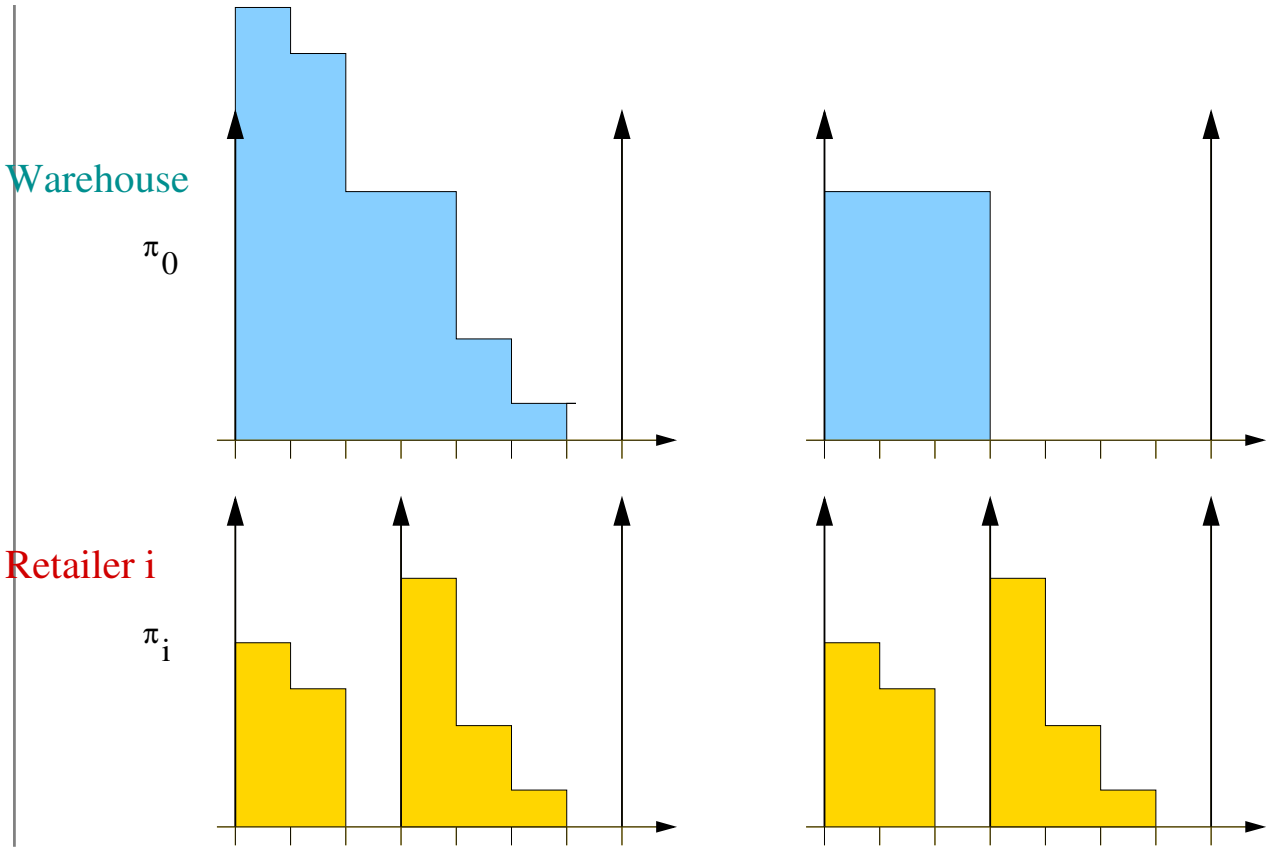
Politique non-croisée

Résultat

Si les politiques locales π_i sont non-croisées, il est possible de construire en temps linéaire $O(NT)$ un politique π^u réalisable de coût au plus $\mathcal{C}(\pi^u) \leq \sum_{i=0}^{N} \mathcal{C}(\pi_i)$.

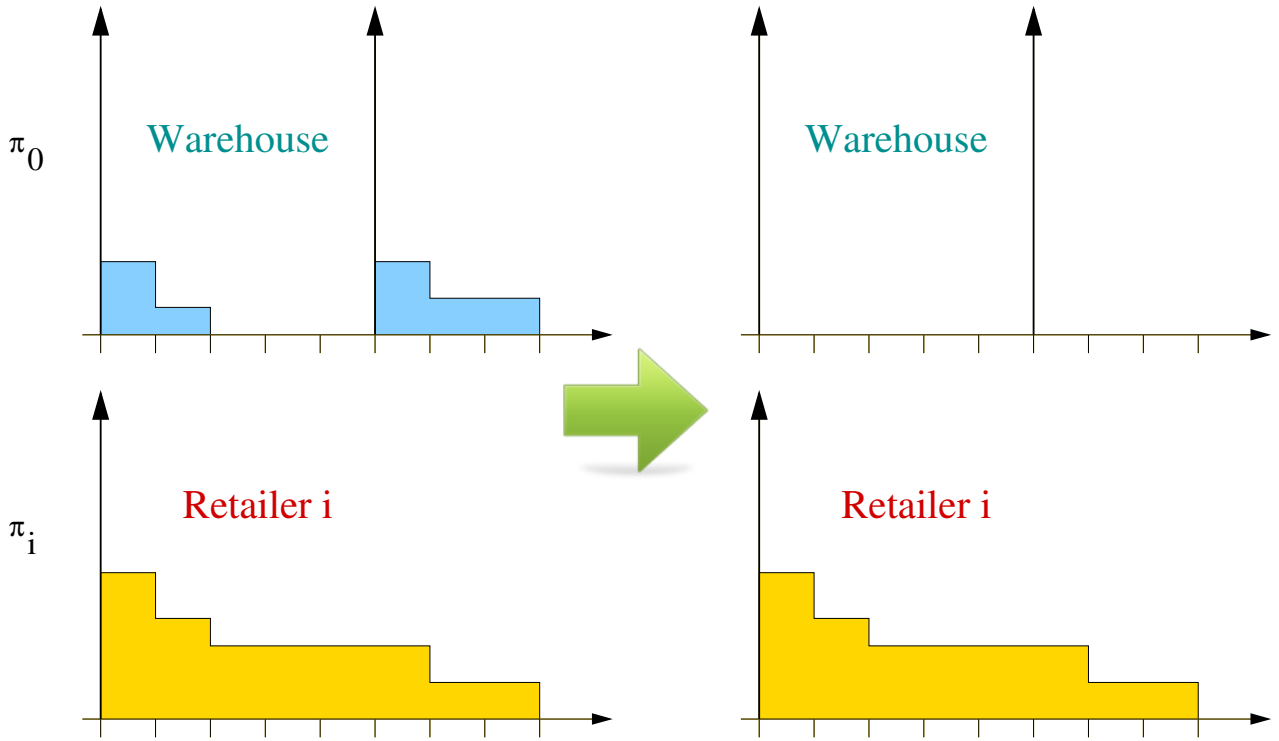
- Un W-détaillant i suit sa politique locale π_i.
- Un J-détaillant i synchronise ses ordres avec ceux de l'entrepôt
- La politique π_0^u à l'entrepôt commande aux mêmes périodes que π_0 et est ZIO.

Idée de la preuve pour les commandes courtes



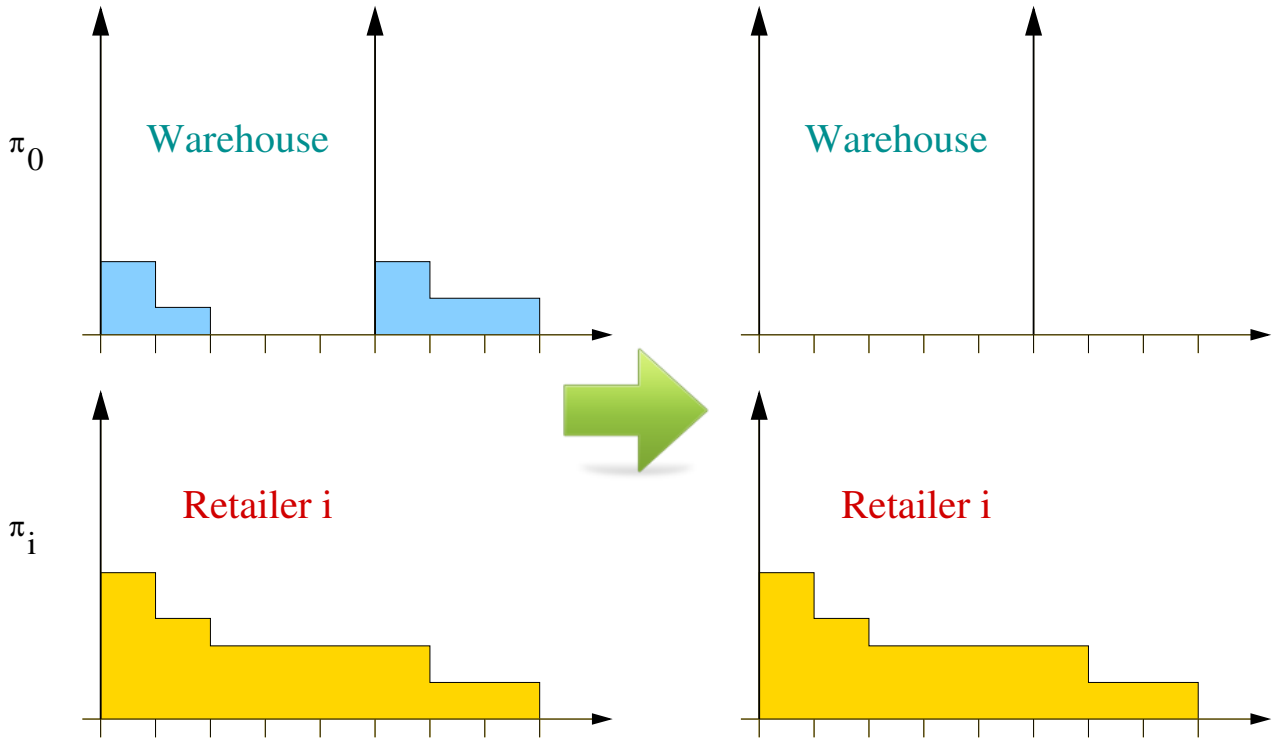
Les coûts de stockage à l'entrepôt diminue !

Idée de la preuve pour les commandes longues



Aucun stock n'est conservé à l'entrepôt

Idée de la preuve pour les commandes longues



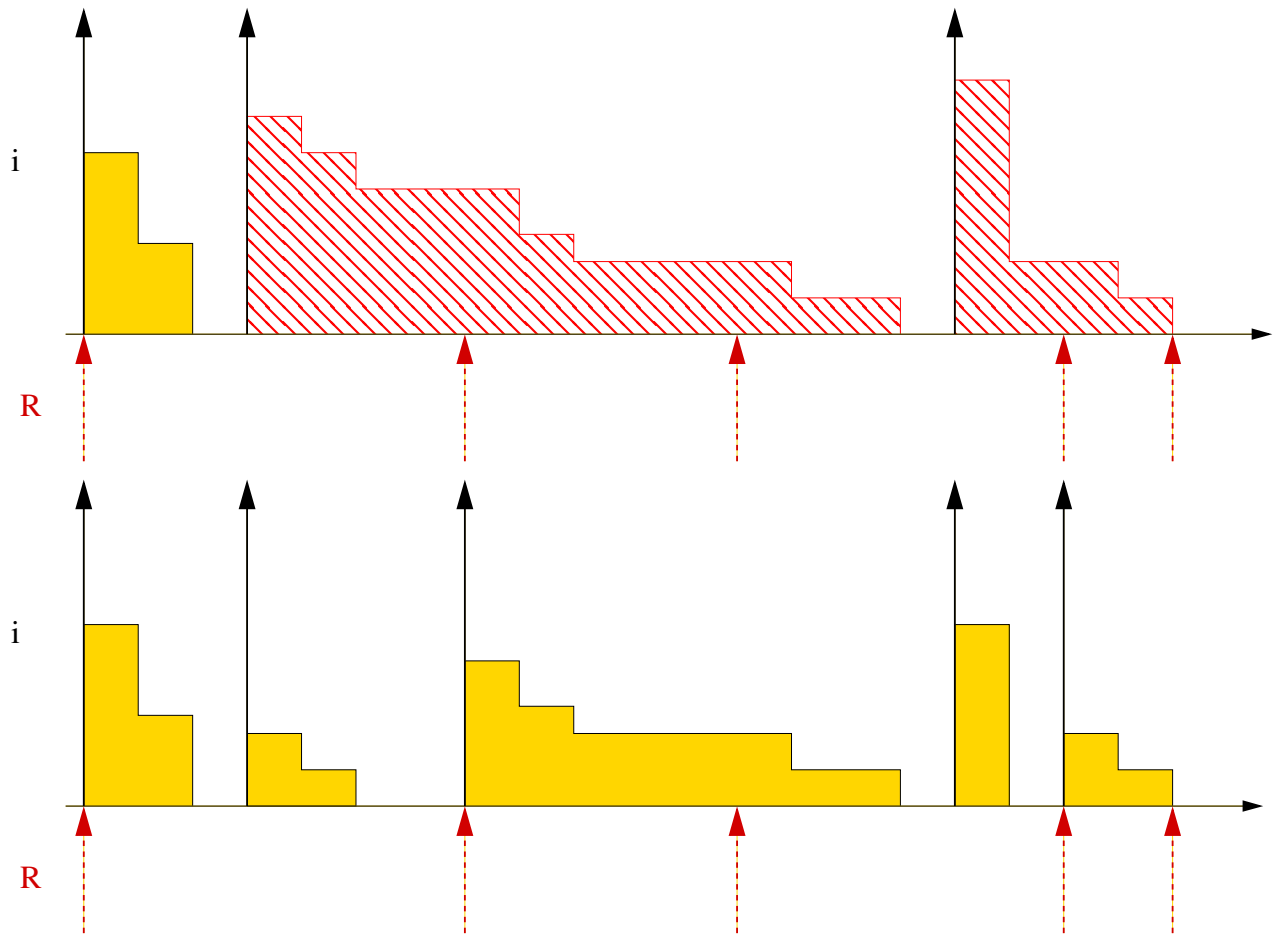
Aucun stock n'est conservé à l'entrepôt

Mais si les politiques locales π sont croisées ?

Comment rendre une politique non-croisée ?

Si deux commandes consécutives s, s' sont croisées avec les commandes r, r' à l'entrepôt :

On ajoute une commande chez le détaillant à la période r'



Reconstruction d'une politique réalisable

Résultat

A partir de politiques locales π_i réalisables pour les problèmes mono-stock (S_i), une politique π^u pour le problème multi-détaillant peut être construite, de coût au plus

$$\mathcal{C}(\pi^u) \quad \leq \quad \sum_{i=0}^{N} \mathcal{C}(\pi_i) + \sum_{i=1}^{N} \mathcal{K}(\pi_i)$$

Le surcoût à payer pour être réalisable est au plus le coût de commande aux détaillants dans leur politique locale.

Pour faire le point

Qu'avons nous ?

- Un algorithme UnCrossing construisant une politique réalisable à partir des politiques locales de la décomposition en problèmes de dimensionnement de lots.
- Avec un faible surcoût (coût total de commande aux détaillants)

De quelles politiques partir ?

- Le plus naturel : les politiques optimales π_i^* de chaque problème de dimensionnement de lots
- Pouvons-nous garantir notre algorithme ?

Garantie de performance

Résultat

L'algorithme UnCrossing partant des politiques locales optimales (π_i^*) est une 3-approximation

Garantie de performance

Résultat

L'algorithme UnCrossing partant des politiques locales optimales (π_i^*) est une 3-approximation

- Résultat vrai pour des coûts de possession très généraux

Garantie de performance

Résultat

L'algorithme UnCrossing partant des politiques locales optimales (π_i^*) est une 3-approximation

- Résultat vrai pour des coûts de possession très généraux
- Mais partir des politiques optimales n'est certainement pas le meilleur choix

Garantie de performance

Résultat

L'algorithme UnCrossing partant des politiques locales optimales (π_i^*) est une 3-approximation

- Résultat vrai pour des coûts de possession très généraux
- Mais partir des politiques optimales n'est certainement pas le meilleur choix
- La recherche d'une meilleure garantie de performance nous a conduit à choisir d'autres politiques

Split & Uncross

Idée : répartir le coût de possession d'un détaillant entre son système local et celui de l'entrepôt.

- Décomposer en problèmes mono-stock ($\bar{S}_i$) **en divisant par** 2 les coûts de possession

Split & Uncross

Idée : répartir le coût de possession d'un détaillant entre son système local et celui de l'entrepôt.

- Décomposer en problèmes mono-stock ($\bar{S}_i$) **en divisant par** 2 les coûts de possession
- Chercher les politiques optimales $\bar{\pi}_i^*$

Split & Uncross

Idée : répartir le coût de possession d'un détaillant entre son système local et celui de l'entrepôt.

- Décomposer en problèmes mono-stock $(\bar{S}_i)$ **en divisant par** 2 les coûts de possession
- Chercher les politiques optimales $\bar{\pi}_i^*$
- Appliquer l'algorithme UnCrossing aux politiques $\bar{\pi}_i^*$ pour déterminer une politique réalisable π^u

$$\mathcal{C}(\pi^u) \leq 2 \sum_{i=0}^{N} \mathcal{C}(\bar{\pi}_i^*)$$

Une nouvelle borne inférieure

Théorème

Pour des coûts de possession linéaires, le coût total des politiques locales $\bar{\pi}_i^*$ est une borne inférieure de l'optimum pour OWMR

$$\sum_{i=0}^{N} \mathcal{C}(\bar{\pi}_i^*) \leq OPT$$

Une nouvelle borne inférieure

Théorème

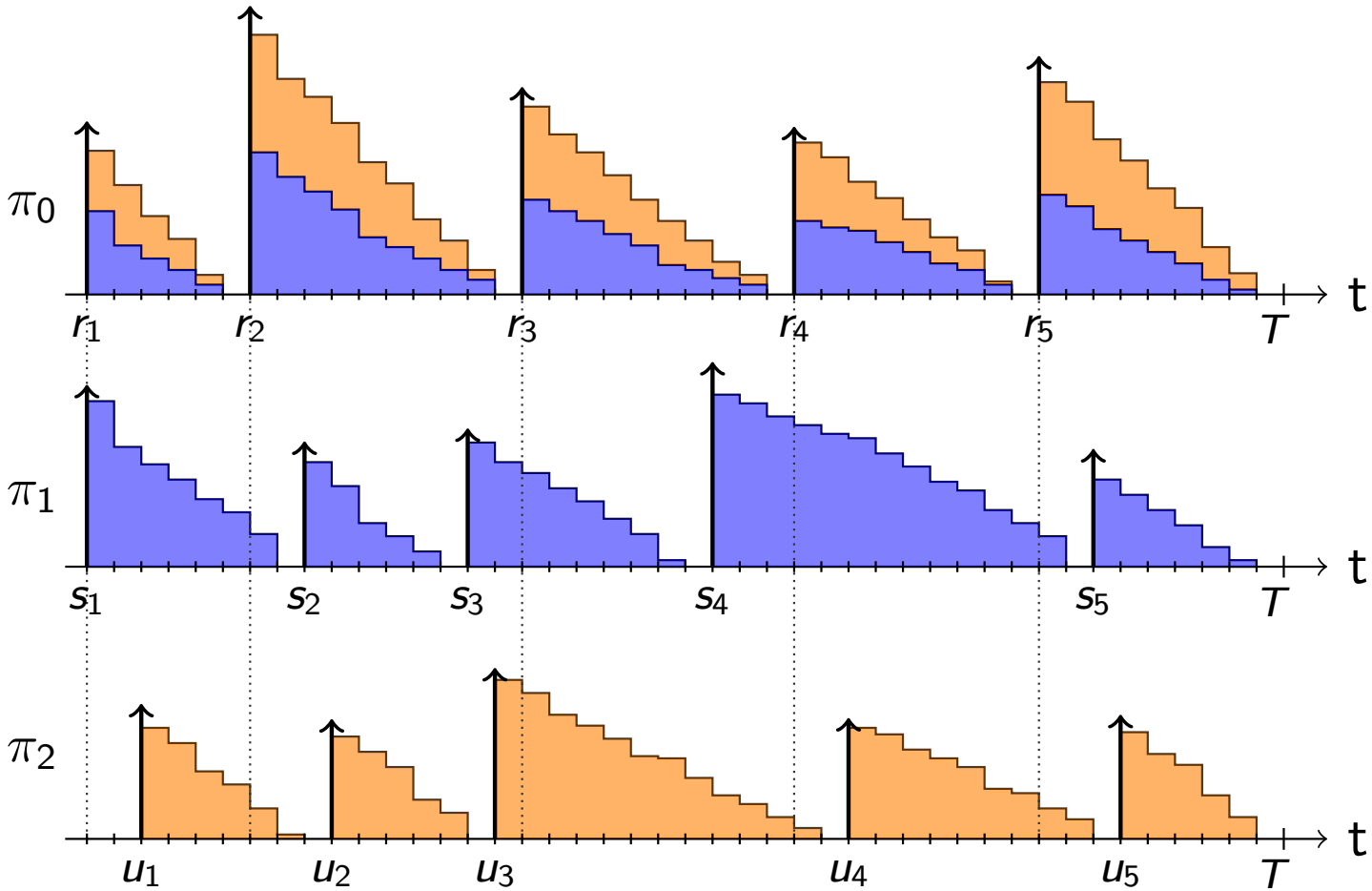
Pour des coûts de possession linéaires, le coût total des politiques locales $\bar{\pi}_i^*$ est une borne inférieure de l'optimum pour OWMR

$$\sum_{i=0}^{N} \mathcal{C}(\bar{\pi}_i^*) \leq OPT$$

Résultat

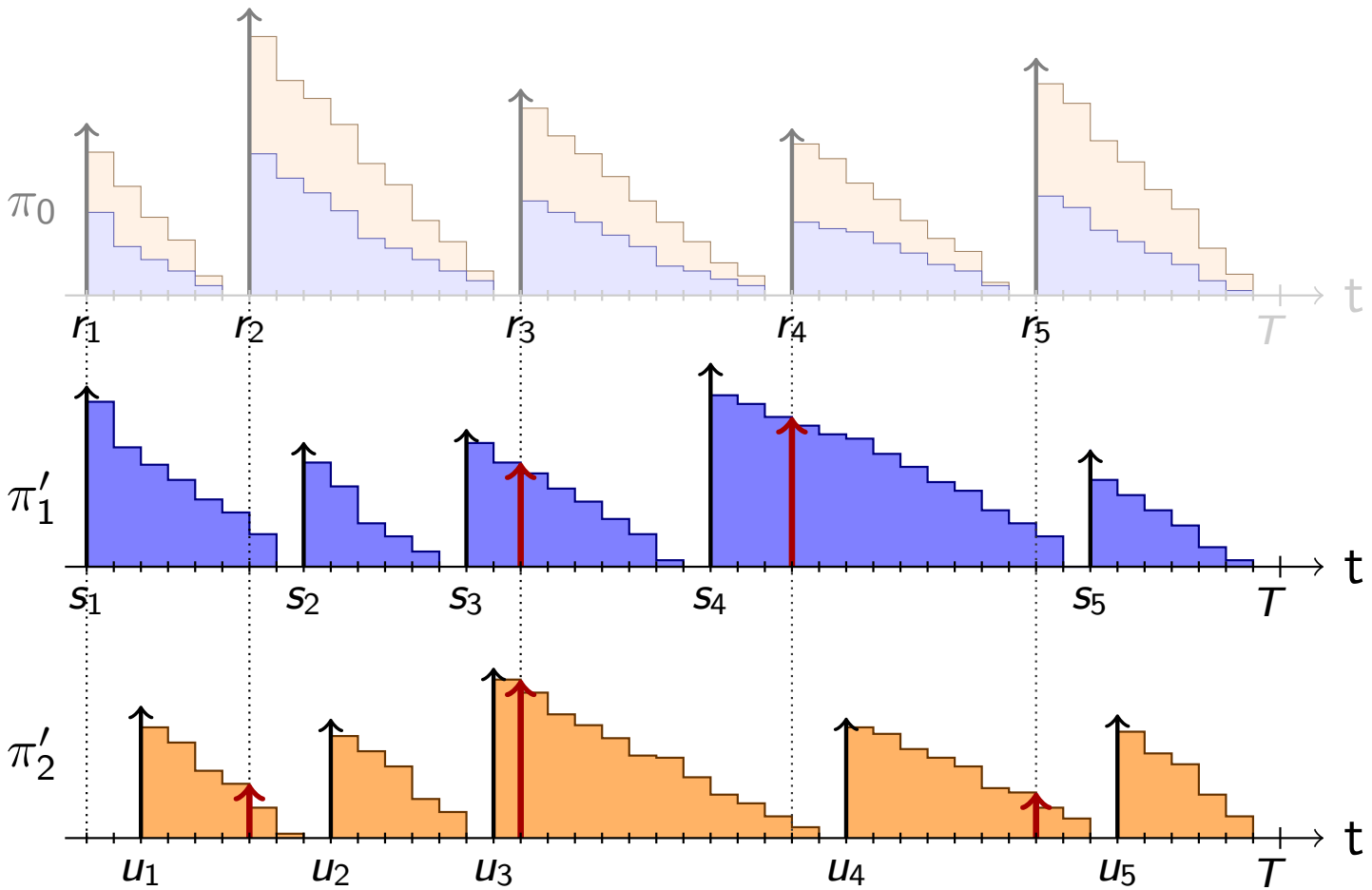
Split & Uncross est un algorithme de garantie 2 et de complexité dominée par la résolution des problèmes de lot-sizing.

Un exemple



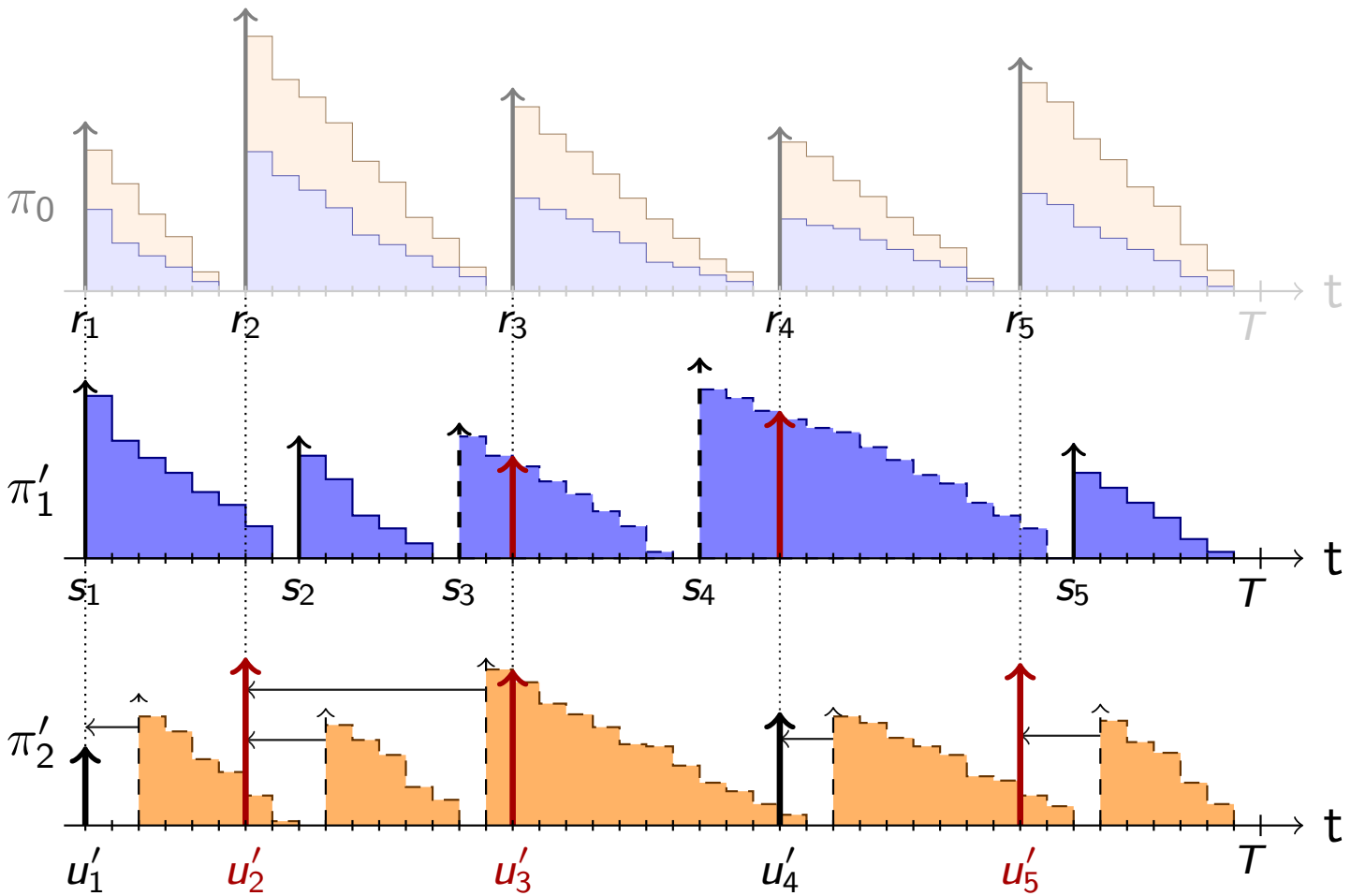
Déterminer les politiques ZIO optimales pour chaque problème de lot-sizing $(\bar{S}_i)$ avec les coûts de possession divisés par 2

Un exemple (2)



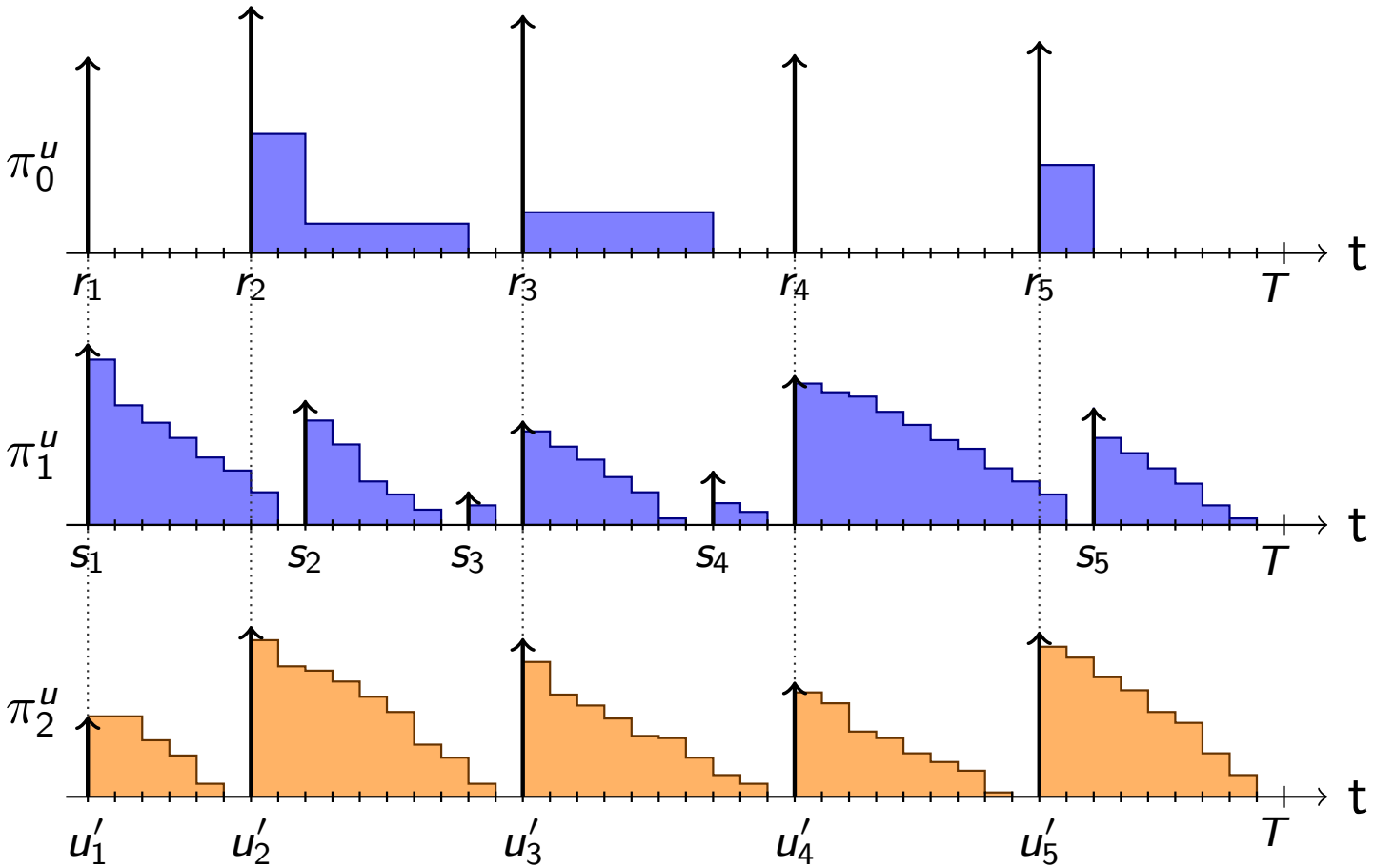
Décroiser les commandes aux détaillants

Un exemple (3)



Synchroniser le J-détaillants (2)

Un exemple (4)



Ajuster les commandes à l'entrepôt pour être ZIO

Généralisation

On aimerait pouvoir traiter des structures de coût plus générales :

- Coût de possession h non linéaire
  $h(x) = a\sqrt{x} + b$

Généralisation

On aimerait pouvoir traiter des structures de coût plus générales :

- Coût de possession h non linéaire
  $h(x) = a\sqrt{x} + b$
- Coût de commande incluant un coût par camion utilisé
  batch delivery

Généralisation

On aimerait pouvoir traiter des structures de coût plus générales :

- Coût de possession h non linéaire
  $h(x) = a\sqrt{x} + b$
- Coût de commande incluant un coût par camion utilisé
  batch delivery
- Des limites sur la taille des commandes
  capacitated lot-sizing

Extension de Split & Uncross

Pour étendre l'algorithme Split & Uncross il faut :

1. Savoir résoudre à l'optimum les problèmes de lot-sizing de la décomposition
2. Pouvoir borner le coût de la politique fournie par l'algorithme unCrossing

$$\mathcal{C}(\pi^u) \leq 2 \sum_{i=0}^{N} \mathcal{C}(\bar{\pi}_i^*)$$

3. Conserver la borne inférieure de la décomposiiton

$$\sum_{i=0}^{N} \mathcal{C}(\bar{\pi}_i^*) \leq OPT$$

Coûts de possession non linéaires

L'algorithme reste garanti à 2 pour des coûts de possession non linéaires :

- croissant avec le niveau de stock
- sous-additif à l'entrepôt

Coûts de possession non linéaires

L'algorithme reste garanti à 2 pour des coûts de possession non linéaires :

- croissant avec le niveau de stock
- sous-additif à l'entrepôt

Définition

Une fonction h est sous-additive ssi $h(x+y) \leq h(x) + h(y)$

Coûts de possession non linéaires

L'algorithme reste garanti à 2 pour des coûts de possession non linéaires :

- croissant avec le niveau de stock
- sous-additif à l'entrepôt

Définition

Une fonction h est sous-additive ssi $h(x+y) \leq h(x) + h(y)$

- Proche de la concavité, modélise les économies d'échelle

Coûts de possession non linéaires

L'algorithme reste garanti à 2 pour des coûts de possession non linéaires :

- croissant avec le niveau de stock
- sous-additif à l'entrepôt

Définition

Une fonction h est sous-additive ssi $h(x + y) \leq h(x) + h(y)$

- Proche de la concavité, modélise les économies d'échelle
- La somme de 2 fonctions sous-additives est sous-additive

Coûts de possession non linéaires

L'algorithme reste garanti à 2 pour des coûts de possession non linéaires :

- croissant avec le niveau de stock
- sous-additif à l'entrepôt

Définition

Une fonction h est sous-additive ssi $h(x+y) \leq h(x) + h(y)$

- Proche de la concavité, modélise les économies d'échelle
- La somme de 2 fonctions sous-additives est sous-additive
- Cas d'un stockage avec un coût fixe par rack.

Coûts de possession non linéaires

L'algorithme reste garanti à 2 pour des coûts de possession non linéaires :

- croissant avec le niveau de stock
- sous-additif à l'entrepôt

Définition

Une fonction h est sous-additive ssi $h(x+y) \leq h(x) + h(y)$

- Proche de la concavité, modélise les économies d'échelle
- La somme de 2 fonctions sous-additives est sous-additive
- Cas d'un stockage avec un coût fixe par rack.
- $h_t(x) = \lceil x/R \rceil c_t + a_t\sqrt{x} + b_t$

Extension de Split & Uncross

On définit exactement la même décomposition

1. Résoudre à l'optimum les problèmes de lot-sizing. OK
2. Borner le coût de la politique fournie par l'algorithme unCrossing. OK
3. **Conserver la borne inférieure de la décomposiiton ?**

$$\sum_{i=0}^{N} \mathcal{C}(\bar{\pi}_i^*) \leq OPT$$

Idée de la preuve

On se limite au cas avec uniquement des $W-$détaillants.

Idée de la preuve

On se limite au cas avec uniquement des $W-$détaillants.
Considérons une politique optimale π^{OPT} pour OWMR.

On définit les politiques $\tilde{\pi}_i$ pour chaque problème $(\bar{S}_i)$:

- Pour chaque détaillant, $\tilde{\pi}_i = \pi_i^{\mathrm{OPT}}$. Elle commande les mêmes quantités que l'optimal.
- Pour l'entrepôt, $\tilde{\pi}_0$ commande aux mêmes périodes que π^{OPT}. A une période r elle commande toute la demande des détaillants jusqu'à la prochaine période de commande.

Idée de la preuve

On se limite au cas avec uniquement des $W-$détaillants.
Considérons une politique optimale π^{OPT} pour OWMR.

On définit les politiques $\tilde{\pi}_i$ pour chaque problème $(\bar{S}_i)$:

- Pour chaque détaillant, $\tilde{\pi}_i = \pi_i^{\mathrm{OPT}}$. Elle commande les mêmes quantités que l'optimal.
- Pour l'entrepôt, $\tilde{\pi}_0$ commande aux mêmes périodes que π^{OPT}. A une période r elle commande toute la demande des détaillants jusqu'à la prochaine période de commande.

On veut établir que $\sum_{i=0}^{N} \mathcal{C}(\tilde{\pi}_i) \ \leq \ OPT$

Idée de la preuve (2)

- Les politiques commandes aux mêmes périodes
  Le coût total de commande est le même
- Pour un détaillant, les 2 politiques sont identiques
  Le coût de possession payé par $\tilde{\pi}_i$ est la moitié de π_i^{OPT}
- Quel sont les coûts de possession à l'entrepôt ?

Idée de la preuve (2)

- Les politiques commandes aux mêmes périodes
  Le coût total de commande est le même
- Pour un détaillant, les 2 politiques sont identiques
  Le coût de possession payé par $\tilde{\pi}_i$ est la moitié de π_i^{OPT}
- Quel sont les coûts de possession à l'entrepôt ?

Définissons les niveaux de stock :

- x_{it}^{OPT} : le niveau de stock à i à la fin de la période t dans π^{OPT}
- $x_t^e \equiv x_{0t}^{\text{OPT}} + \sum_{i \in I_W} x_{it}^{\text{OPT}}$: le stock échelon
- $\tilde{y}_t$: le niveau de stock dans le problème à l'entrepôt dans $\tilde{\pi}_0$

Idée de la preuve (3)

$$
\begin{aligned}
\tfrac{1}{2} h_t^0(\tilde{y}_t) &\leq \frac{1}{2} h_t^0(x_t^e) \\
&\leq \frac{1}{2} h_t^0(x_{0t}^{\mathrm{OPT}}) + \frac{1}{2} \sum_{i \in I_W} h_t^0(x_{it}^{\mathrm{OPT}}) \\
&\leq \frac{1}{2} h_t^0(x_{0t}^{\mathrm{OPT}}) + \frac{1}{2} \sum_{i \in I_W} h_t^i(x_{it}^{\mathrm{OPT}})
\end{aligned}
$$

Idée de la preuve (3)

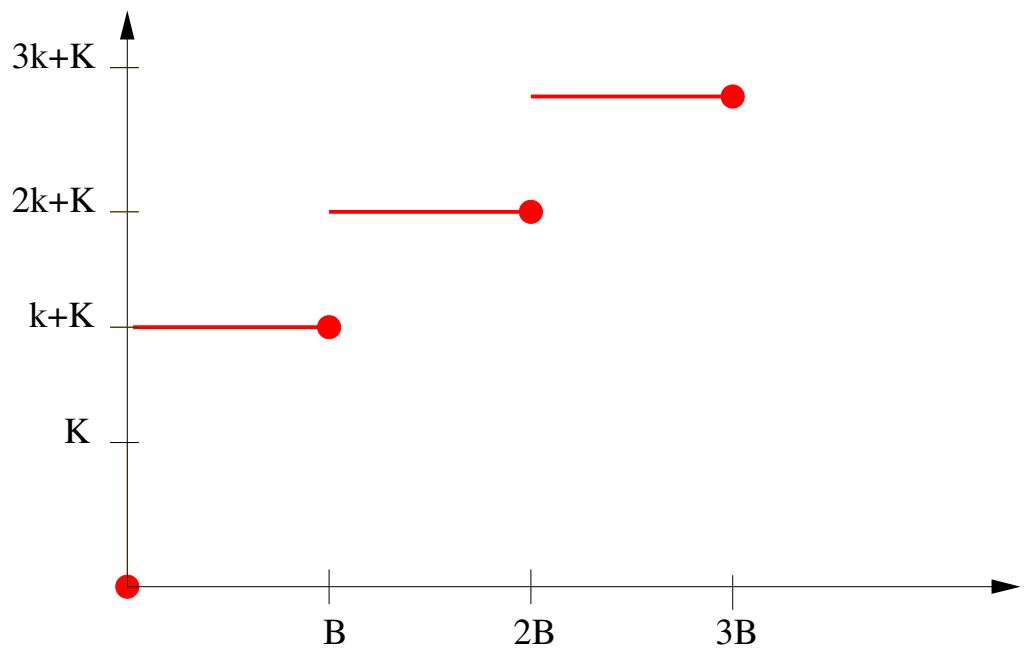
$$\begin{aligned}
\tfrac{1}{2} h_t^0(\tilde{y}_t) &\leq \frac{1}{2} h_t^0(x_t^e) \\
&\leq \frac{1}{2} h_t^0(x_{0t}^{\mathrm{OPT}}) + \frac{1}{2} \sum_{i \in I_W} h_t^0(x_{it}^{\mathrm{OPT}}) \\
&\leq \frac{1}{2} h_t^0(x_{0t}^{\mathrm{OPT}}) + \frac{1}{2} \sum_{i \in I_W} h_t^i(x_{it}^{\mathrm{OPT}})
\end{aligned}$$

Le problème à l'entrepôt paie en coût de possession la moitié des coûts de possession de l'optimal, ce qui établit la borne inférieure.

Coûts de commande FTL

- Modélise la livraison par camion, de capacité B
- En plus d'un coût de commande fixe K, un coût k est payé par camion

$$p(x) = K + \lceil x/B \rceil k$$



Lot-sizing avec coût FTL

- Le problème de lot-sizing mono-item $(\bar{S}_i)$ pour chaque détaillant peut être résolu en temps $\mathcal{O}(T^3)$ [Li *et al* Op. Research 2004]
- Mais le problème multi-item $(\bar{S}_0)$ à l'entrepôt ?
- ZIO n'est plus une propriété dominante avec des coûts FTL Algorithme UnCrossing reste valide ?

Nous allons utiliser 2 ingrédients :

1. Les politiques PCO
2. Les côuts sandwich

Algorithme UNCROSSING

L'algorithme UNCROSSING fonctionne avec des politiques plus générales que des politiques ZIO :

Politique PCO

Une politique est *PCO* (Positive Consumption Ordering) si pour chaque commande, au moins une unité est consommée à la période de commande.

Les politiques PCO sont dominantes si les coûts de commandes sont stationnaires.

Coûts sandwich

Linearly sandwiched

Le coût de commande $p_t()$ est $\lambda-$*linearly sandwiched* si pour toute période

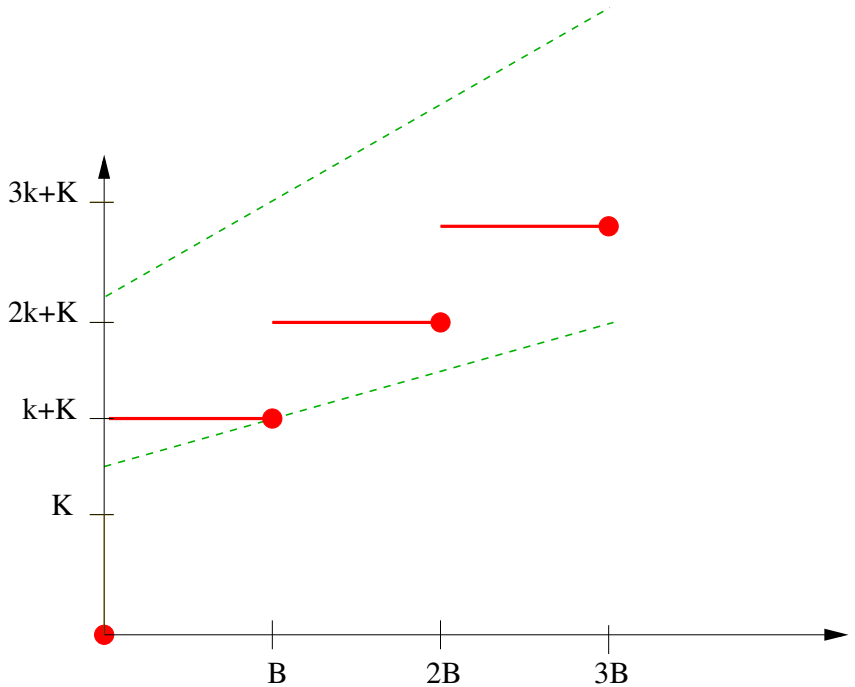
$$A_t + bx \leq p_t(x) \leq \lambda(A_t + bx) \quad \forall x > 0$$

avec $(A_t)_{t=1,\ldots,T}$ et b des constantes.

Coûts sandwich

Propriété

Un coût de commande FTL est 2−linearly sandwiched.



$$A_t = K_t + \frac{k}{2} \quad \text{et} \quad b = \frac{k}{2B}$$

Coûts sandwich

Quel est l'intérêt des coûts de commande $\lambda-$linearly sandwiched ?

Propriété

Soient π et σ des politiques commandant respectivement x_t et z_t telles que

$$x_t = 0 \;\Rightarrow\; z_t = 0$$

Alors le coût total de commande de σ est au plus λ celui de π.

Coûts sandwich

Quel est l'intérêt des coûts de commande $\lambda-$linearly sandwiched ?

Propriété

Soient π et σ des politiques commandant respectivement x_t et z_t telles que

$$x_t = 0 \;\Rightarrow\; z_t = 0$$

Alors le coût total de commande de σ est au plus λ celui de π.

Coûts sandwich

Quel est l'intérêt des coûts de commande $\lambda-$linearly sandwiched ?

Propriété

Soient π et σ des politiques commandant respectivement x_t et z_t telles que

$$x_t = 0 \;\Rightarrow\; z_t = 0$$

Alors le coût total de commande de σ est au plus λ celui de π.

- Soient X et Z les périodes où les politiques commandent

Coûts sandwich

Quel est l'intérêt des coûts de commande $\lambda-$linearly sandwiched ?

Propriété

Soient π et σ des politiques commandant respectivement x_t et z_t telles que

$$x_t = 0 \;\Rightarrow\; z_t = 0$$

Alors le coût total de commande de σ est au plus λ celui de π.

- Soient X et Z les périodes où les politiques commandent
- Pour π : $\sum_t p_t(x_t) \geq \sum_{t \in X} A_t + b \sum_t x_t$

Coûts sandwich

Quel est l'intérêt des coûts de commande $\lambda-$linearly sandwiched ?

Propriété

Soient π et σ des politiques commandant respectivement x_t et z_t telles que

$$x_t = 0 \;\Rightarrow\; z_t = 0$$

Alors le coût total de commande de σ est au plus λ celui de π.

- Soient X et Z les périodes où les politiques commandent
- Pour π : $\sum_t p_t(x_t) \geq \sum_{t\in X} A_t + b\sum_t x_t$
- Pour σ : $\sum_t p_t(z_t) \leq \lambda(\sum_{t\in Z} A_t + b\sum_t x_t)$

Coûts sandwich

Quel est l'intérêt des coûts de commande $\lambda-$linearly sandwiched ?

Propriété

Soient π et σ des politiques commandant respectivement x_t et z_t telles que

$$x_t = 0 \;\Rightarrow\; z_t = 0$$

Alors le coût total de commande de σ est au plus λ celui de π.

- Soient X et Z les périodes où les politiques commandent
- Pour π : $\sum_t p_t(x_t) \geq \sum_{t\in X} A_t + b\sum_t x_t$
- Pour σ : $\sum_t p_t(z_t) \leq \lambda(\sum_{t\in Z} A_t + b\sum_t x_t)$
- On a $\sum_t x_t = \sum_t z_t$ dans une politique dominante

Coûts sandwich

Quel est l'intérêt des coûts de commande $\lambda-$linearly sandwiched ?

Propriété

Soient π et σ des politiques commandant respectivement x_t et z_t telles que

$$x_t = 0 \;\Rightarrow\; z_t = 0$$

Alors le coût total de commande de σ est au plus λ celui de π.

- Soient X et Z les périodes où les politiques commandent
- Pour π : $\sum_t p_t(x_t) \geq \sum_{t\in X} A_t + b\sum_t x_t$
- Pour σ : $\sum_t p_t(z_t) \leq \lambda(\sum_{t\in Z} A_t + b\sum_t x_t)$
- On a $\sum_t x_t = \sum_t z_t$ dans une politique dominante
- Et $Z \subseteq X$ par notre choix de σ

Décomposition à l'entrepôt

Le problème de lot-sizing FTL à l'entrepôt est relaxé en utilisant les coûts sandwich :

- Les coûts de possession sont divisés par 2
- Le coût de commande à la période t est le côut fixe A_t

Décomposition à l'entrepôt

Le problème de lot-sizing FTL à l'entrepôt est relaxé en utilisant les coûts sandwich :

- Les coûts de possession sont divisés par 2
- Le coût de commande à la période t est le côut fixe A_t

C'est un problème ULSP multi-item $(\bar{S}_0)$ comme précédemment !

- On peut trouver très efficacement une politique optimale $\bar{\pi}^*$
- La propriété précédente montre qu'en commandant aux mêmes instants que $\bar{\pi}^*$, on paie en coût de commande au plus 2 fois celui de $\bar{\pi}^*$.

$$\mathcal{C}(\pi^u) \leq 2 \sum_{i=0}^{N} \mathcal{C}(\bar{\pi}_i^*)$$

Conclusion

- Un nouvel algorithme Split & Uncross pour le problème OWMR
- Très rapide : résolution en en temps linéaire pour des coûts de possession linéaires,
- Très général : garantie de performance de 2 pour des coûts de possession non-linéaires (sous-additif à l'entrepôt), des coûts de commandes FTL/LTL, ...
- L'algorithme unCrossing repose sur des idées très simples : généralisation à d'autres problèmes multi-échelon (ventes perdues, différées, réseau de distribution...)

Un algorithme d'approximation pour le problème One-Warehouse Multi-Retailers

Jean-Philippe Gayon, Guillaume Massonnet, Christophe Rapine, Gautier Stauffer

JFRO - mars 2013