Algorithme d'approximation pour le problème *One Warehouse Multi-Retailer*

G. Massonnet[1], J.-P. Gayon[2], C. Rapine[3] et G. Stauffer[2]

[1]INRIA Grenoble Rhône-Alpes

[2]G-SCOP, Université Grenoble-Alpes

[3]LGPIM, Université de Lorraine

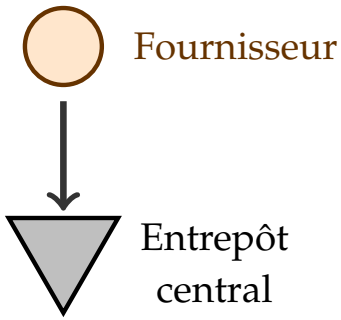
Journées STP du GDR MACS, 19 mai 2016

PLAN

PLAN DE LA PRÉSENTATION

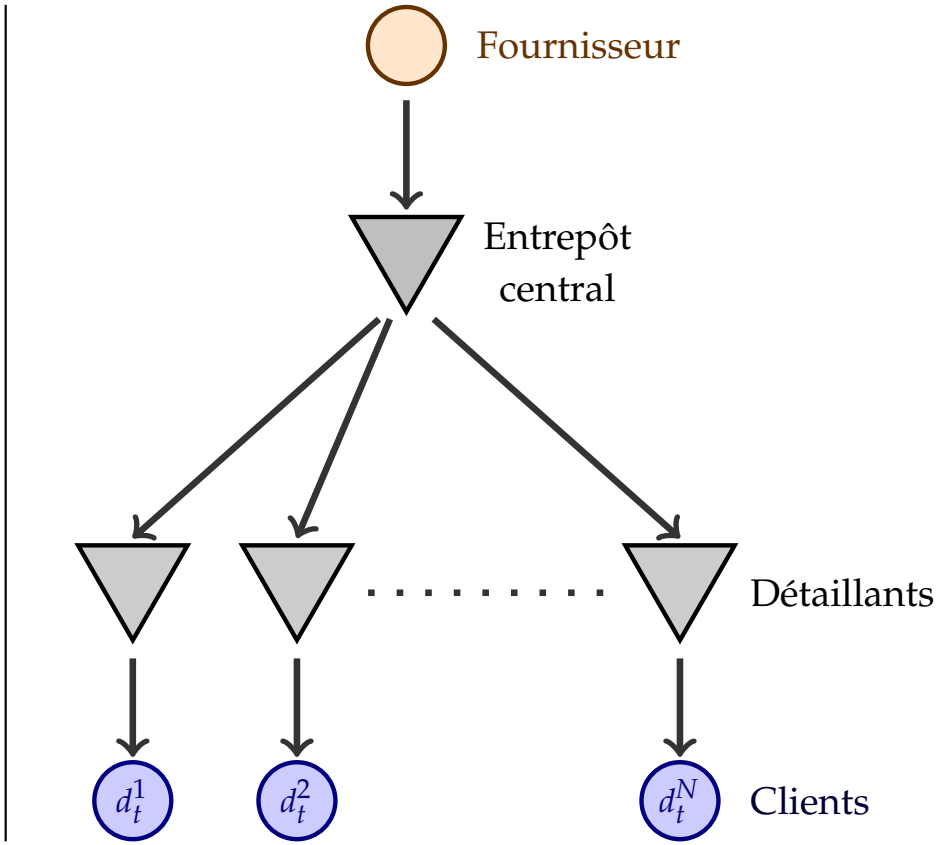
LE PROBLÈME *One-Warehouse Multi Retailers*

- Un entrepôt central



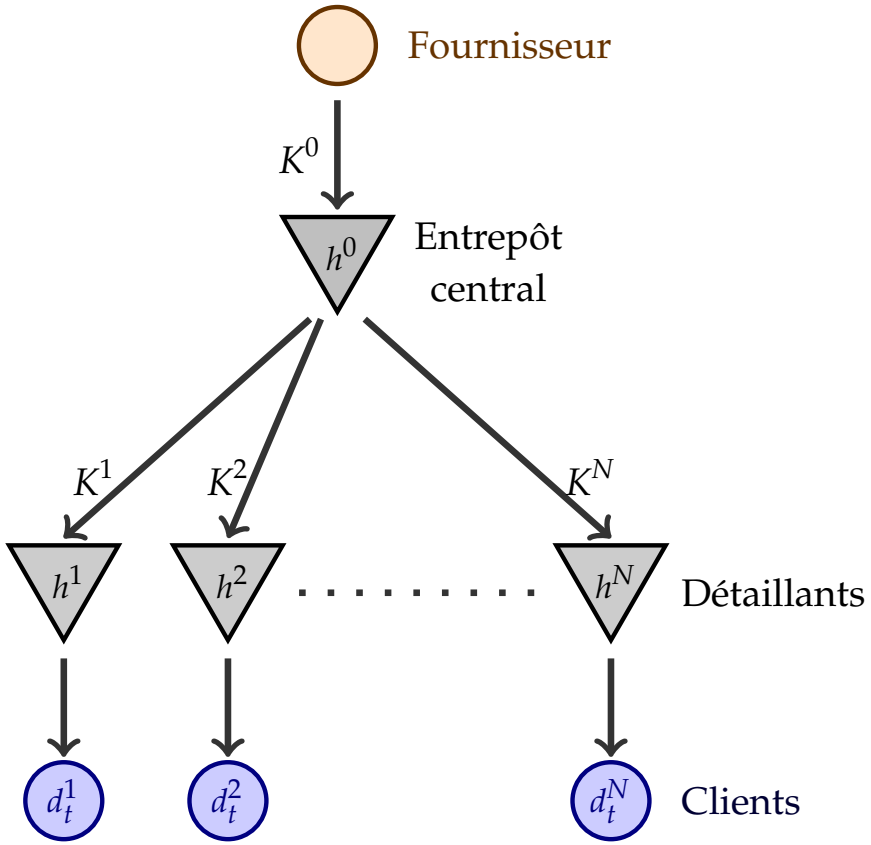
LE PROBLÈME *One-Warehouse Multi Retailers*

- Un entrepôt central
- N détaillants
- Horizon de temps T
- Demande pour chaque détaillant : d_t^i



LE PROBLÈME *One-Warehouse Multi Retailers*

- Un entrepôt central
- N détaillants
- Horizon de temps T
- Demande pour chaque détaillant : d_t^i
- Coûts fixes de commande : K^0, K^i
- Coûts unitaire de possession (par période) : $h^0, h^i \geq h^0$



Politique

Un ensemble de règles de réapprovisionnement du stock.

→ Quand commander ? Quelle quantité ?

Objectif

Satisfaire toute la demande des clients sur l'horizon de temps en minimisant les coûts encourus par le système.

→ Une solution *optimale* minimise les coûts.

Algorithmes d'approximation

Arkin et al. (1989), Chan et al. (2000)

Le problème OWMR discret est NP-difficile.

→ Algorithmes "brute force" : solution optimale, temps de calcul (très) long.

→ Heuristiques : solution arbitrairement mauvaise pour certaines instances, faible temps de calcul.

Algorithmes d'approximation

Arkin et al. (1989), Chan et al. (2000)

Le problème OWMR discret est NP-difficile.

→ Algorithmes "brute force" : solution optimale, temps de calcul (très) long.

→ Heuristiques : solution arbitrairement mauvaise pour certaines instances, faible temps de calcul.

Compromis : algorithmes d'approximation avec garantie de performance (constante)

Soit I une instance du problème considéré.

- C^* : coût d'une solution optimale
- C^{P} : coût de la solution trouvée par π

π a une garantie de performance α (ou est une α-approximation) ssi pour tout instance du problème

$$C^{\mathrm{P}} \leq \alpha C^*$$

Résultats existants

- ROUNDY (1985) : approximation pour le problème OWMR en temps continu.
- LEVI ET AL. (2006) : 2-approximation pour un cas particulier (JRP)
- LEVI ET AL. (2008) : 1.8-approximation pour OWMR basée sur une technique de rounding.
- NONNER & SOUZA (2009) : 5/3-approximation pour JRP avec deadlines.
- BIENKOWSKI ET AL. (2013) : 1.574-approximation JRP avec deadlines.
- BIENKOWSKI ET AL. (2014) : 1.791-approximation pour le JRP.

Résultats existants

- ROUNDY (1985) : approximation pour le problème OWMR en temps continu.
- LEVI ET AL. (2006) : 2-approximation pour un cas particulier (JRP)
- LEVI ET AL. (2008) : 1.8-approximation pour OWMR basée sur une technique de rounding.
- NONNER & SOUZA (2009) : 5/3-approximation pour JRP avec deadlines.
- BIENKOWSKI ET AL. (2013) : 1.574-approximation JRP avec deadlines.
- BIENKOWSKI ET AL. (2014) : 1.791-approximation pour le JRP.

Notre contribution

Un algorithme *combinatoire* simple et de faible complexité avec une garantie de 2 pour le OWMR.

PLAN DE LA PRÉSENTATION

DÉCOMPOSITION DU PROBLÈME

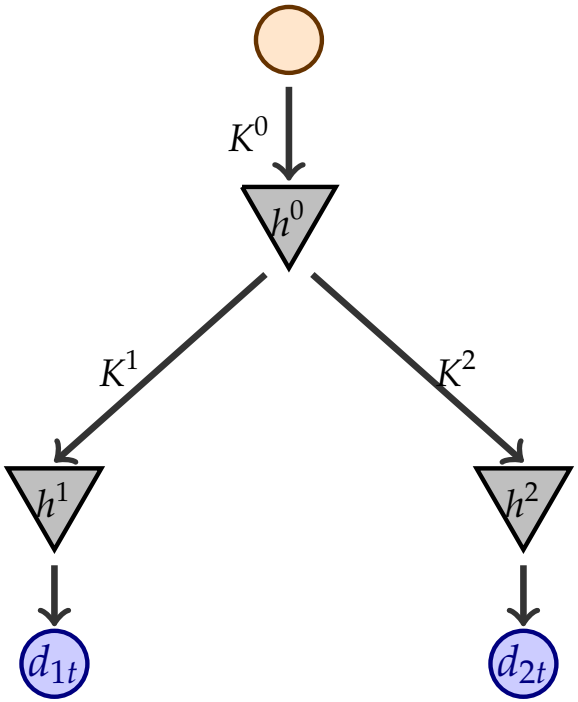


FIGURE – Point de vue "détaillants"

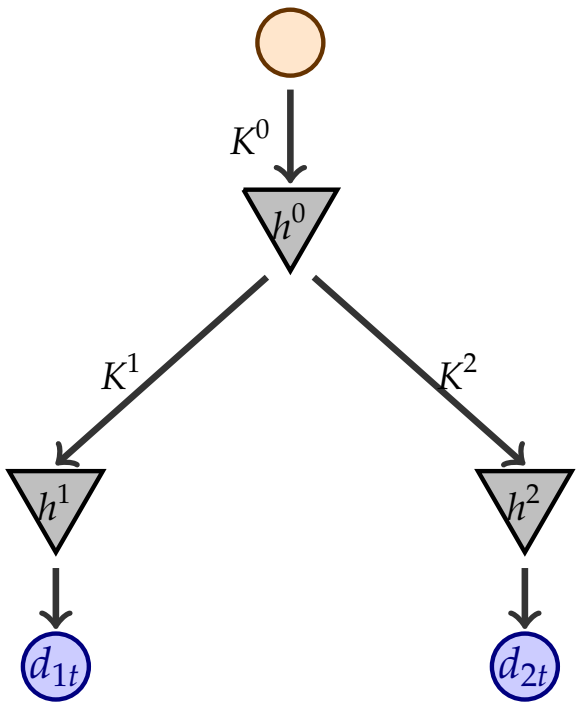


FIGURE – Point de vue "warehouse"

DÉCOMPOSITION DU PROBLÈME

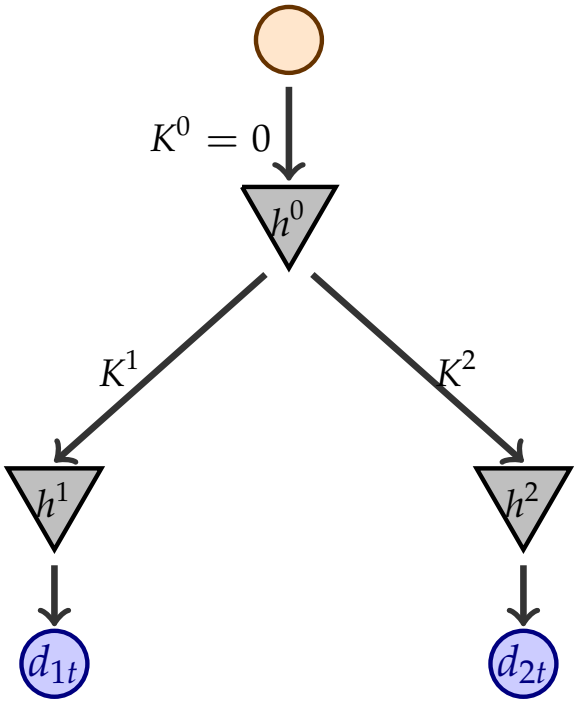


FIGURE – Point de vue "détaillants"

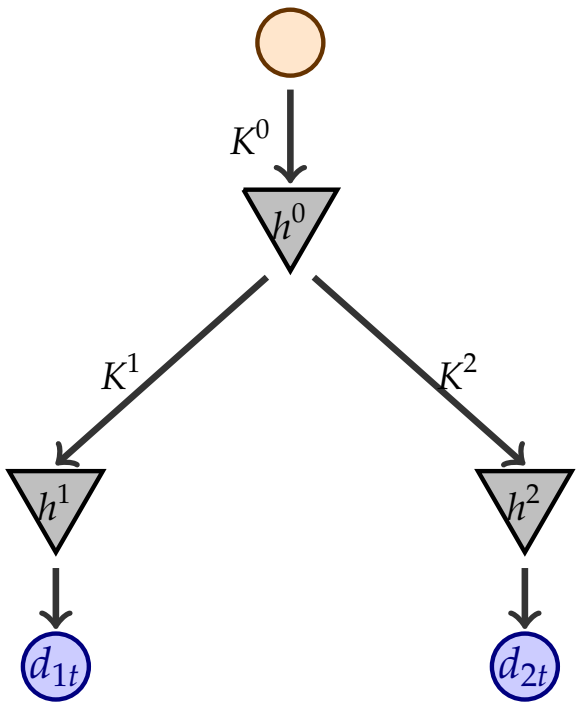


FIGURE – Point de vue "warehouse"

DÉCOMPOSITION DU PROBLÈME

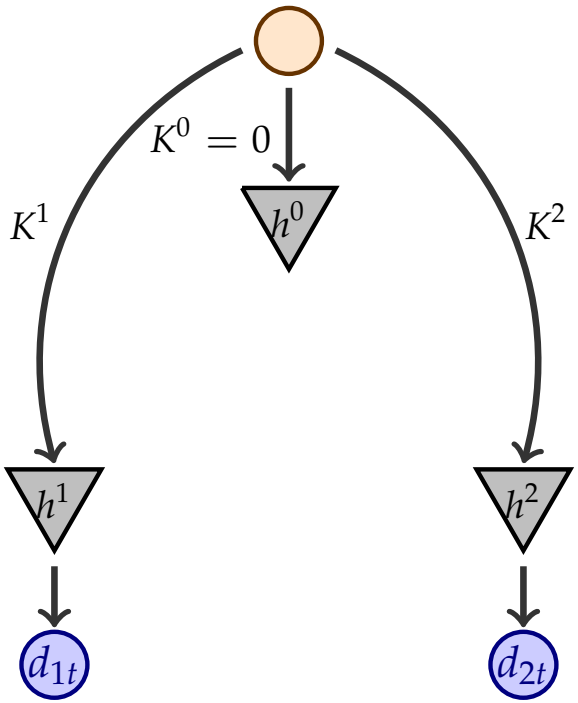


FIGURE – Point de vue "détaillants"

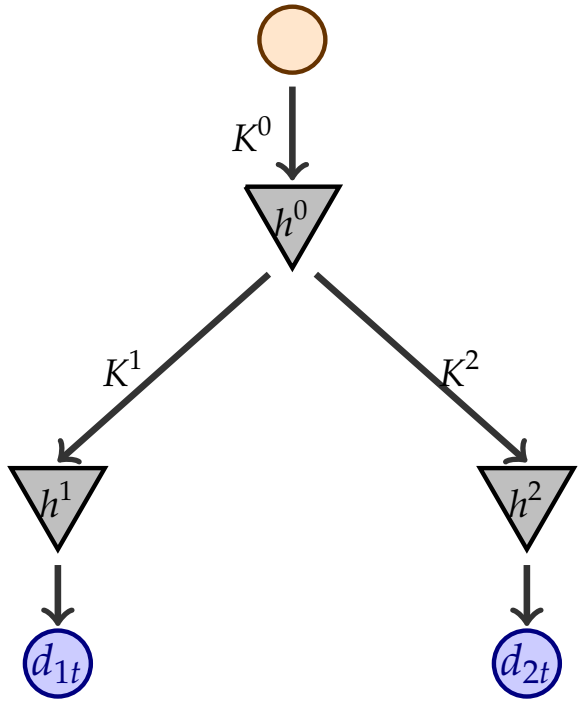


FIGURE – Point de vue "warehouse"

DÉCOMPOSITION DU PROBLÈME

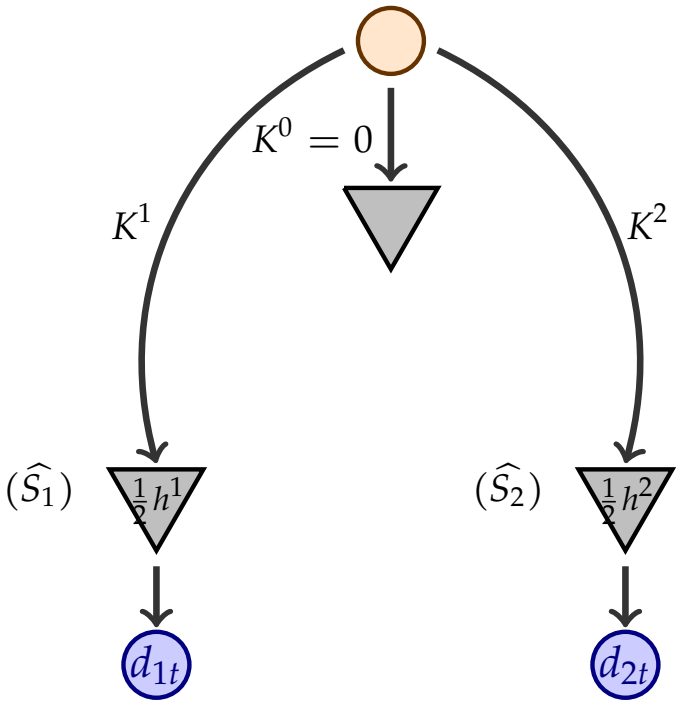


FIGURE – Point de vue "détaillants"

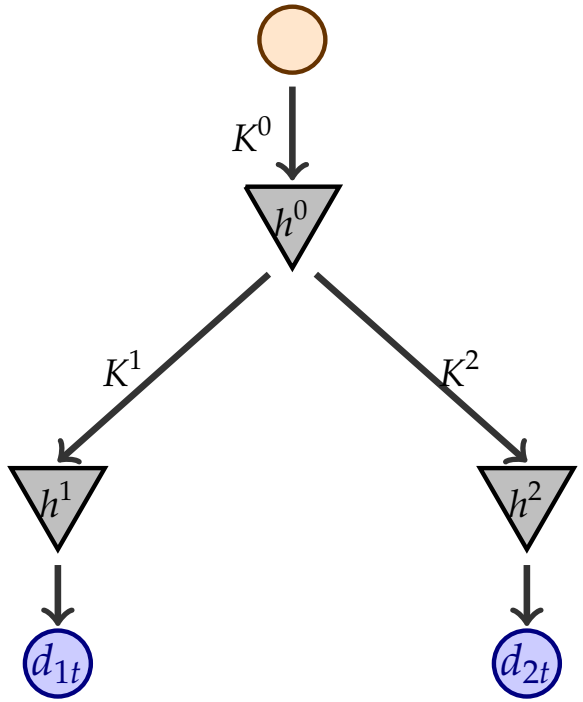


FIGURE – Point de vue "warehouse"

DÉCOMPOSITION DU PROBLÈME

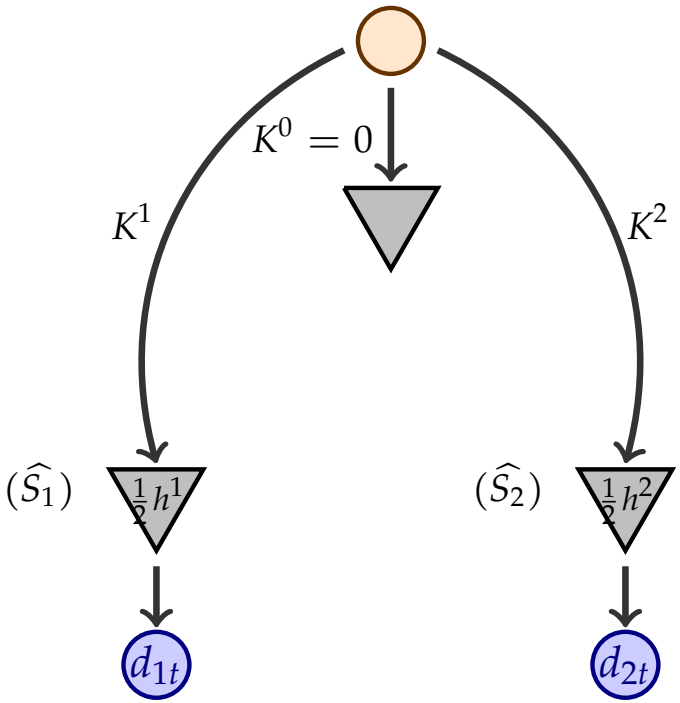


FIGURE – Point de vue "détaillants"

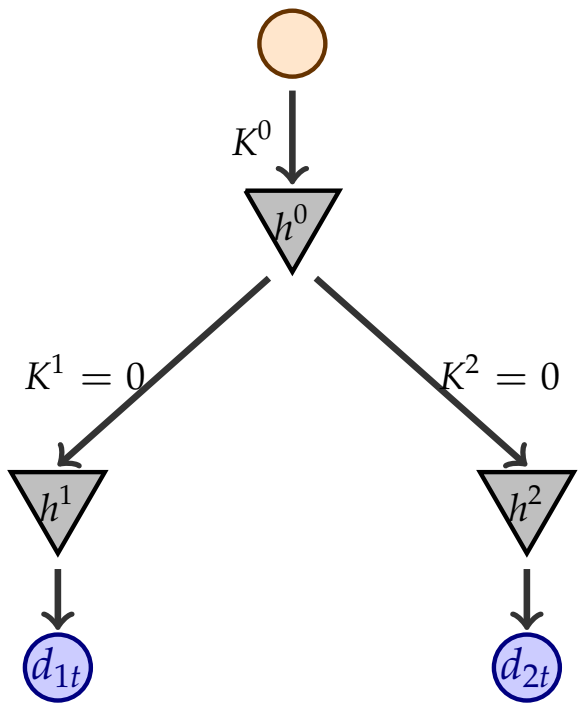


FIGURE – Point de vue "warehouse"

DÉCOMPOSITION DU PROBLÈME

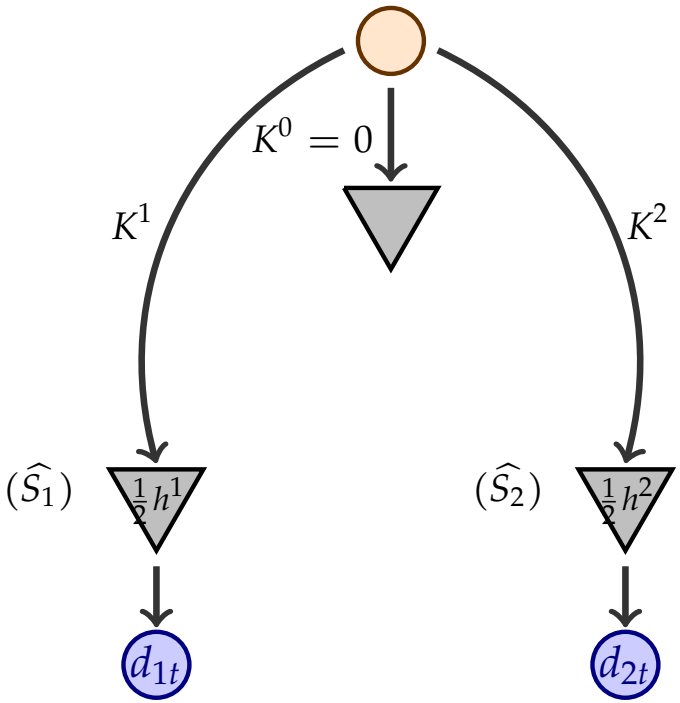


FIGURE – Point de vue "détaillants"

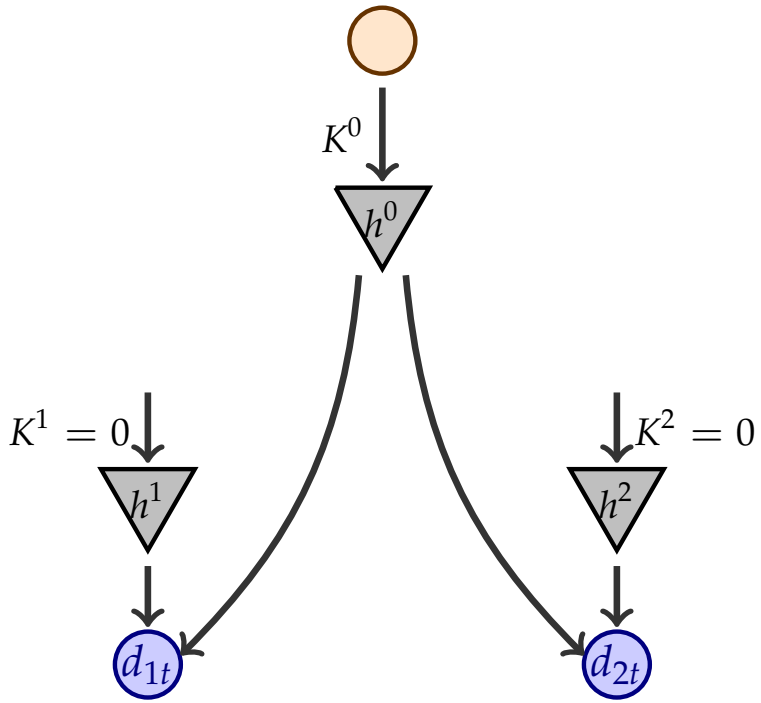


FIGURE – Point de vue "warehouse"

DÉCOMPOSITION DU PROBLÈME

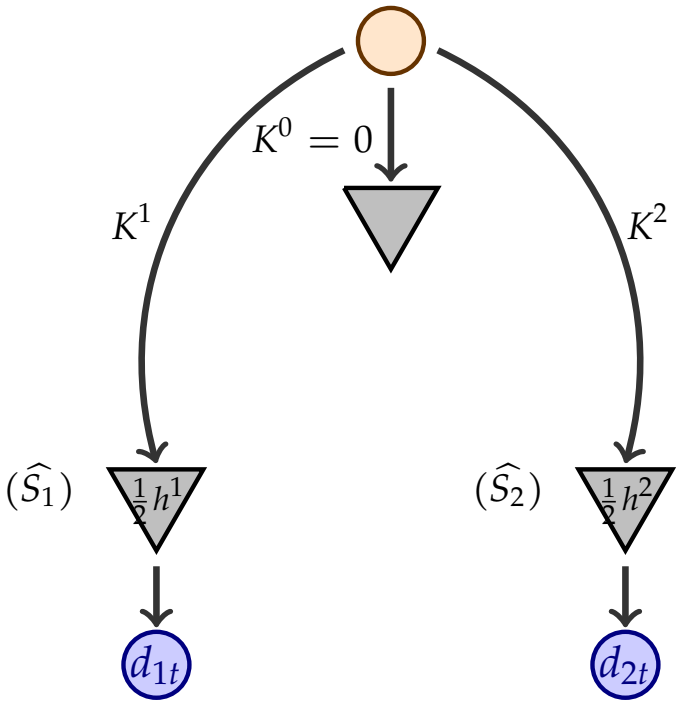


FIGURE – Point de vue "détaillants"

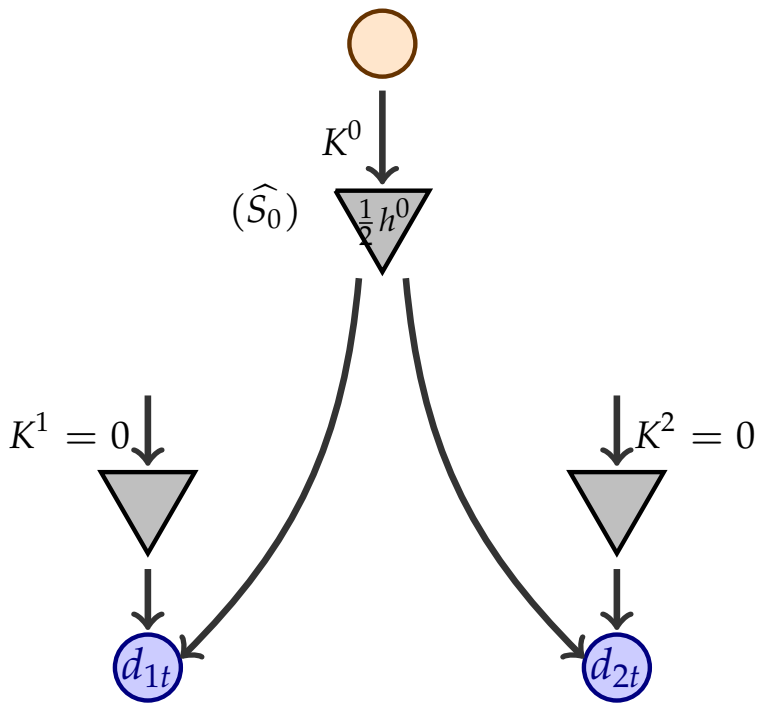


FIGURE – Point de vue "warehouse"

DÉCOMPOSITION DU PROBLÈME

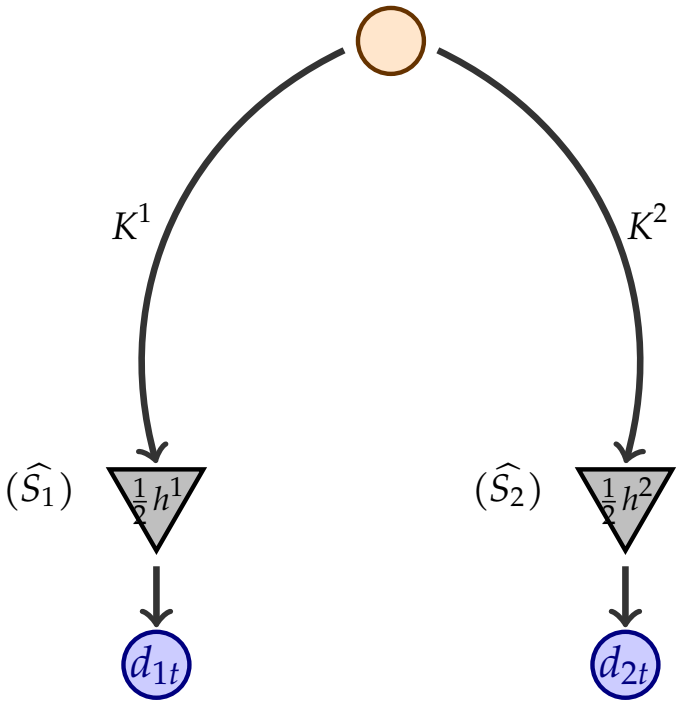


FIGURE – Point de vue "détaillants"

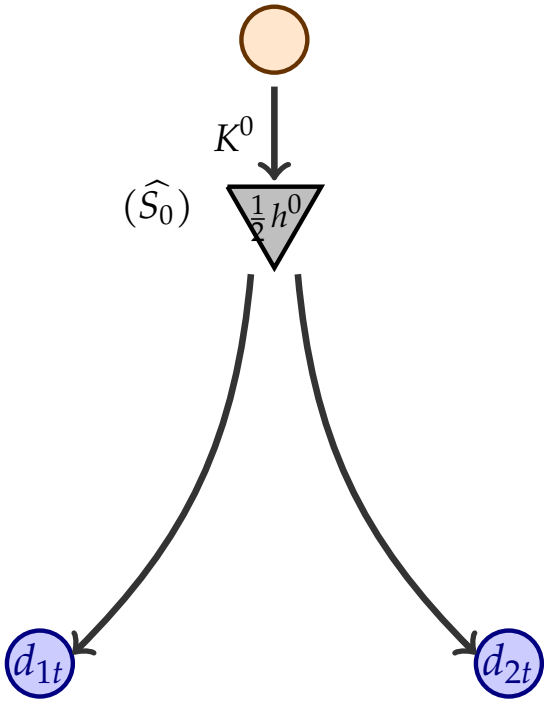
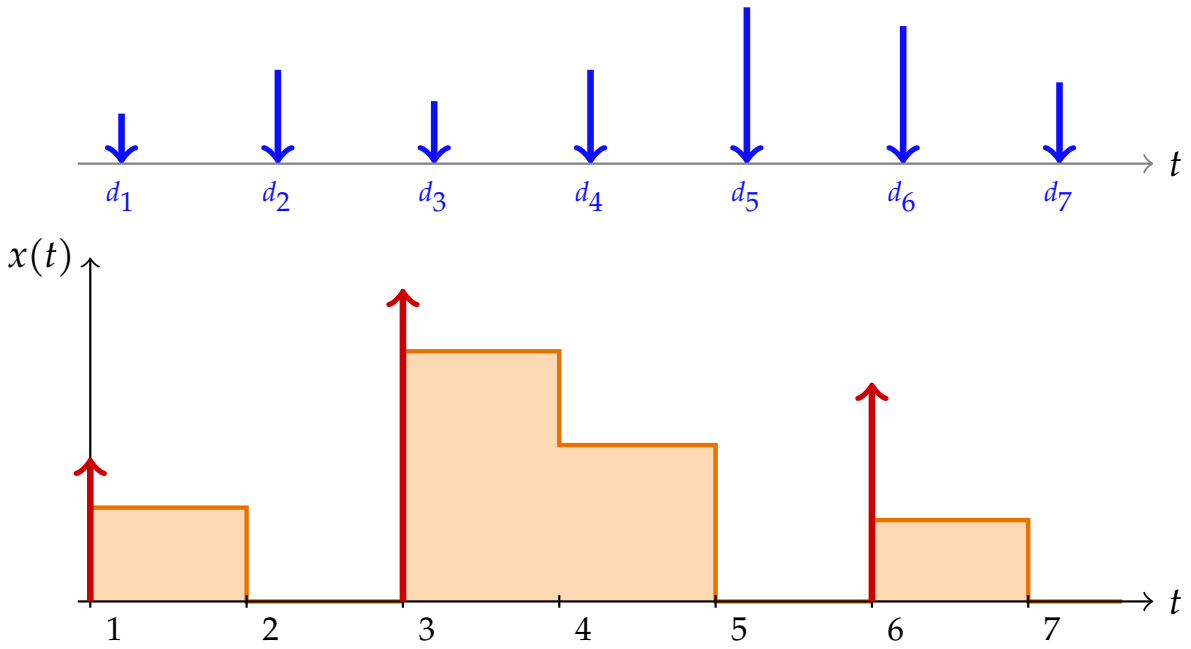
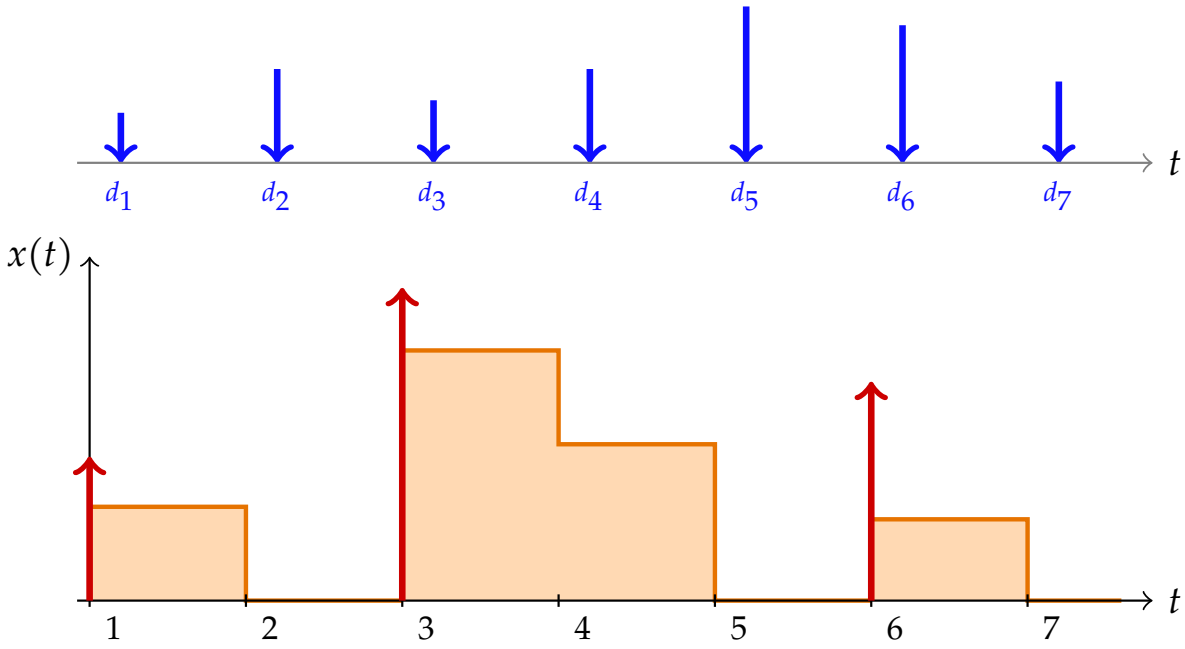


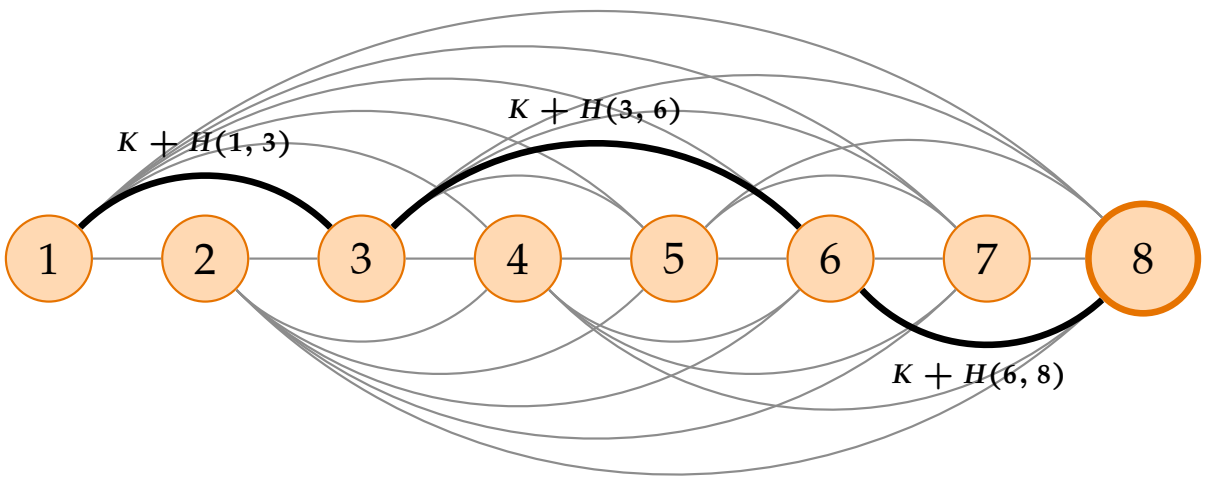
FIGURE – Point de vue "warehouse"

Modèle à un échelon

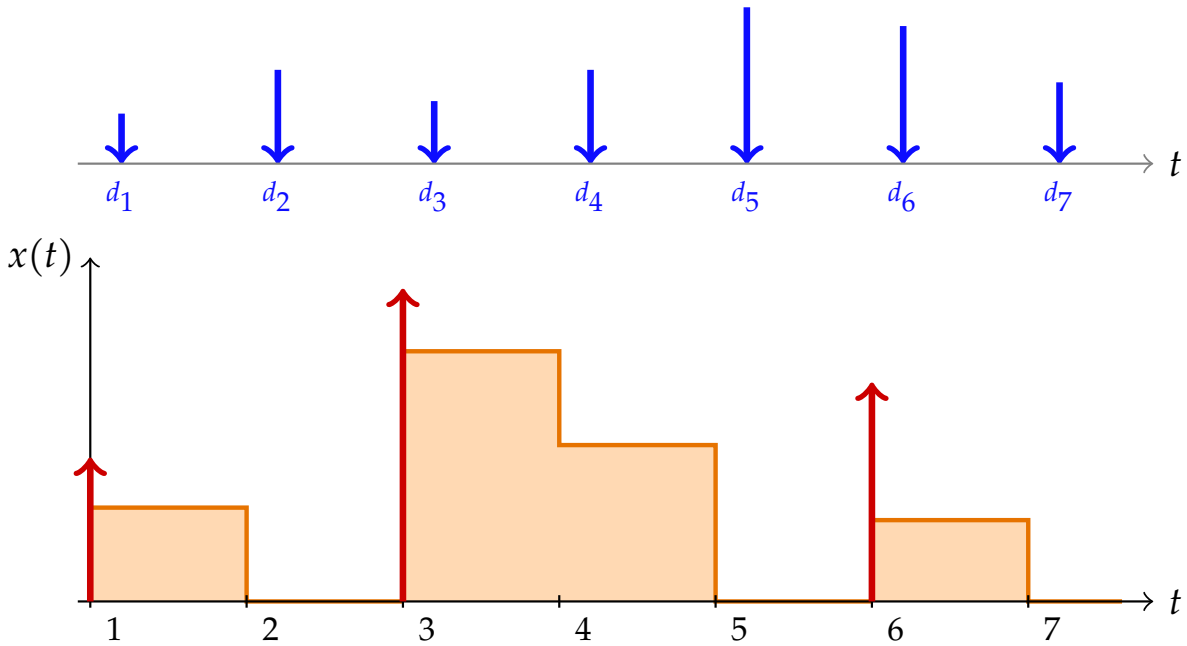


Modèle à un échelon





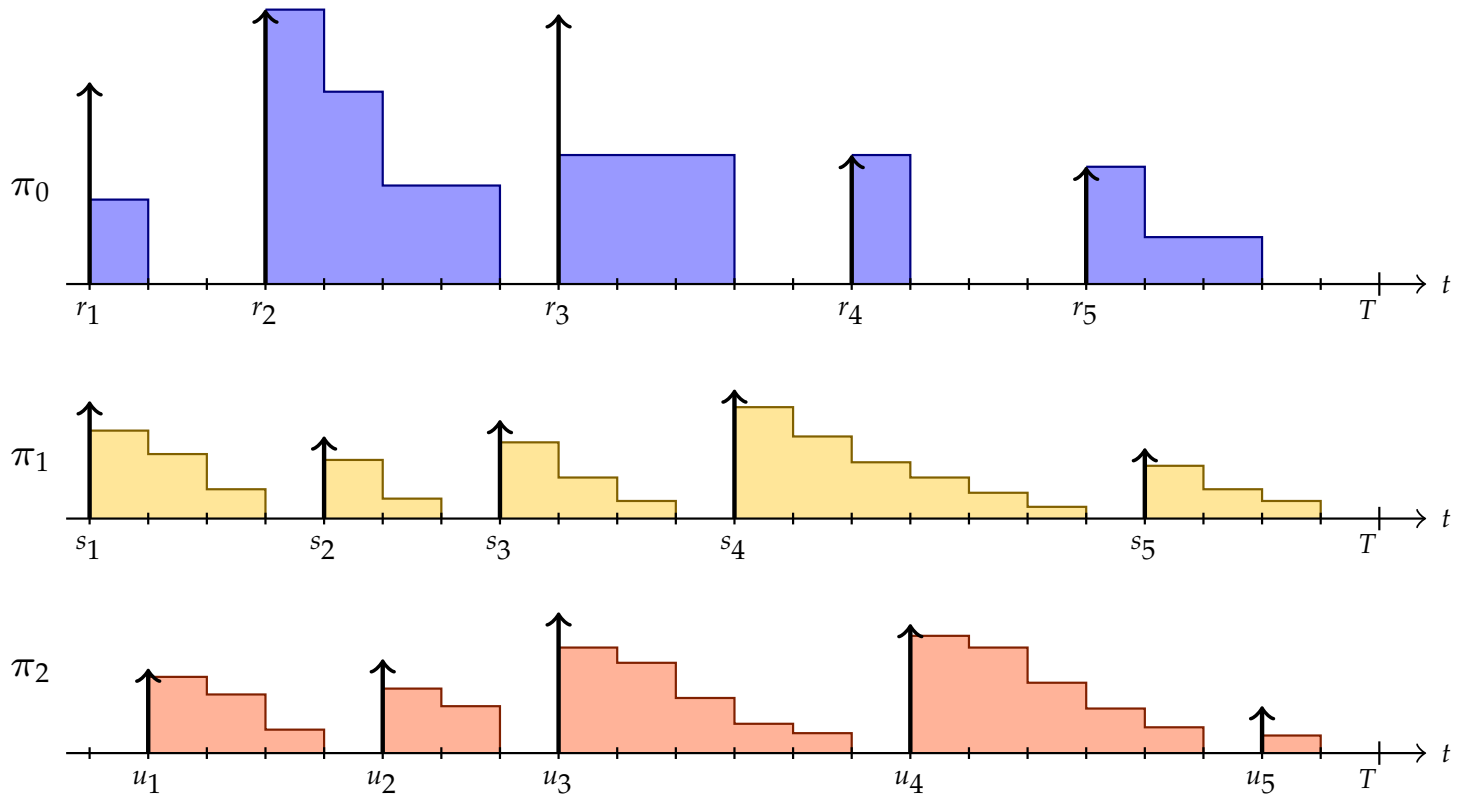
Modèle à un échelon



→ Calcul de la solution optimale par programmation dynamique : Wagner & Whitin (1958), Federguren & Tzur(1991), Wagelmans et al. (1992), Aggarwal & Park (1993).

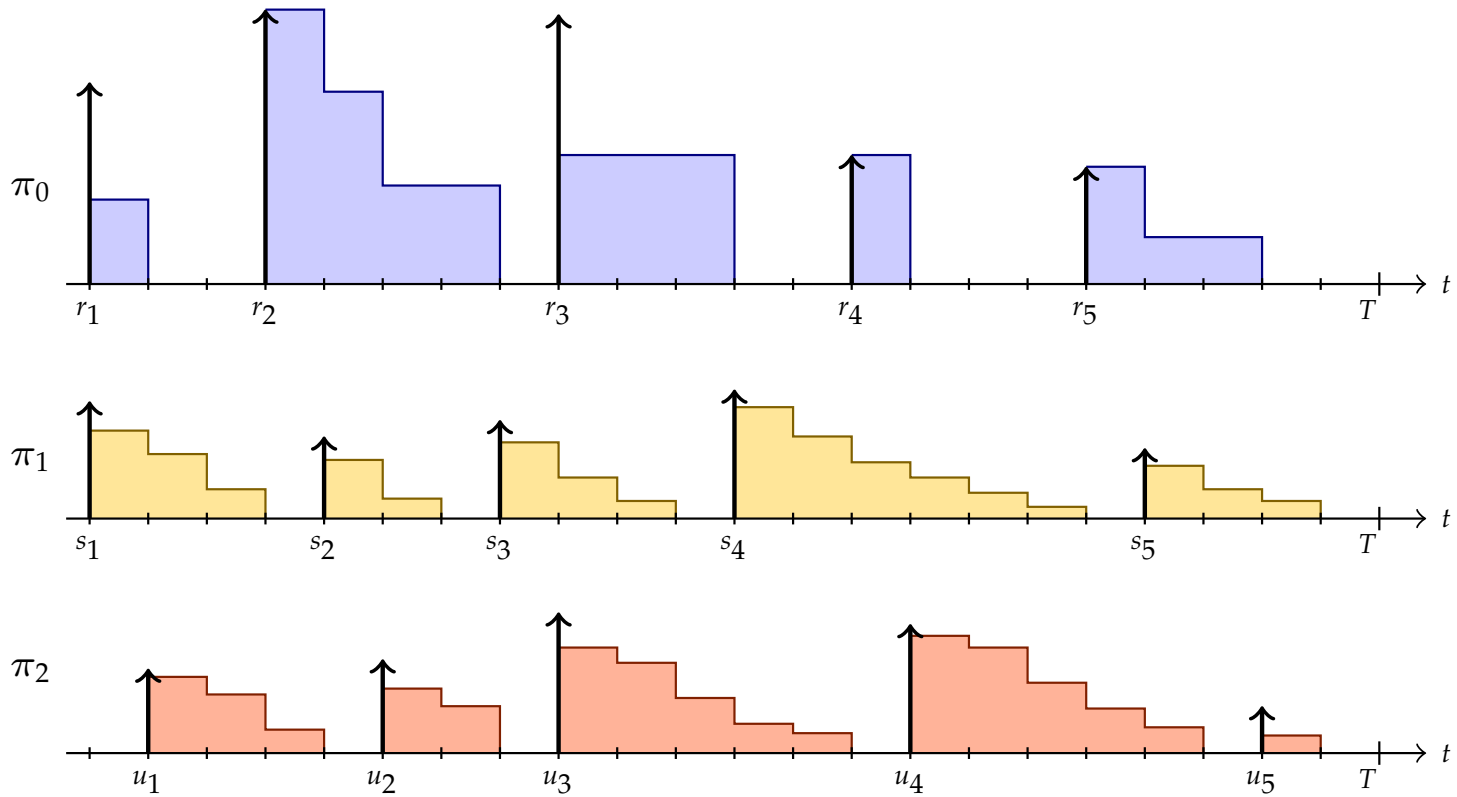
Borne inférieure

$$\mathcal{C}^{\pi} \leq \mathcal{K}_0 + \mathcal{H}_0 + \mathcal{K}_1 + \mathcal{H}_1 + \mathcal{K}_2 + \mathcal{H}_2$$



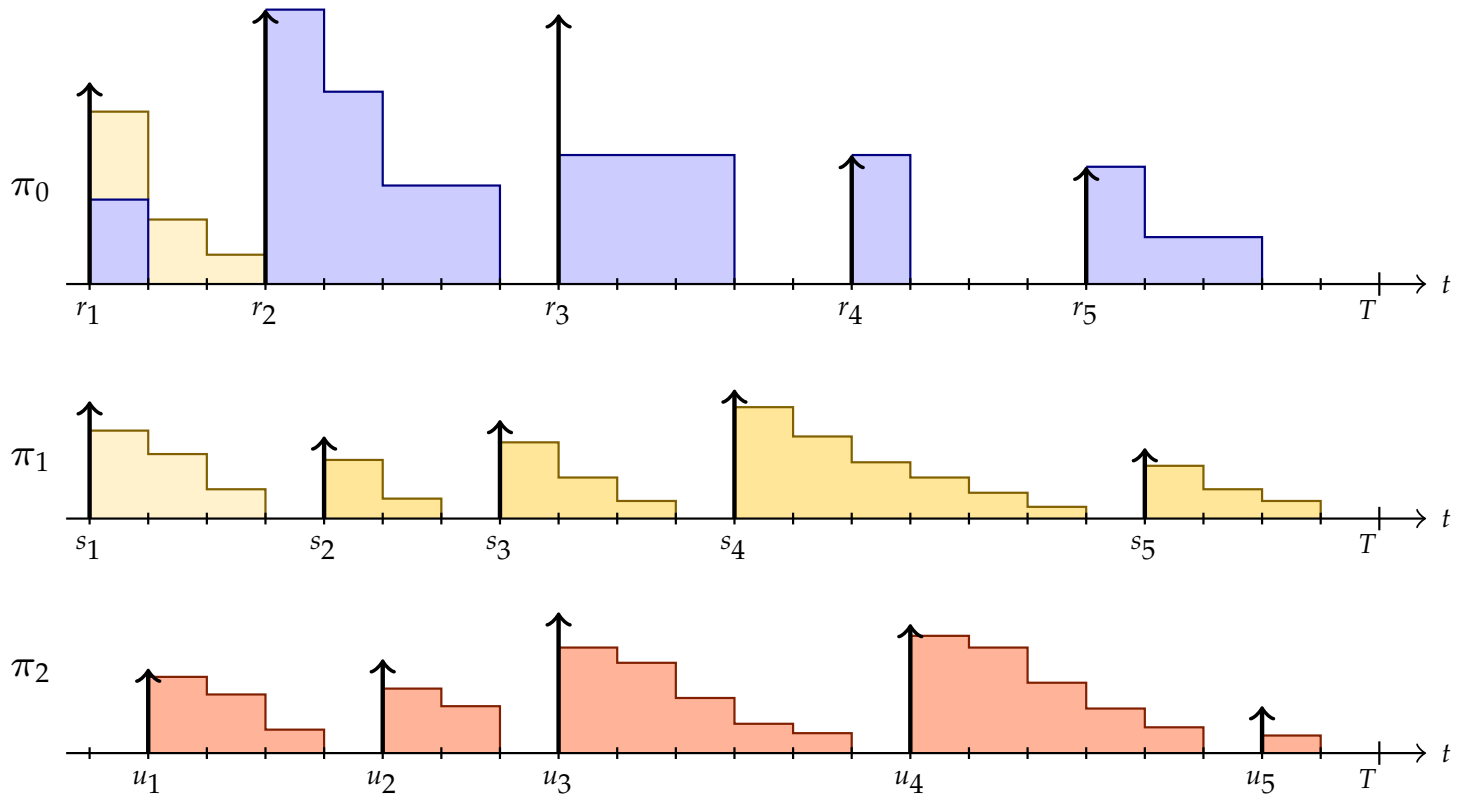
Borne inférieure

$$\mathcal{C}^{\pi} \leq \mathcal{K}_0 + \frac{1}{2}\mathcal{H}_0 + \mathcal{K}_1 + \mathcal{H}_1 + \mathcal{K}_2 + \mathcal{H}_2$$



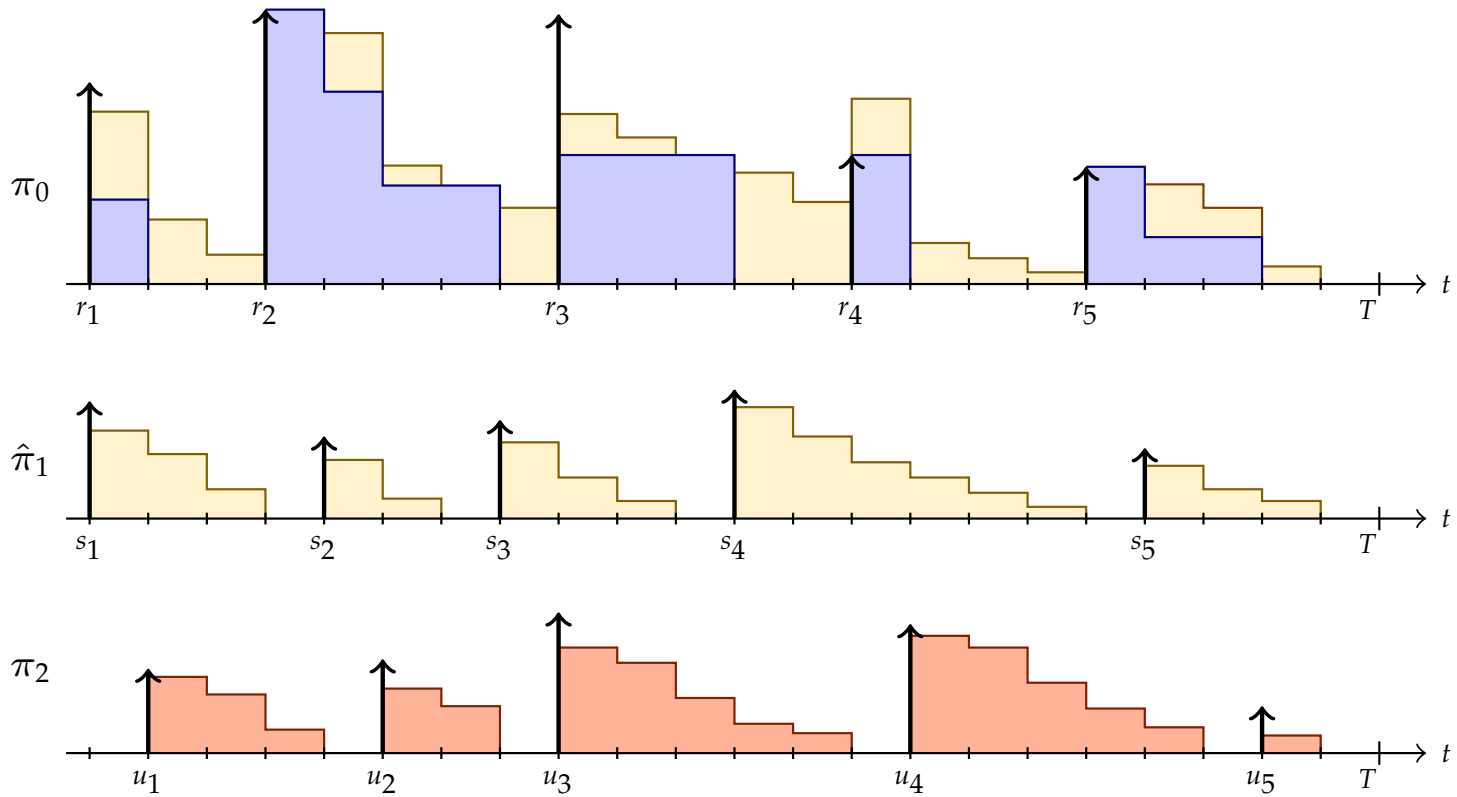
Borne inférieure

$$\mathcal{C}^{\pi} \leq \mathcal{K}_0 + \frac{1}{2}\mathcal{H}_0 + \mathcal{K}_1 + \mathcal{H}_1 + \mathcal{K}_2 + \mathcal{H}_2$$



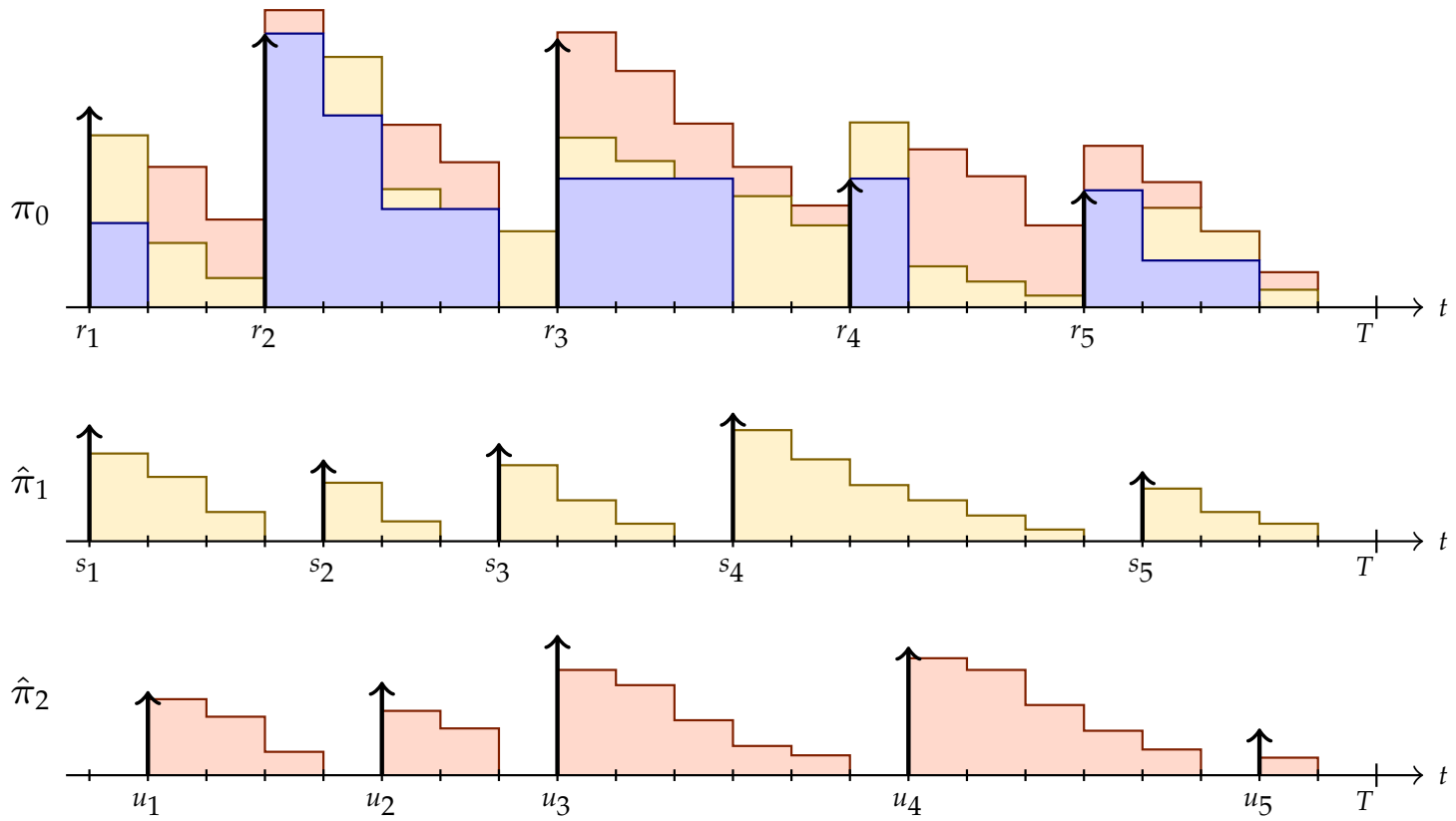
Borne inférieure

$$\mathcal{C}^{\pi} \leq \mathcal{K}_0 + \frac{1}{2}\mathcal{H}_0 + \frac{1}{2}\mathcal{H}_1 + \mathcal{K}_1 + \frac{1}{2}\mathcal{H}_1 + \mathcal{K}_2 + \mathcal{H}_2$$



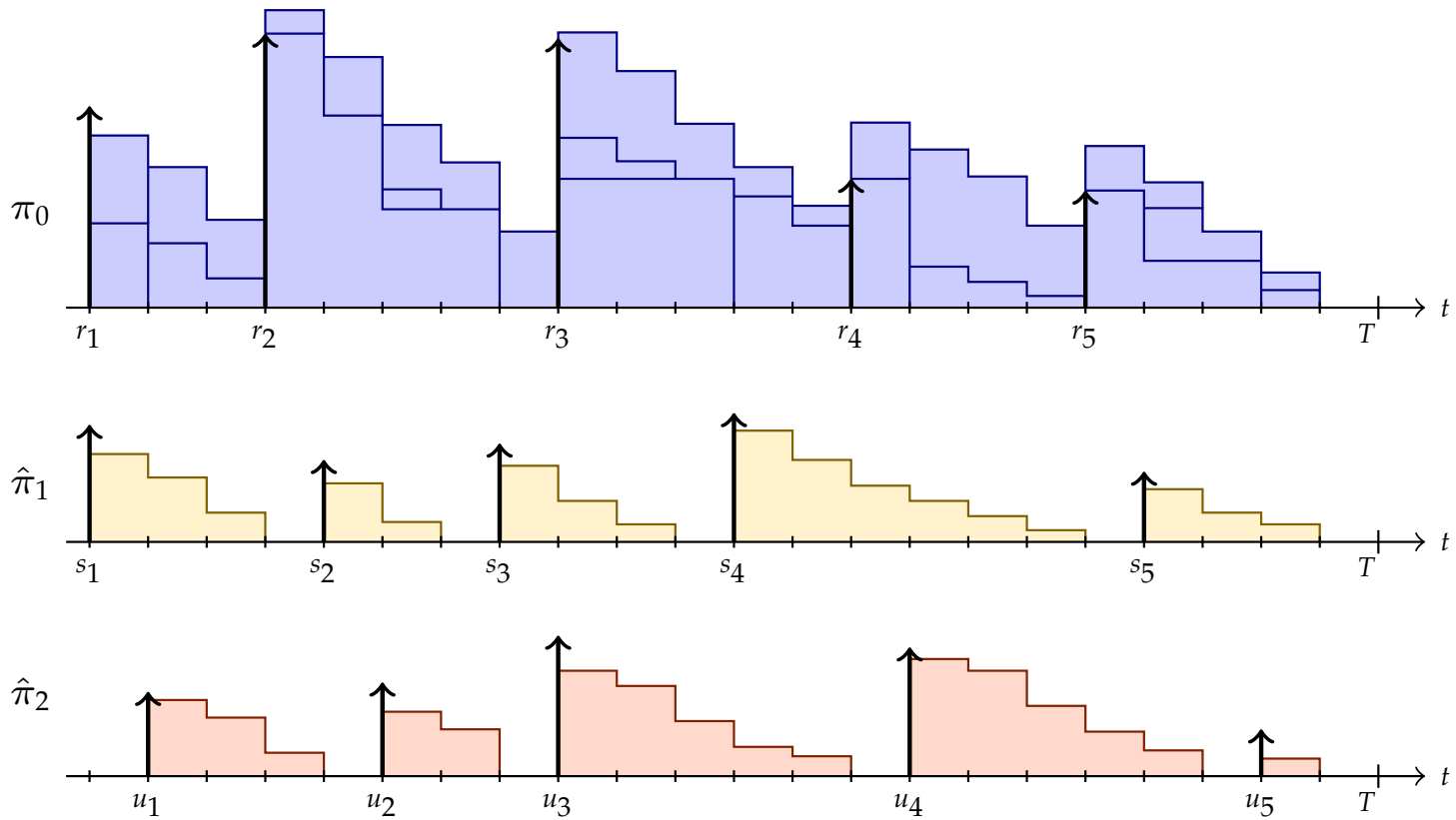
Borne inférieure

$$\mathcal{C}^{\pi} \leq \mathcal{K}_0 + \frac{1}{2}\mathcal{H}_0 + \frac{1}{2}\mathcal{H}_1 + \frac{1}{2}\mathcal{H}_2 + \mathcal{K}_1 + \frac{1}{2}\mathcal{H}_1 + \mathcal{K}_2 + \frac{1}{2}\mathcal{H}_2$$



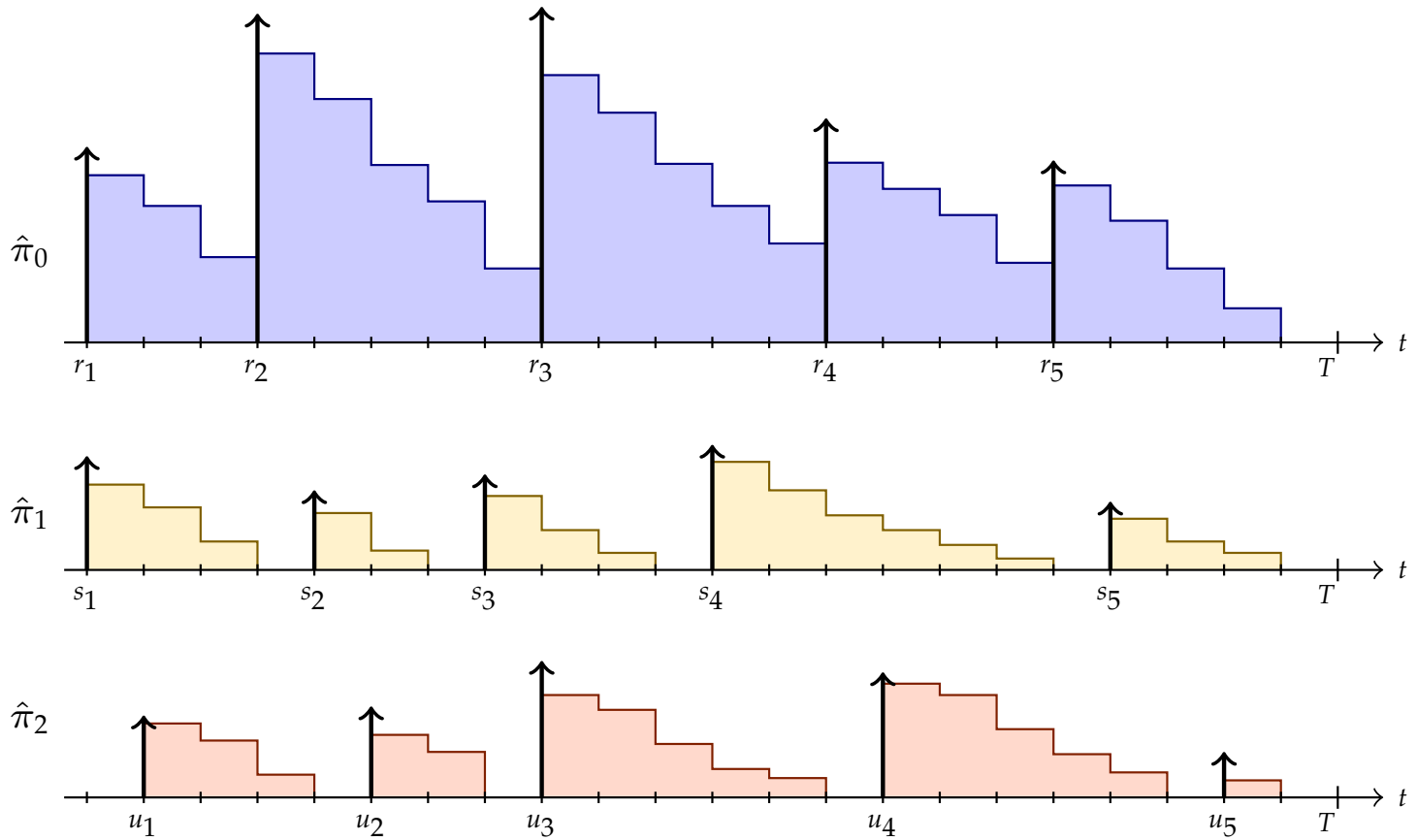
Borne inférieure

$$\mathcal{C}^{\pi} \leq \mathcal{K}_0 + \widehat{\mathcal{H}_0} + \mathcal{K}_1 + \widehat{\mathcal{H}}_1 + \mathcal{K}_2 + \widehat{\mathcal{H}}_2$$



Borne inférieure

$$\mathcal{C}^{\pi} \leq \mathcal{K}_0 + \widehat{\mathcal{H}_0} + \mathcal{K}_1 + \widehat{\mathcal{H}}_1 + \mathcal{K}_2 + \widehat{\mathcal{H}}_2$$



PLAN DE LA PRÉSENTATION

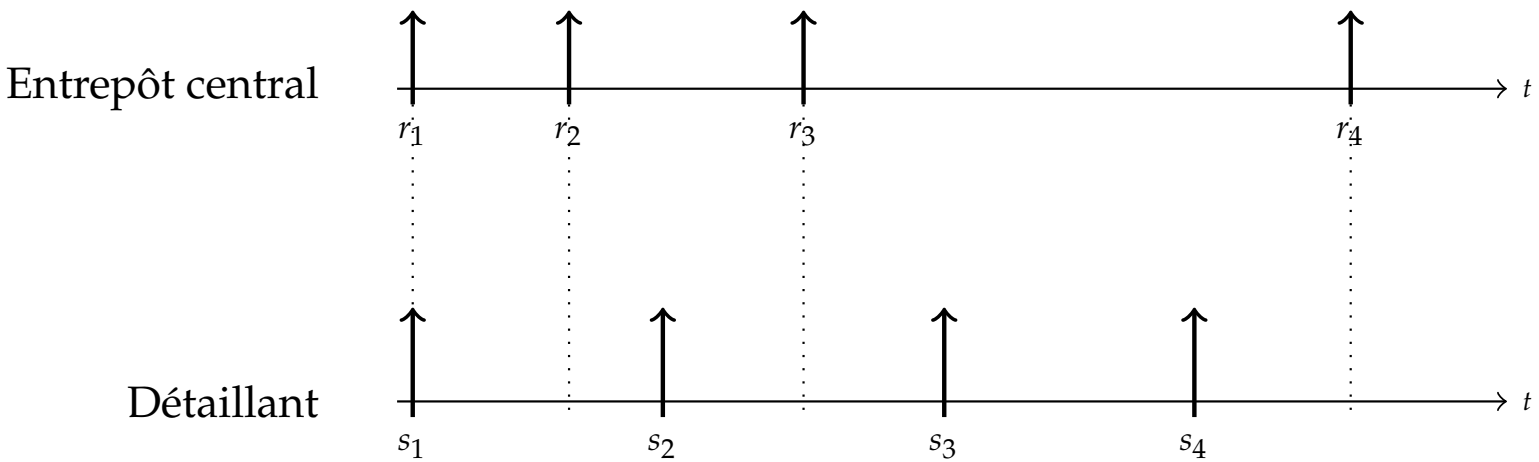
Idée de l'algorithme

Utiliser la décomposition en sous-systèmes à un échelon pour résoudre le problème OWMR.

→ **Problème** : commandes croisées !

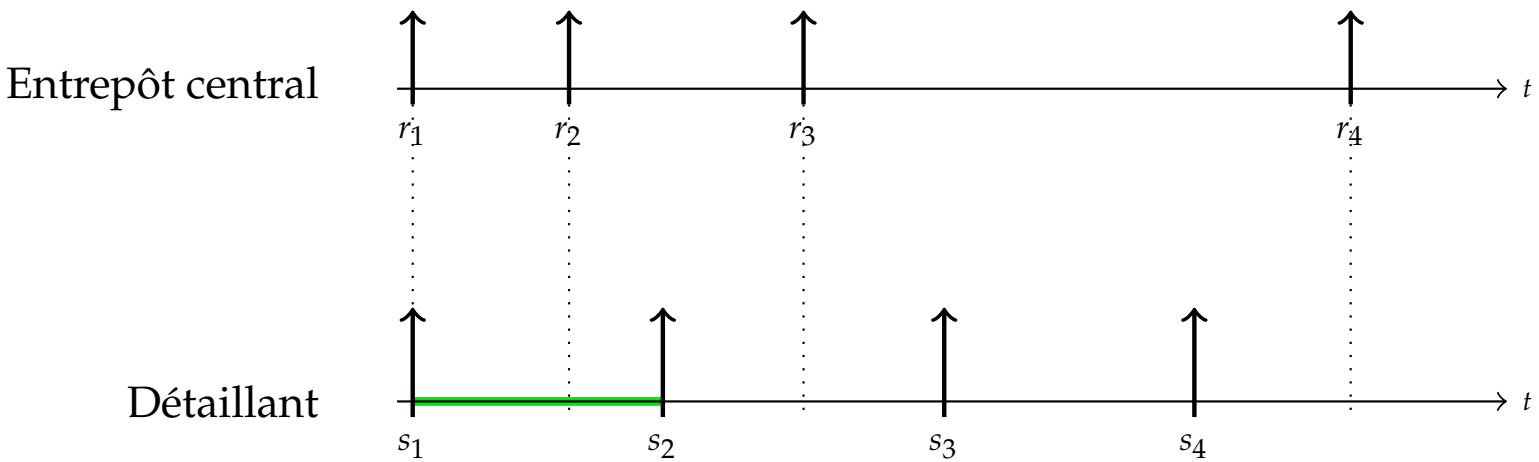
Commandes croisées

Soit $\pi = (\pi_0, \pi_1, \ldots, \pi_N)$ une politique pour le système OWMR considéré. Un intervalle de commande $(s, s']$ de π_i est dit *croisé* si il existe un intervalle de commande $(r, r']$ de π_0 tel que $r < s < r' < s'$.



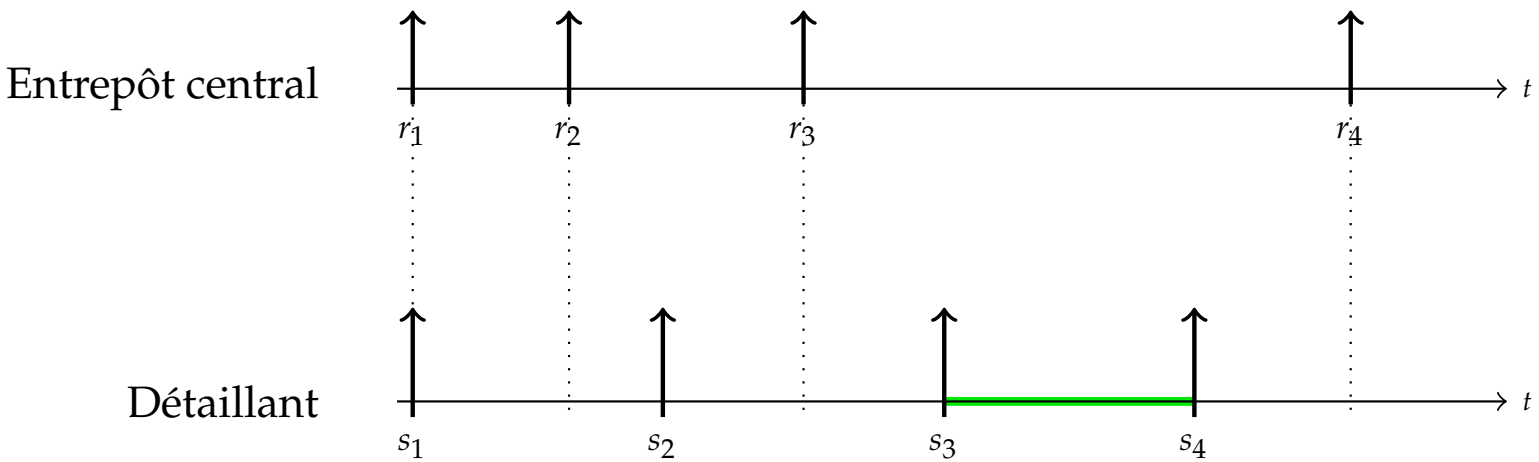
Commandes croisées

Soit $\pi = (\pi_0, \pi_1, \ldots, \pi_N)$ une politique pour le système OWMR considéré. Un intervalle de commande $(s, s']$ de π_i est dit *croisé* si il existe un intervalle de commande $(r, r']$ de π_0 tel que $r < s < r' < s'$.



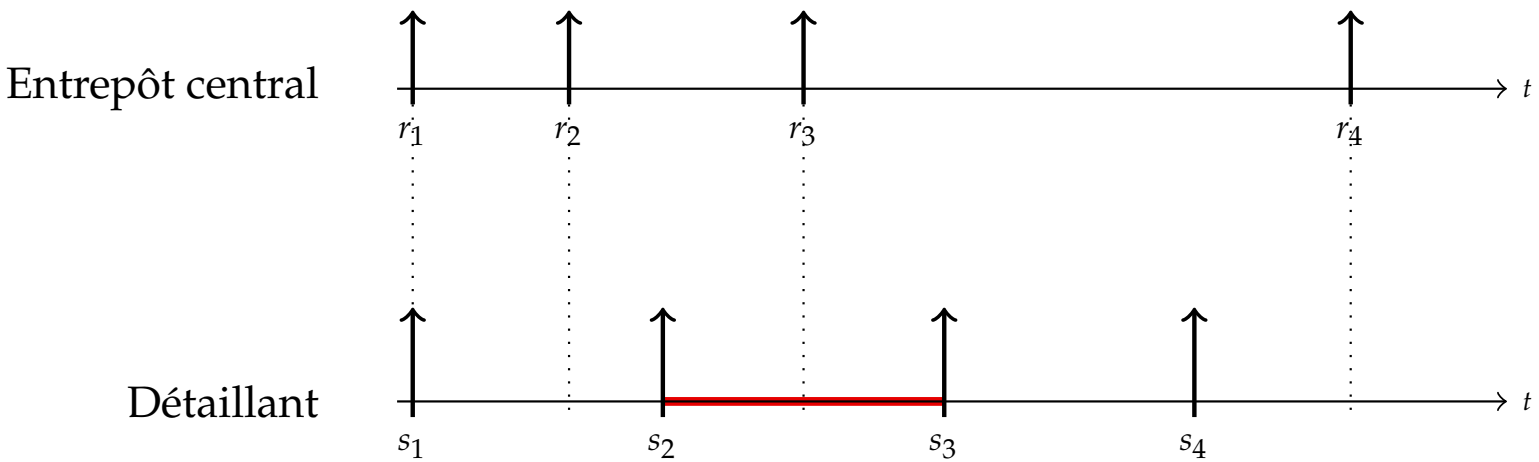
Commandes croisées

Soit $\pi = (\pi_0, \pi_1, \ldots, \pi_N)$ une politique pour le système OWMR considéré. Un intervalle de commande $(s, s']$ de π_i est dit *croisé* si il existe un intervalle de commande $(r, r']$ de π_0 tel que $r < s < r' < s'$.



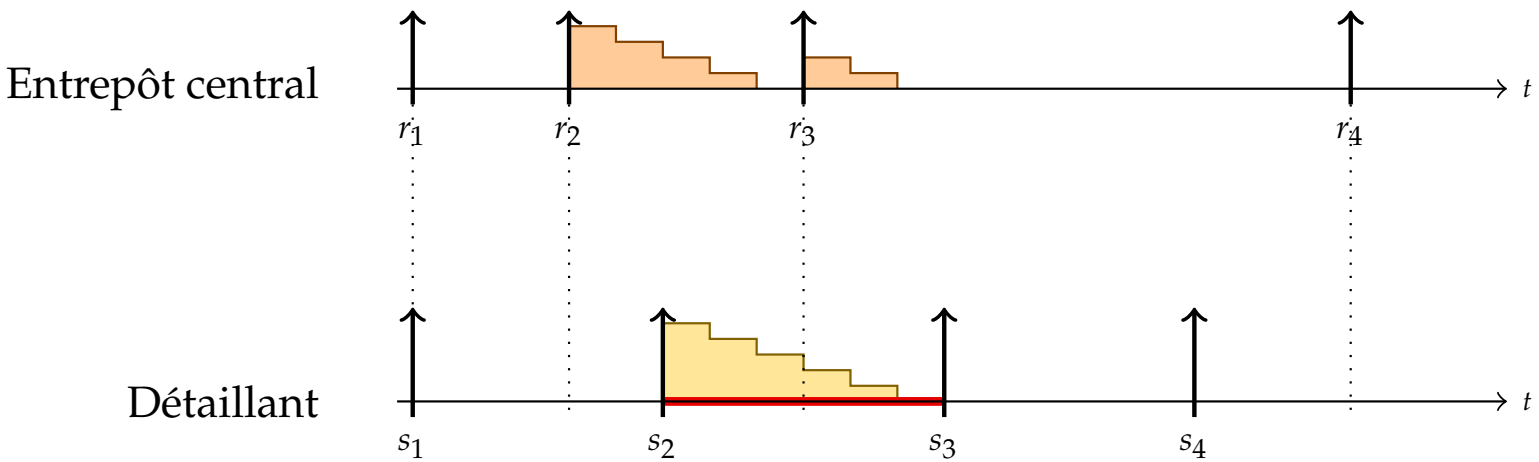
Commandes croisées

Soit $\pi = (\pi_0, \pi_1, \ldots, \pi_N)$ une politique pour le système OWMR considéré. Un intervalle de commande $(s, s']$ de π_i est dit *croisé* si il existe un intervalle de commande $(r, r']$ de π_0 tel que $r < s < r' < s'$.



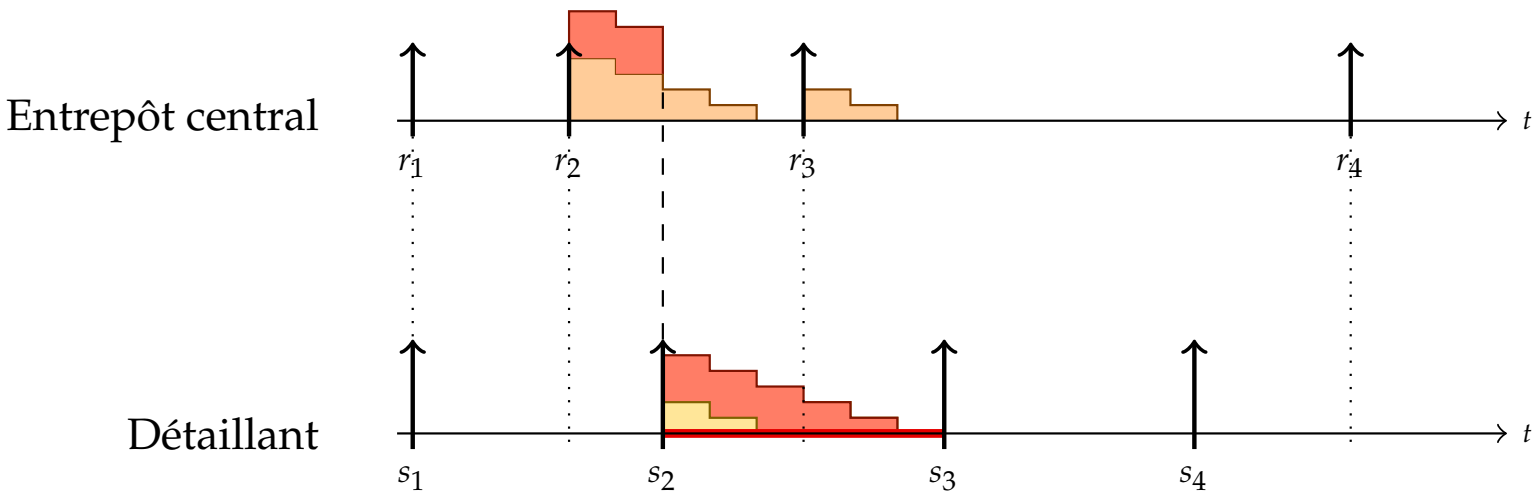
Commandes croisées

Soit $\pi = (\pi_0, \pi_1, \ldots, \pi_N)$ une politique pour le système OWMR considéré. Un intervalle de commande $(s, s']$ de π_i est dit *croisé* si il existe un intervalle de commande $(r, r']$ de π_0 tel que $r < s < r' < s'$.



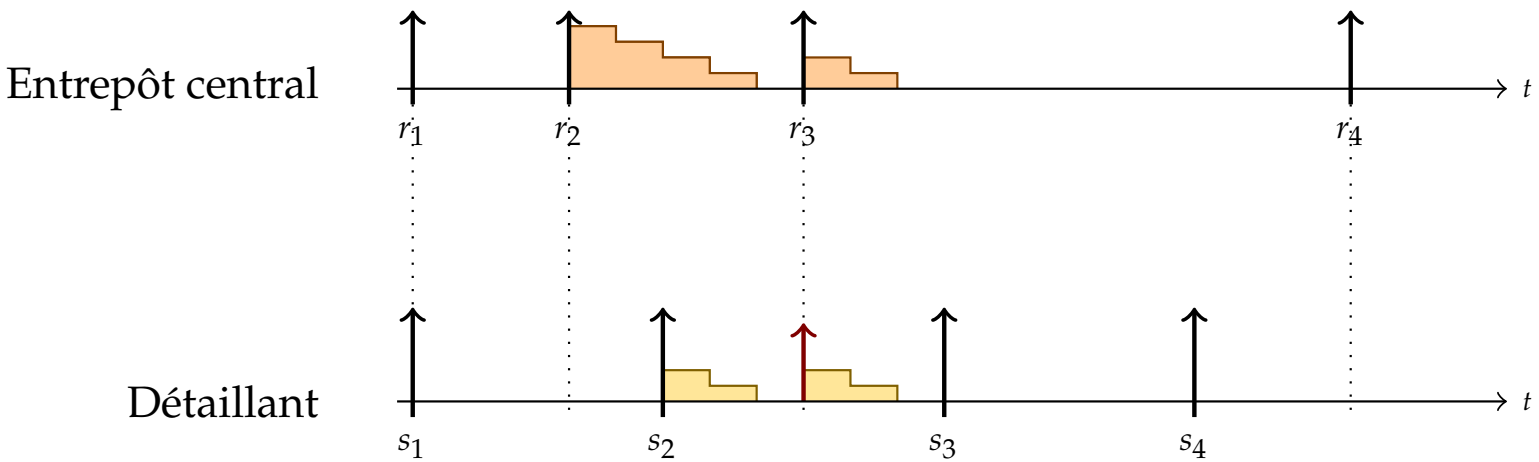
Commandes croisées

Soit $\pi = (\pi_0, \pi_1, \ldots, \pi_N)$ une politique pour le système OWMR considéré. Un intervalle de commande $(s, s']$ de π_i est dit *croisé* si il existe un intervalle de commande $(r, r']$ de π_0 tel que $r < s < r' < s'$.



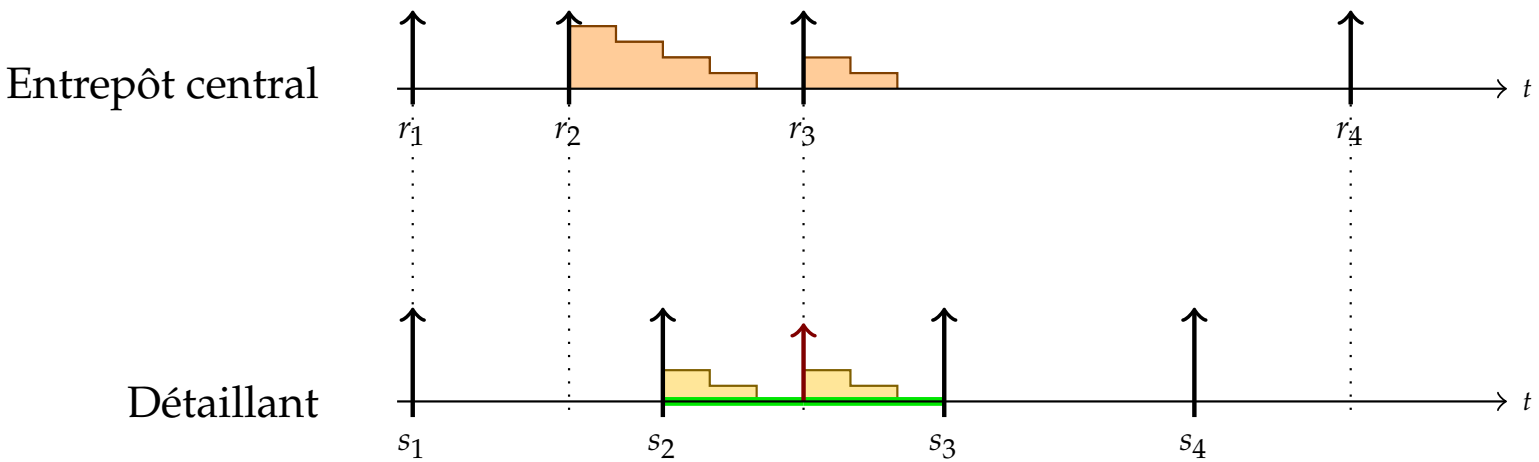
Commandes croisées

Soit $\pi = (\pi_0, \pi_1, \ldots, \pi_N)$ une politique pour le système OWMR considéré. Un intervalle de commande $(s, s']$ de π_i est dit *croisé* si il existe un intervalle de commande $(r, r']$ de π_0 tel que $r < s < r' < s'$.



Commandes croisées

Soit $\pi = (\pi_0, \pi_1, \ldots, \pi_N)$ une politique pour le système OWMR considéré. Un intervalle de commande $(s, s']$ de π_i est dit *croisé* si il existe un intervalle de commande $(r, r']$ de π_0 tel que $r < s < r' < s'$.



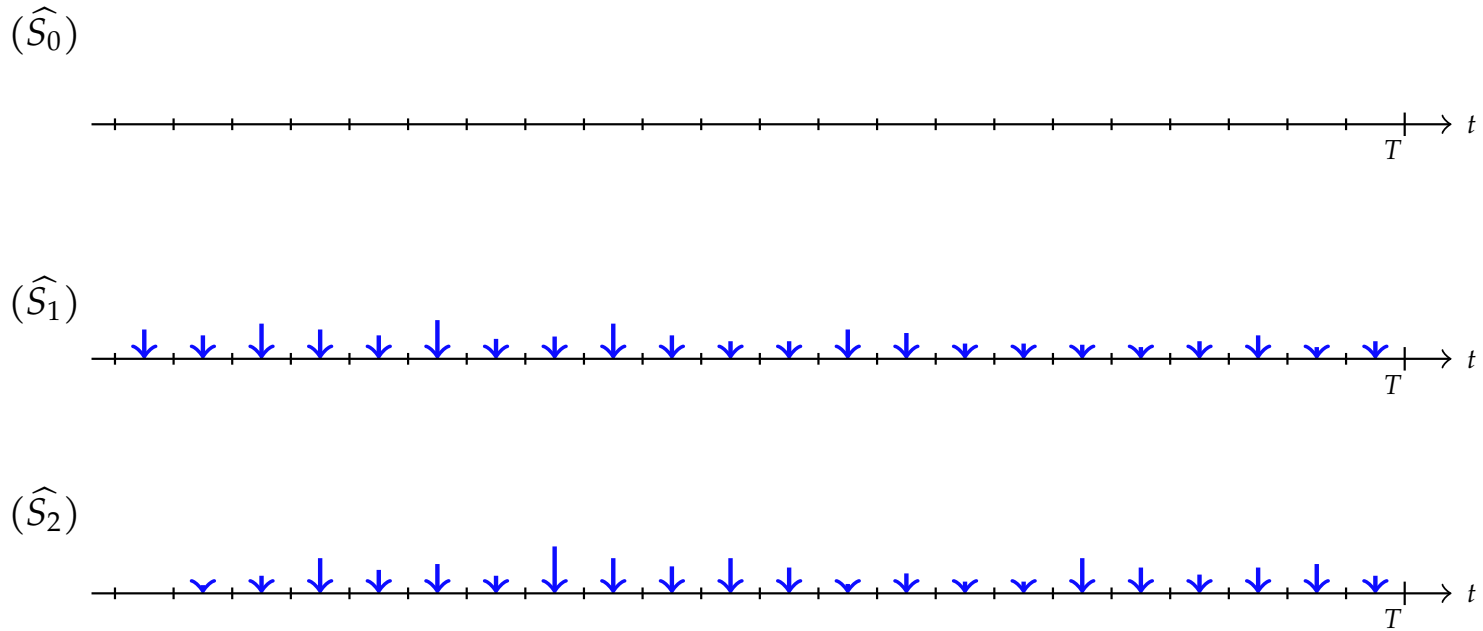
Algorithme "Split & Uncross"

Décomposer Transformer le problème en $N+1$ sous-problèmes $(S_0), (S_1), \ldots, (S_N)$ et résoudre chacun d'entre eux à l'optimal $\pi_0^*, \pi_1^*, \ldots, \pi_N^*$.

Décroiser Pour chaque détaillant i et chaque intervalle de commande s, s' de π_i^* croisé avec un intervalle r, r' de π_0^*, ajouter une commande à π_i^* en r'.

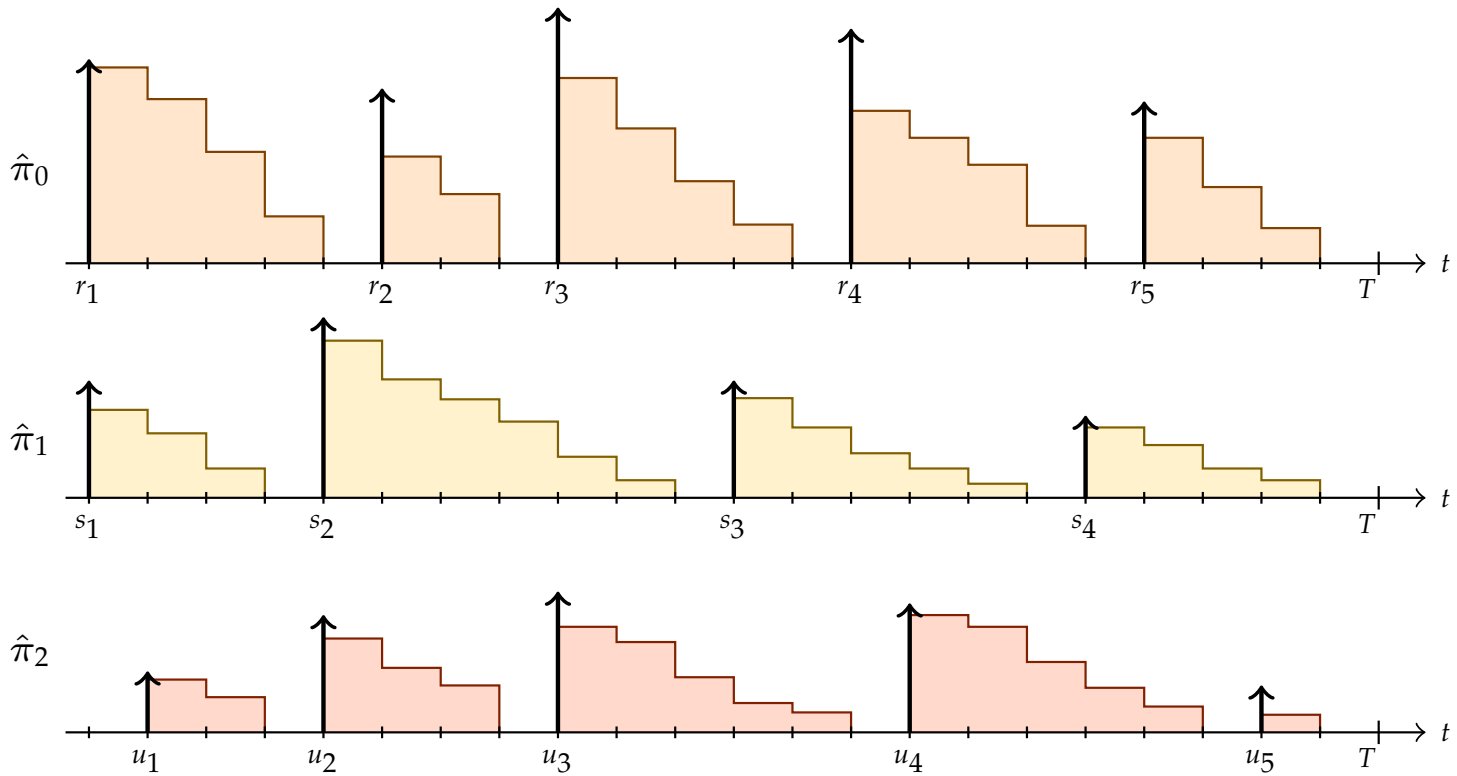
Ajuster les quantités commandées pour retrouver une politique ZIO aux détaillants et à l'entrepôt central.

Exemple d'application



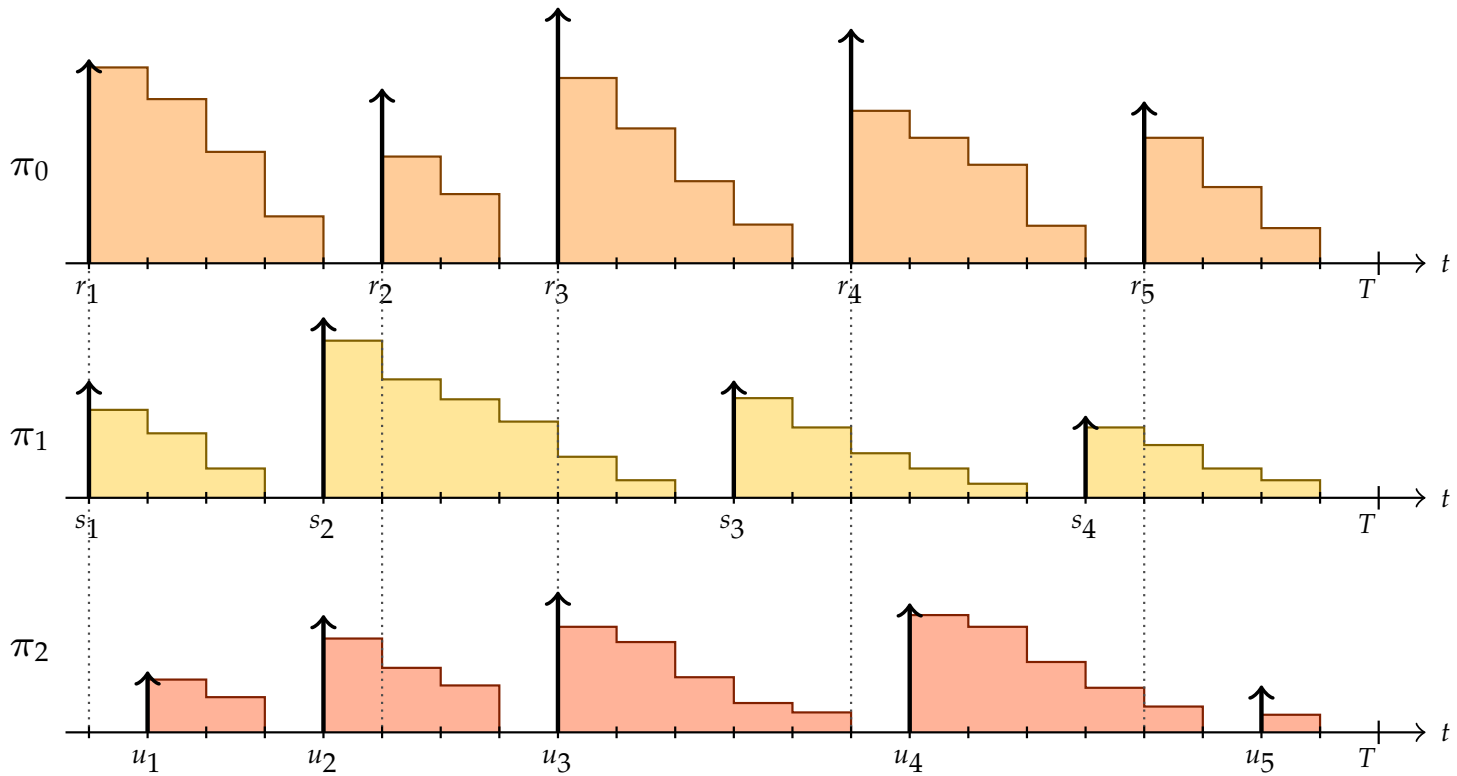
Exemple d'application

$$\mathcal{C}^{\hat{\pi}^*} = \mathcal{K}_0 + \widehat{\mathcal{H}}_0 + \mathcal{K}_1 + \widehat{\mathcal{H}}_1 + \mathcal{K}_2 + \widehat{\mathcal{H}}_2$$



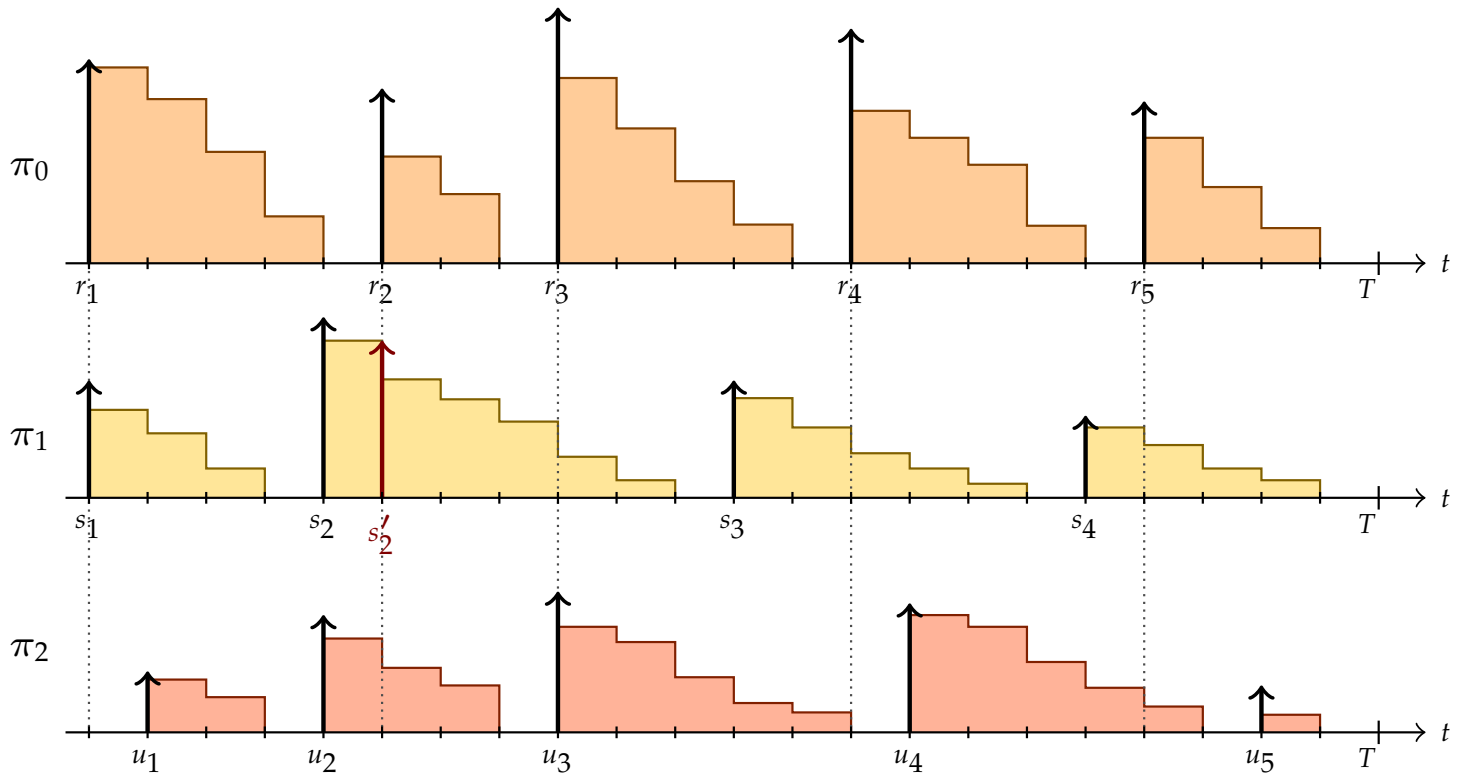
Exemple d'application

$$\mathcal{C}^{\pi} = \mathcal{K}_0 + 2\widehat{\mathcal{H}}_0 + \mathcal{K}_1 + 2\widehat{\mathcal{H}}_1 + \mathcal{K}_2 + 2\widehat{\mathcal{H}}_2$$



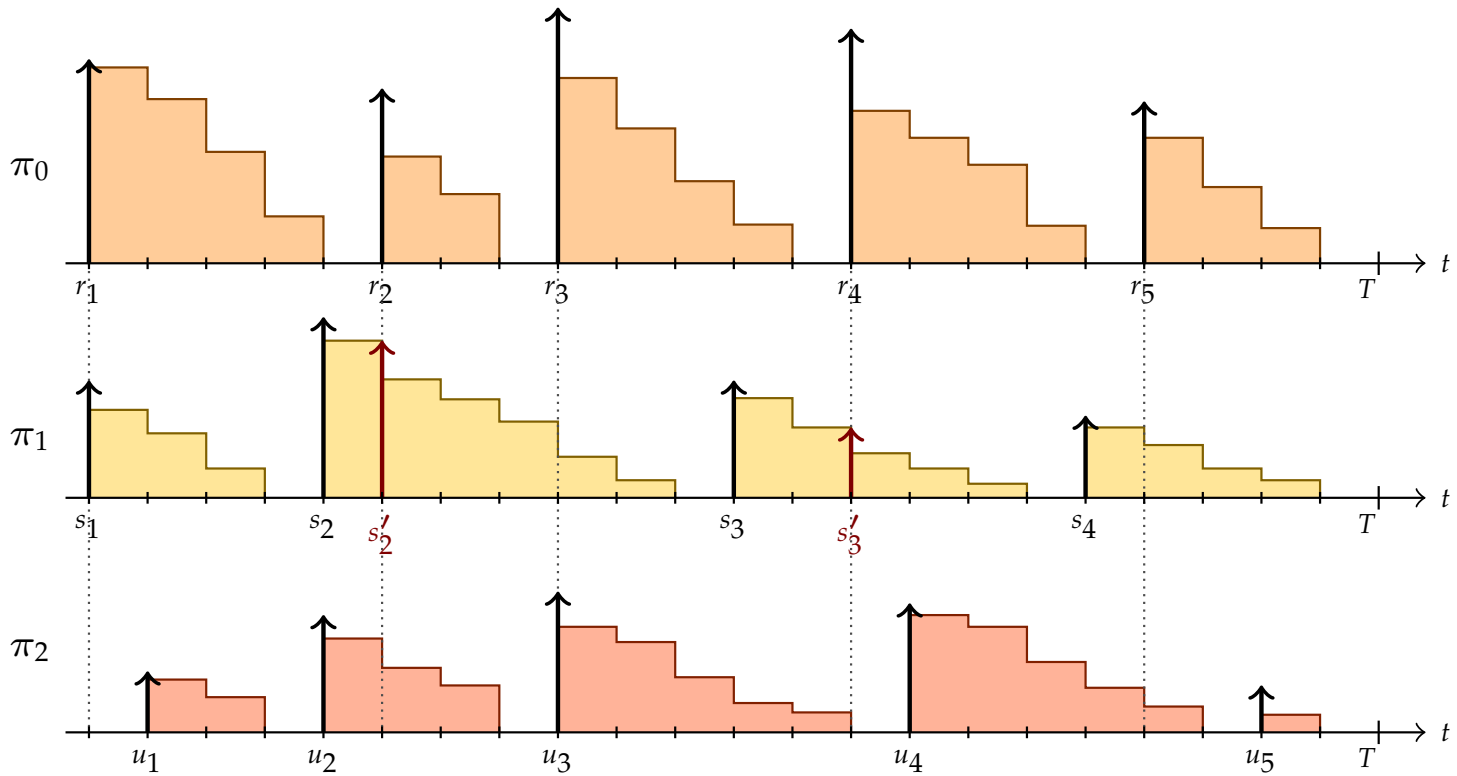
Exemple d'application

$$\mathcal{C}^{\pi} = \mathcal{K}_0 + 2\widehat{\mathcal{H}}_0 + \mathcal{K}_1 + 2\widehat{\mathcal{H}}_1 + \mathcal{K}_2 + 2\widehat{\mathcal{H}}_2$$



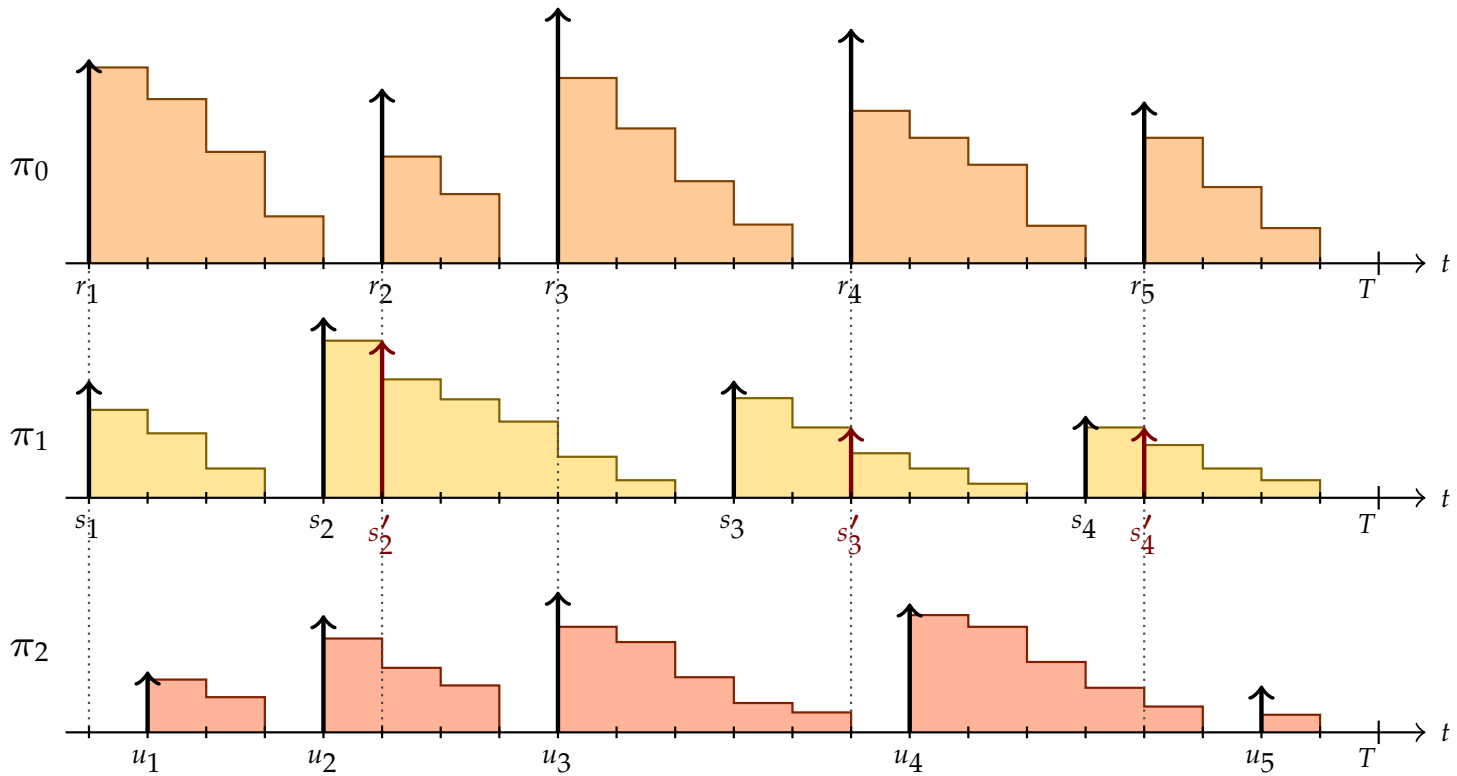
Exemple d'application

$$\mathcal{C}^{\pi} = \mathcal{K}_0 + 2\widehat{\mathcal{H}}_0 + \mathcal{K}_1 + 2\widehat{\mathcal{H}}_1 + \mathcal{K}_2 + 2\widehat{\mathcal{H}}_2$$



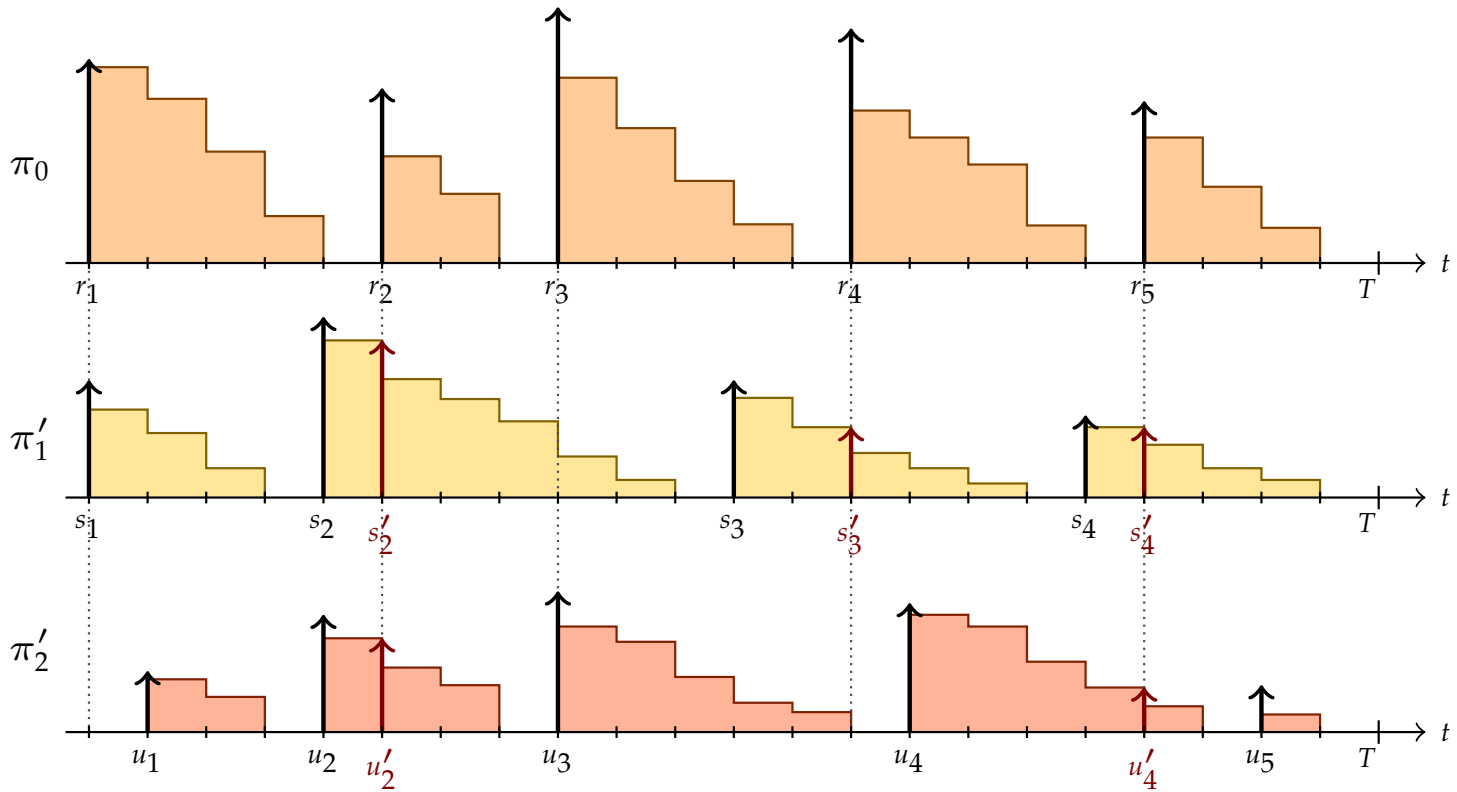
Exemple d'application

$$\mathcal{C}^{\pi} = \mathcal{K}_0 + 2\widehat{\mathcal{H}}_0 + 2\mathcal{K}_1 + 2\widehat{\mathcal{H}}_1 + \mathcal{K}_2 + 2\widehat{\mathcal{H}}_2$$



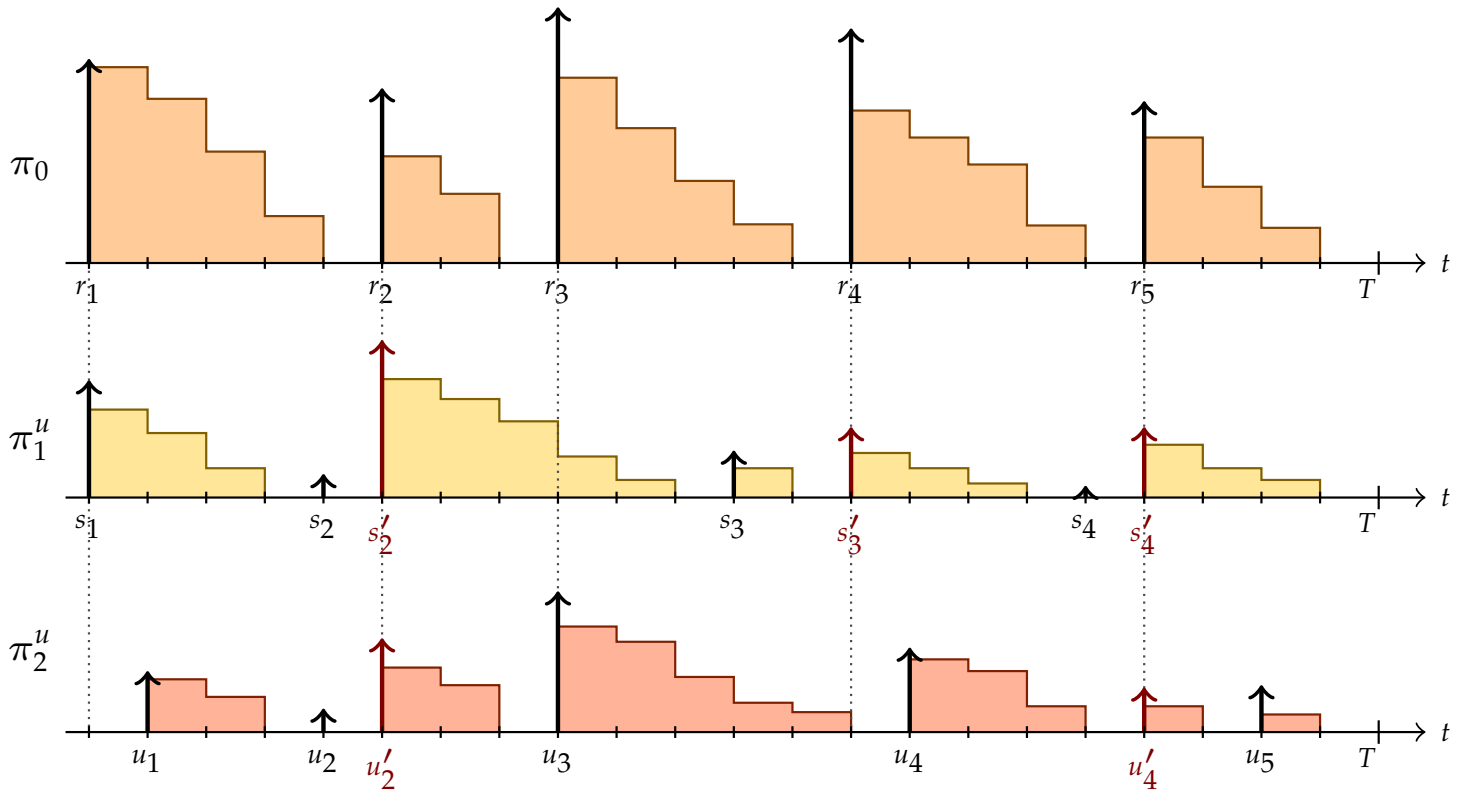
Exemple d'application

$$\mathcal{C}^{\pi} = \mathcal{K}_0 + 2\widehat{\mathcal{H}}_0 + 2\mathcal{K}_1 + 2\widehat{\mathcal{H}}_1 + 2\mathcal{K}_2 + 2\widehat{\mathcal{H}}_2$$

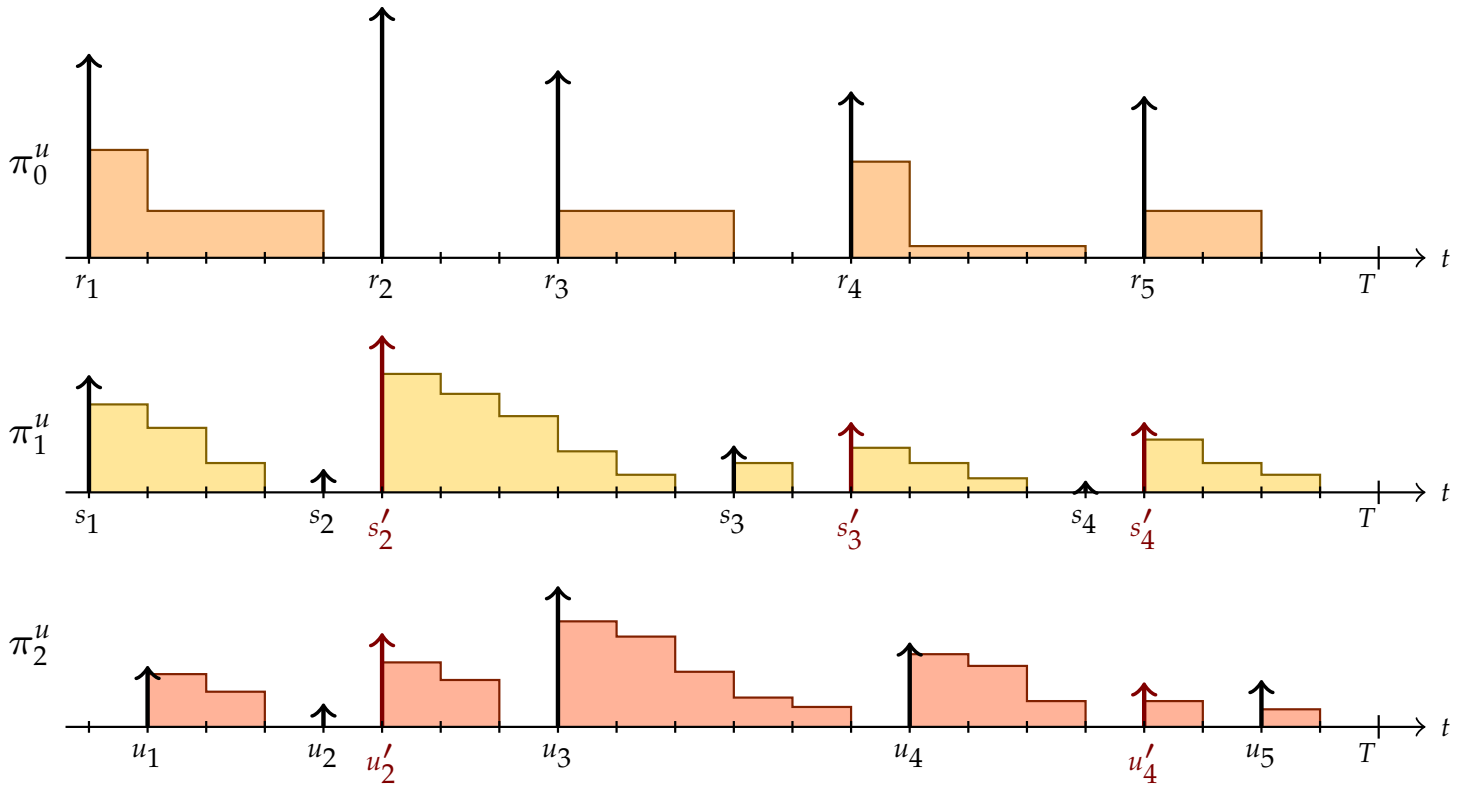


Exemple d'application

$$\mathcal{C}^{\pi} = \mathcal{K}_0 + 2\widehat{\mathcal{H}}_0 + 2\mathcal{K}_1 + 2\widehat{\mathcal{H}}_1 + 2\mathcal{K}_2 + 2\widehat{\mathcal{H}}_2$$



Exemple d'application



Théorème

L'algorithme "Split & Uncross" a une garantie de performance égale à 2 pour le problème OWMR discret.

Preuve :

1. $(\hat{\pi}_0^*, \hat{\pi}_1^*, \ldots, \hat{\pi}_N^*)$ donne une borne inférieure de π^*.
2. "Split & Uncross" double les coûts de $\hat{\pi}_0^*$ et $\hat{\pi}_i^*$ dans le pire cas.
3. Ajuster les quantités n'augmente pas les coûts de stockage.

Plan de la présentation

CONCLUSION

Extensions

- Deux types de détaillants ($h_t^i \leq h_t^0$ ou $h_t^i \geq h_t^0$).
- Structure de coûts de possession plus générale : *shelf-age* et non linéaire.
- Structure de coûts de commande plus générale : FTL/LTL.
- Capacité de commande pour certains détaillants.
- Possibilité de mise en attente (3-approximation) ou de ventes perdues (2-approximation).

Perspectives

- Extension des contraintes de capacité de commandes à tous les type de détaillants.
- Amélioration de la borne pour le modèle avec mise en attente.
- Extension pour des réseaux de distributions à 3 "étages" ou plus.
- Analyse numérique et comparaison avec les techniques de la littérature.

Merci pour votre attention !

Questions