Choix de Vitesses pour la Minimisation de la Consommation d'Energie Hors-Ligne en $\mathcal{O}(n)$

Bruno Gaujal, Alain Girault and **Stephan Plassart**

13 Novembre 2019

Sommaire

Présentation du modèle

Ensemble fini de tâches, exécutées sur un seul processeur mono-coeur.

Présentation du modèle

Ensemble fini de tâches, exécutées sur un seul processeur mono-coeur.

Tâche (r, c, d) caractérisée par:

- r: instant d'arrivée
- c: temps d'exécution
- d: échéance relative

Présentation du modèle

Ensemble fini de tâches, exécutées sur un seul processeur mono-coeur.

Tâche (r, c, d) caractérisée par:

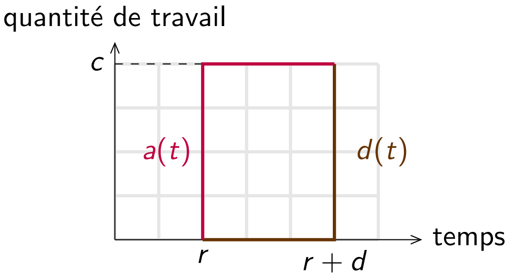
- r: instant d'arrivée
- c: temps d'exécution
- d: échéance relative

Objectif: Choisir la vitesse du processeur s tel que:

1. Chaque tâche se termine avant son échéance.
2. La consommation d'énergie soit minimale.

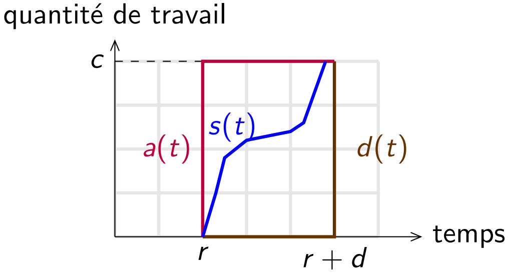
La Vitesse Constante est la Meilleure

Chaque **tâche** (r, c, d) est vue comme une "boîte" (courbes d'arrivée & d'échéance).



La Vitesse Constante est la Meilleure

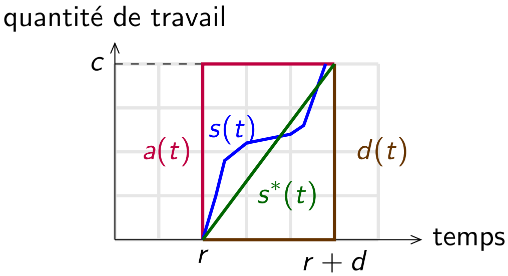
Chaque **tâche** (r, c, d) est vue comme une “boîte” (courbes d'arrivée & d'échéance).



- **La vitesse:** $s(t)$ doit satisfaire $d(t) \leq \int_r^t s(u)du \leq a(t)$
- **Puissance** consommée: $P_{ower}(s(t))$, **Energie:** $\int_r^{r+d} P_{ower}(s(u))du$

La Vitesse Constante est la Meilleure

Chaque **tâche** (r, c, d) est vue comme une "boîte" (courbes d'arrivée & d'échéance).



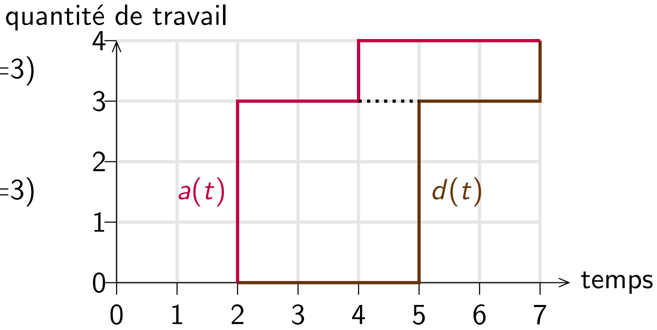
- **La vitesse:** $s(t)$ doit satisfaire $d(t) \leq \int_r^t s(u)du \leq a(t)$
- **Puissance** consommée: $P_{ower}(s(t))$, **Energie:** $\int_r^{r+d} P_{ower}(s(u))du$
- **Convexité**:
$$\int_r^{r+d} P_{ower}(s(u))du \geq d \cdot P_{ower}\left(\frac{1}{d}\int_r^{r+d} s(u)du\right) = d \cdot P_{ower}\left(\frac{c}{d}\right)$$

La Vitesse Constante est Meilleure, mais pas toujours possible

Dans le cas général, garder une vitesse constante peut ne pas être possible. Voici un exemple sur 2 tâches:

$J_2(r_2{=}4, c_2{=}1, d_2{=}3)$

$J_1(r_1{=}2, c_1{=}3, d_1{=}3)$

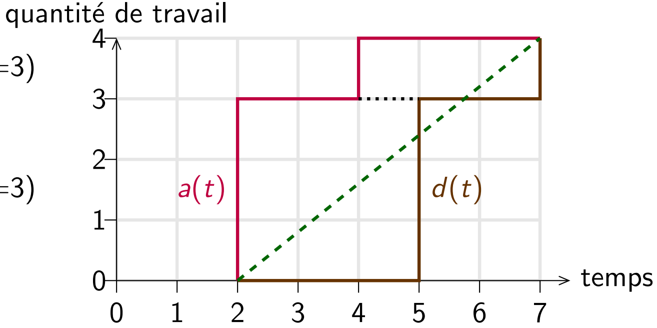


La Vitesse Constante est Meilleure, mais pas toujours possible

Dans le cas général, garder une vitesse constante peut ne pas être possible. Voici un exemple sur 2 tâches:

$J_2(r_2{=}4, c_2{=}1, d_2{=}3)$

$J_1(r_1{=}2, c_1{=}3, d_1{=}3)$

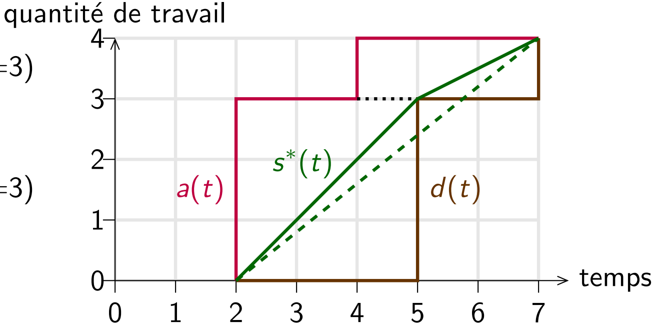


La Vitesse Constante est Meilleure, mais pas toujours possible

Dans le cas général, garder une vitesse constante peut ne pas être possible. Voici un exemple sur 2 tâches:

$J_2(r_2{=}4, c_2{=}1, d_2{=}3)$

$J_1(r_1{=}2, c_1{=}3, d_1{=}3)$



Historique de la Complexité

Beaucoup d'efforts ont été réalisés pour résoudre ce problème:

- 1995: $O(n^3)$ F. Yao et al., Spuri et al.
- 2005: $O(n \log n)$ Gaujal et al. pour des tâches FIFO.
- 2007: $O(Ln^2)$ Gaujal et al. ($L \leq n$: niveau imbriqué)
- 2017: $O(n^2)$ F. Yao et al.

Cas pour m vitesses discrètes:

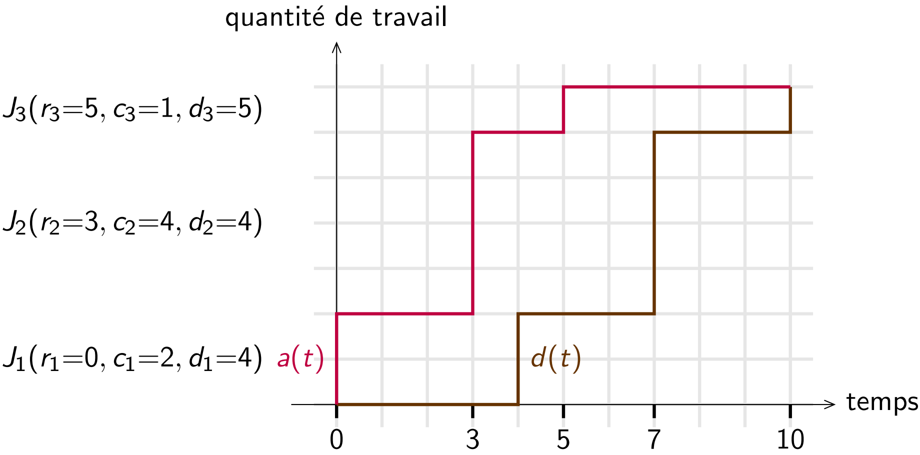
- 1995: $O(n^3)$ F. Yao et al.
- 2005: $O(n \log n)$ Gaujal et al. pour des tâches FIFO.
- 2005: $(mn \log n)$ Yao et al.
- 2017: $O(n \log n)$ Yao et al.

Notre résultat: $\boxed{\text{avec } (r_i, c_i, d_i \leq \Delta)_{1 \leq i \leq n} \in \mathbb{N}\text{: } O(n).}$

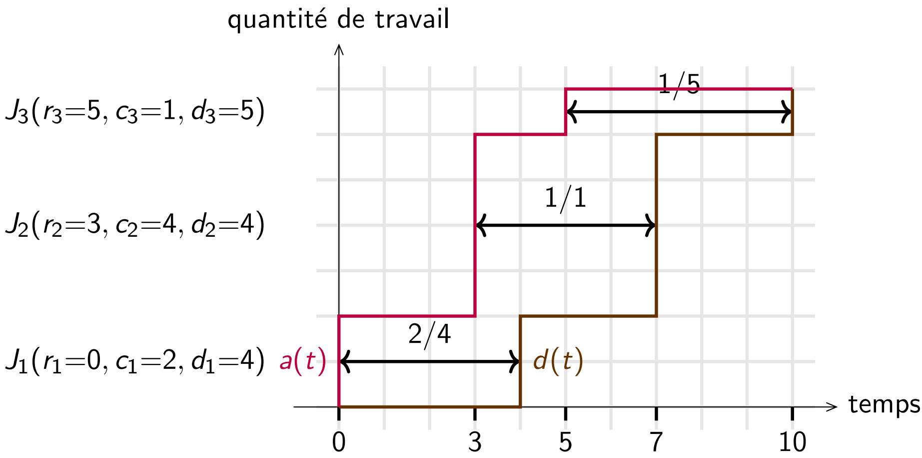
Algorithmes présentés

1. **Algorithme basé sur les intervalles critiques:** $O(n \log n)$
2. **Algorithme basé sur la programmation dynamique:** $O(n)$

Algorithme avec Intervalles Critiques, Yao et al.

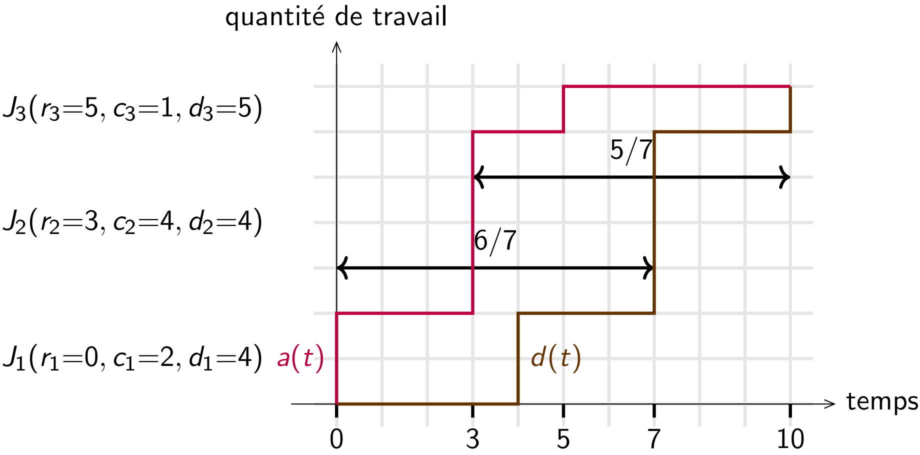


Algorithme avec Intervalles Critiques, Yao et al.



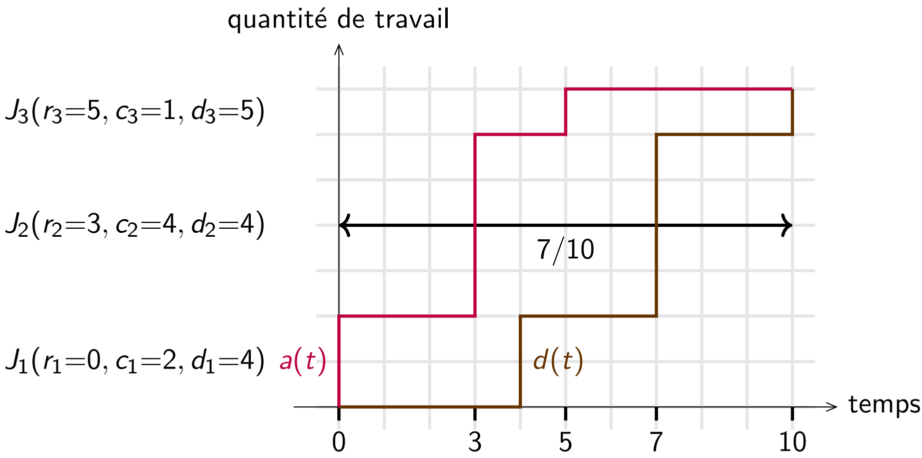
Ensemble d'intervalles $= (2/4, 1/5, 1)$

Algorithme avec Intervalles Critiques, Yao et al.



Ensemble d'intervalles $= (2/4, 1/5, 1, 5/7, 6/7)$

Algorithme avec Intervalles Critiques, Yao et al.



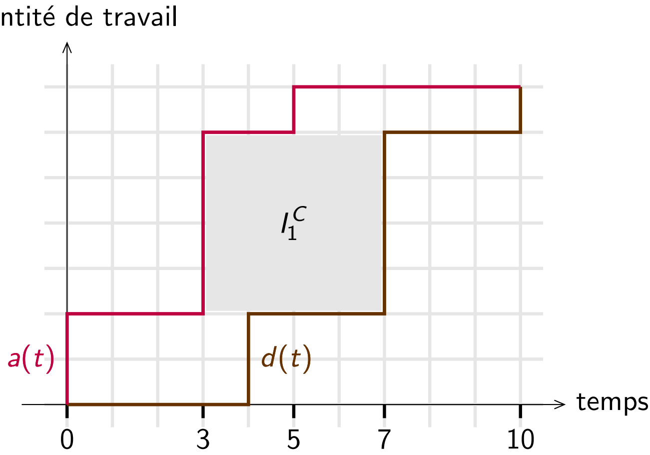
Ensemble d'intervalles $= (2/4, 1/5, \mathbf{1}, 5/7, 6/7, 7/10)$ **Max**

Algorithme avec Intervalles Critiques, Yao et al.

$J_3(r_3{=}5, c_3{=}1, d_3{=}5)$

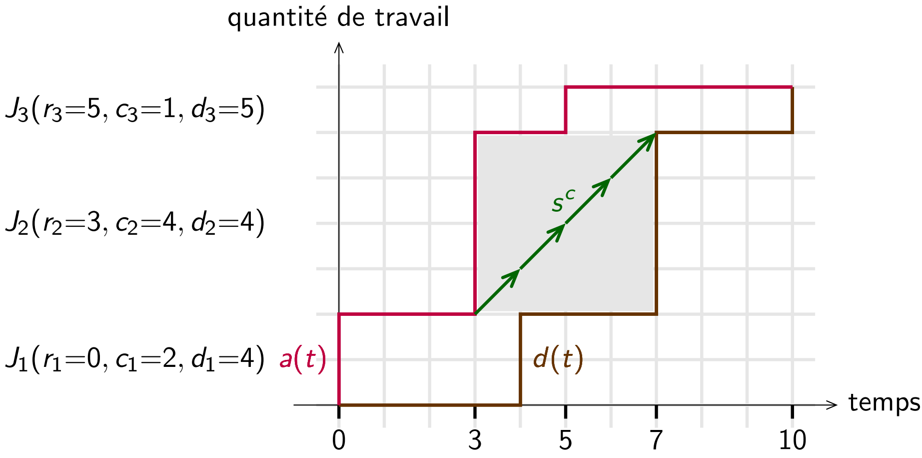
$J_2(r_2{=}3, c_2{=}4, d_2{=}4)$

$J_1(r_1{=}0, c_1{=}2, d_1{=}4)$



Algorithme avec Intervalles Critiques, Yao et al.

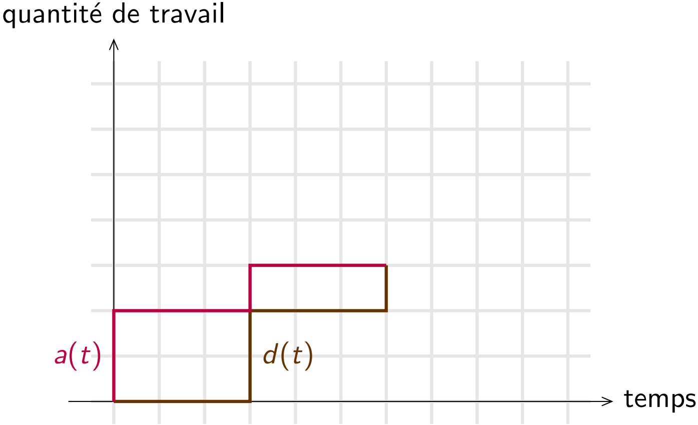
Vitesses choisies $(t_0, t_1, t_2, 1, 1, 1, 1, t_7, t_8, t_9)$



Sur l'intervalle critique, la vitesse optimale s^c est constante

Algorithme avec Intervalles Critiques, Yao et al.

Vitesses choisies $(t_0, t_1, t_2, 1, 1, 1, 1, t_7, t_8, t_9)$

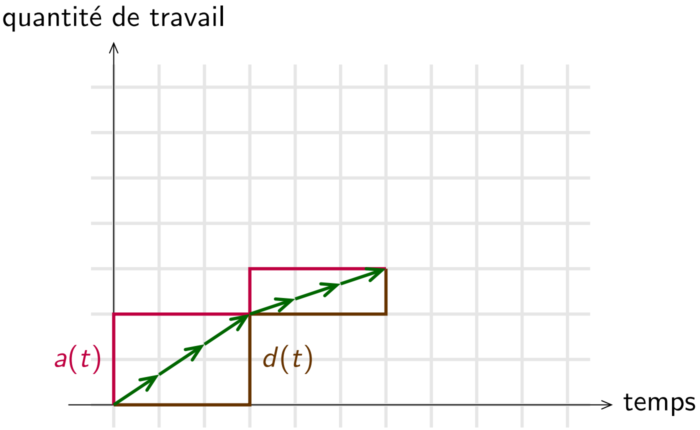


Suppression de l'intervalle critique

$\Rightarrow$ Nouvelle étude de l'intervalle critique sur le système restreint.

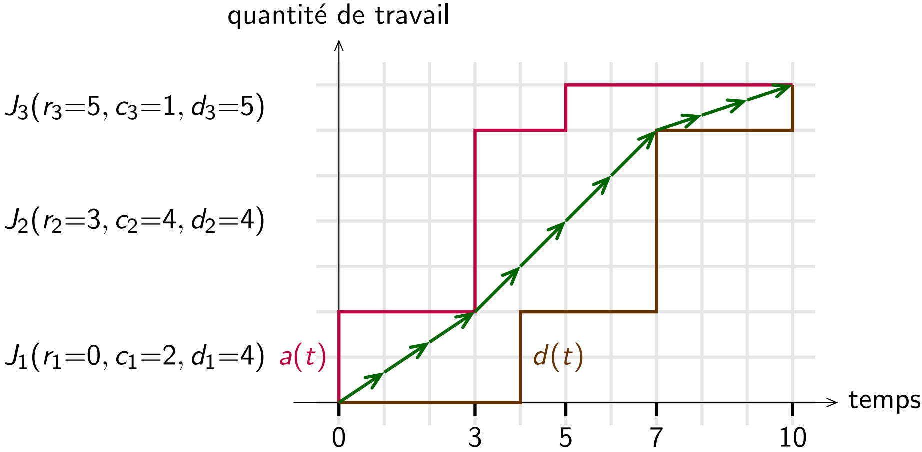
Algorithme avec Intervalles Critiques, Yao et al.

$\pi^* =$ Vitesses choisies $= (\frac{2}{3}, \frac{2}{3}, \frac{2}{3}, 1, 1, 1, 1, \frac{1}{3}, \frac{1}{3}, \frac{1}{3})$



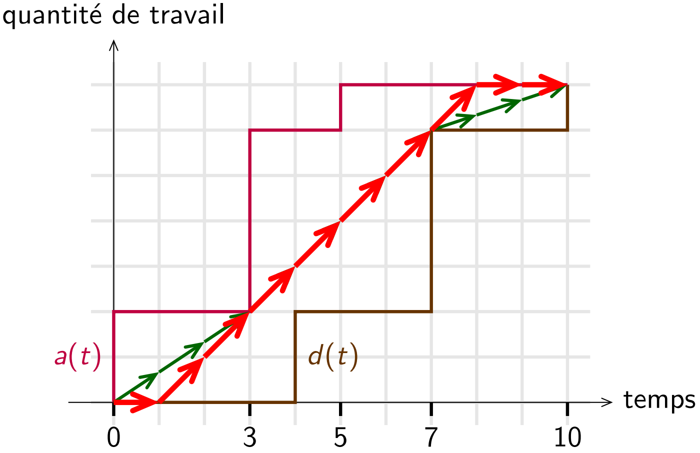
Algorithme avec Intervalles Critiques, Yao et al.

$\pi^* = \text{Vitesses choisies} = (\frac{2}{3}, \frac{2}{3}, \frac{2}{3}, 1, 1, 1, 1, \frac{1}{3}, \frac{1}{3}, \frac{1}{3})$



Algorithme avec Intervalles Critiques, Yao et al.

Vitesses disponibles: $\mathcal{S} = \{0, 1\}$
Vitesses choisies $\pi^* = (0, 1, 1, 1, 1, 1, 1, 1, 0, 0)$



Pour chaque intervalle critique: $s^c = \alpha\lfloor s^c \rfloor + (1-\alpha)\lfloor s^c + 1 \rfloor$.

Programmation Dynamique

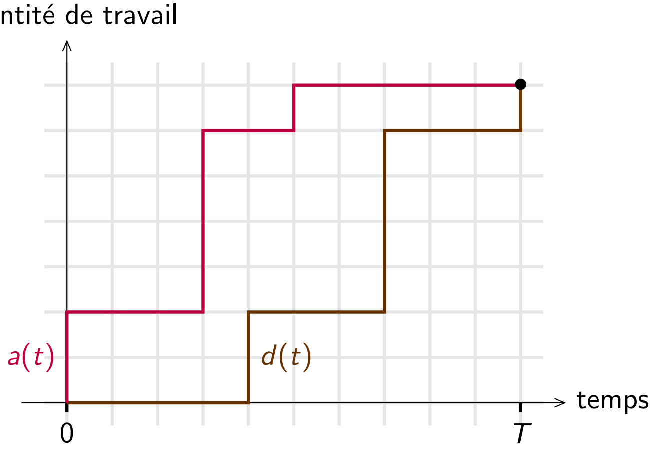
Programmation Dynamique

Vitesses disponibles: $\mathcal{S} = \{0, 1, 2\}$

$J_3(r_3{=}5, c_3{=}1, d_3{=}5)$

$J_2(r_2{=}3, c_2{=}4, d_2{=}4)$

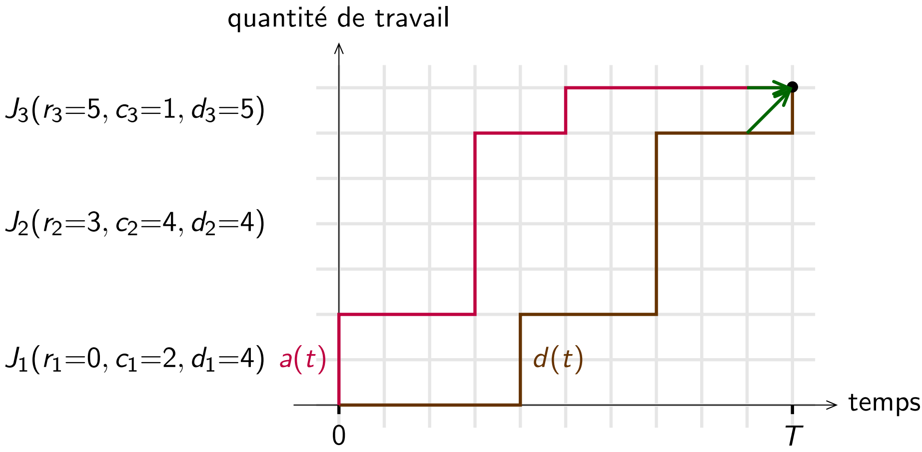
$J_1(r_1{=}0, c_1{=}2, d_1{=}4)$



Energie pour w en T: $E^*_T(w_T) = 0$

Programmation Dynamique

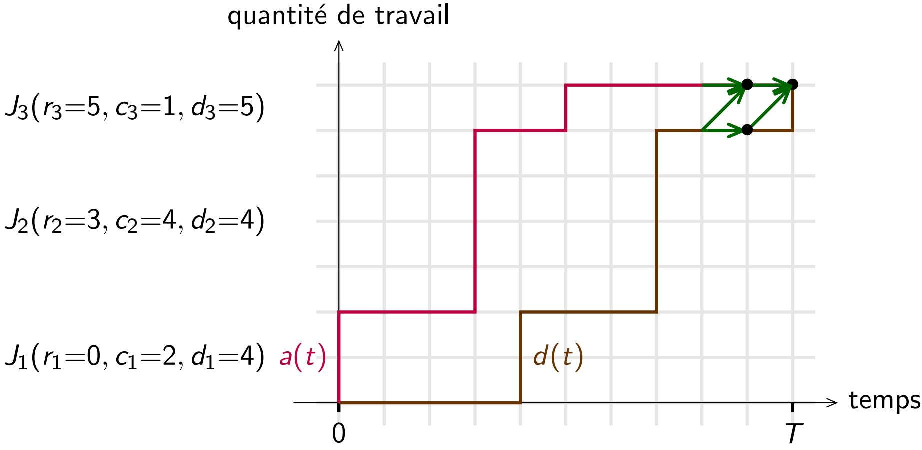
Vitesses disponibles: $\mathcal{S} = \{0, 1, 2\}$



Energie pour w en T-1: $E^*_{T-1}(w) \quad = \quad \min_{s \in \mathcal{S}} \left(P_{ower}(s) + E^*_T(w') \right)$

Programmation Dynamique

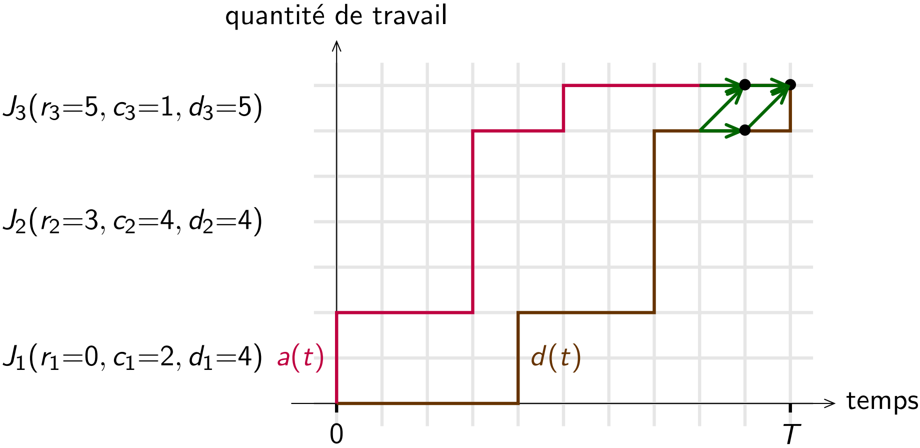
Vitesses disponibles: $\mathcal{S} = \{0, 1, 2\}$



Energie pour w en T-2: $E^*_{T-2}(w) \quad = \quad \min_{s \in \mathcal{S}} \left(P_{ower}(s) + E^*_{T-1}(w') \right)$

Programmation Dynamique

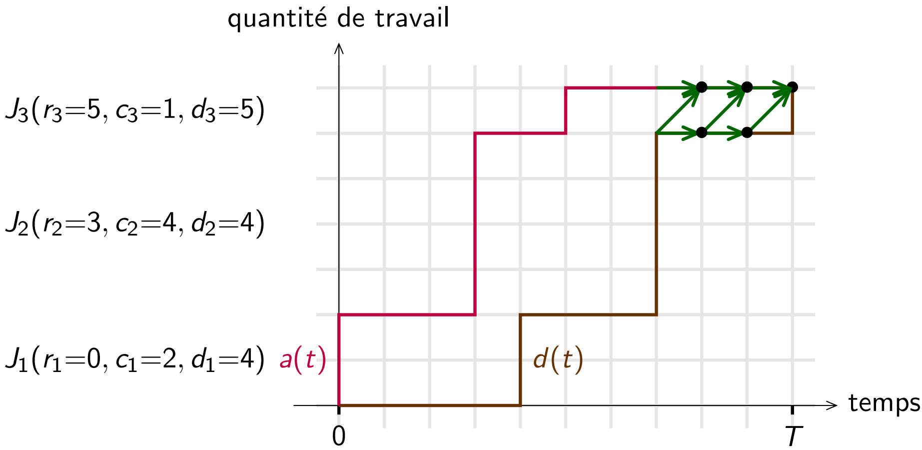
Vitesses disponibles: $\mathcal{S} = \{0, 1, 2\}$



Vitesse pour w en T-2: $s^*(T-2)(w) \quad = \quad \arg\min_{s \in \mathcal{S}} \left(P_{ower}(s) + E^*_{T-1}(w')\right)$

Programmation Dynamique

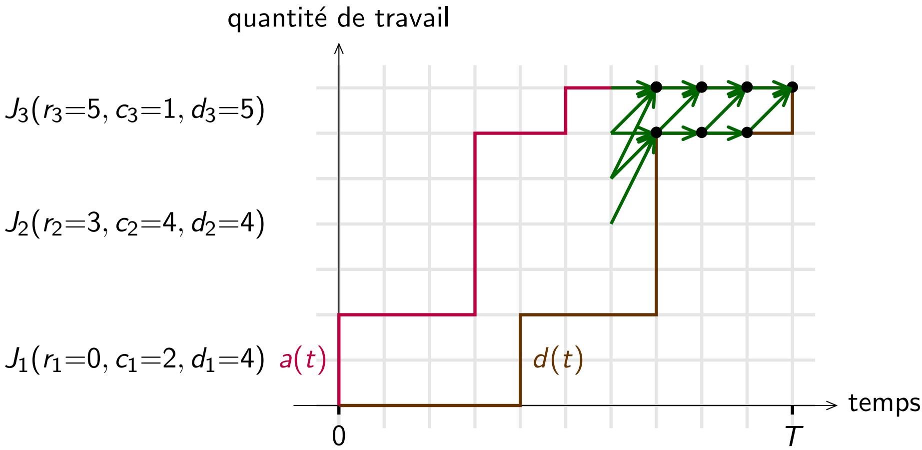
Vitesses disponibles: $\mathcal{S} = \{0, 1, 2\}$



Energie pour w en t-1: $E^*_{t-1}(w) \quad = \quad \min_{s \in \mathcal{S}} \left(P_{ower}(s) + E^*_t(w') \right)$

Programmation Dynamique

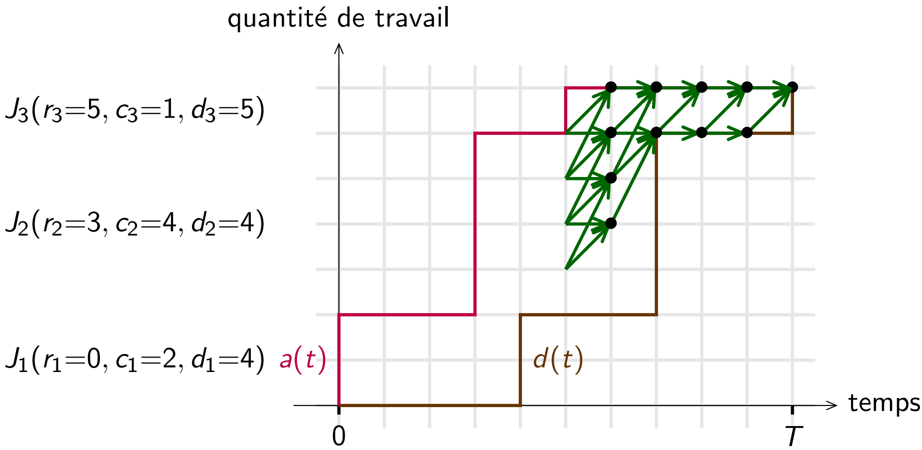
Vitesses disponibles: $\mathcal{S} = \{0, 1, 2\}$



Energie pour w en t-1: $E^*_{t-1}(w) \;=\; \min_{s\in\mathcal{S}} \left(P_{ower}(s) + E^*_t(w')\right)$

Programmation Dynamique

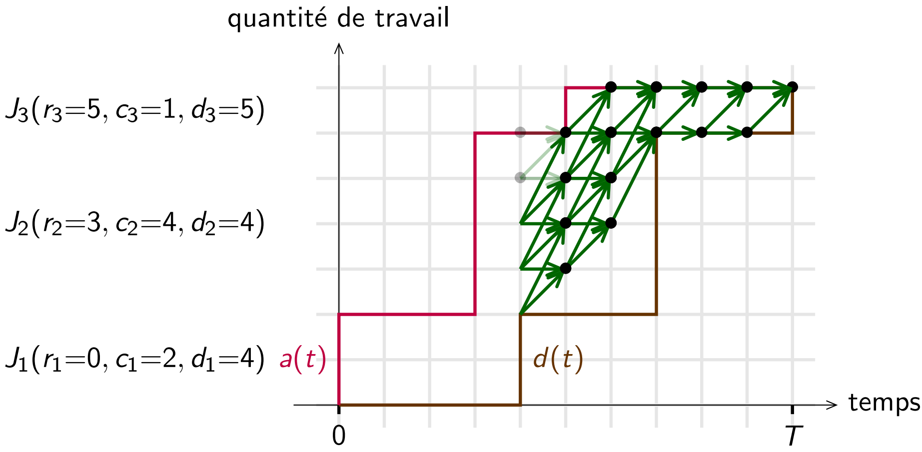
Vitesses disponibles: $\mathcal{S} = \{0, 1, 2\}$



Energie pour w en t-1: $E^*_{t-1}(w) \quad = \quad \min_{s \in \mathcal{S}} \left(P_{ower}(s) + E^*_t(w')\right)$

Programmation Dynamique

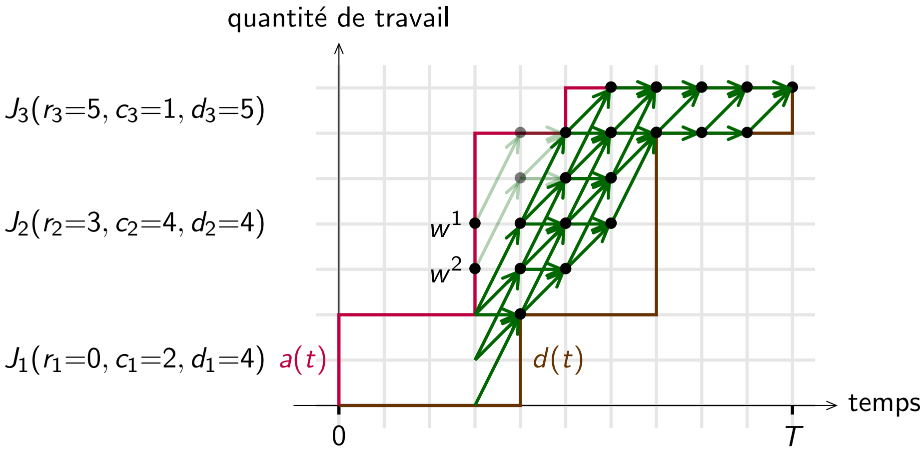
Vitesses disponibles: $\mathcal{S} = \{0, 1, 2\}$



Energie pour w en t-1: $E^*_{t-1}(w) \quad = \quad \min_{s \in \mathcal{S}} \left(P_{ower}(s) + E^*_t(w') \right)$

Programmation Dynamique

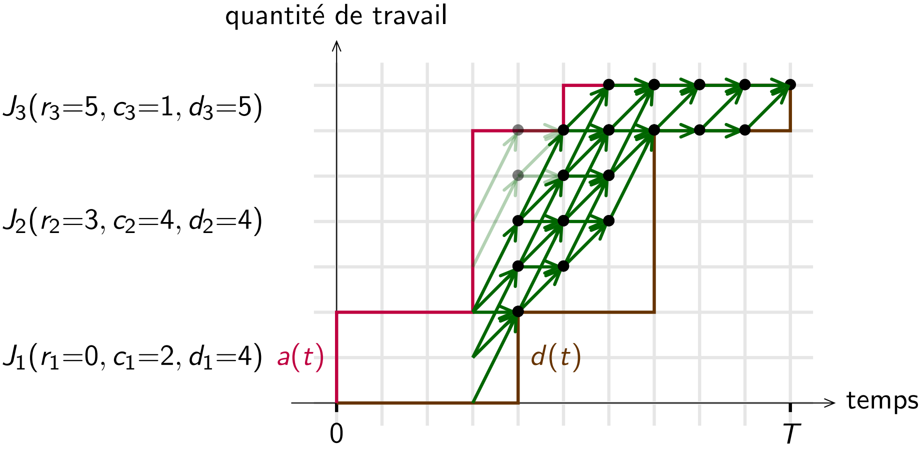
Vitesses disponibles: $\mathcal{S} = \{0, 1, 2\}$



Energie états inaccessibles: $E_t^*(w^1) = E_t^*(w^2) = +\infty$

Programmation Dynamique

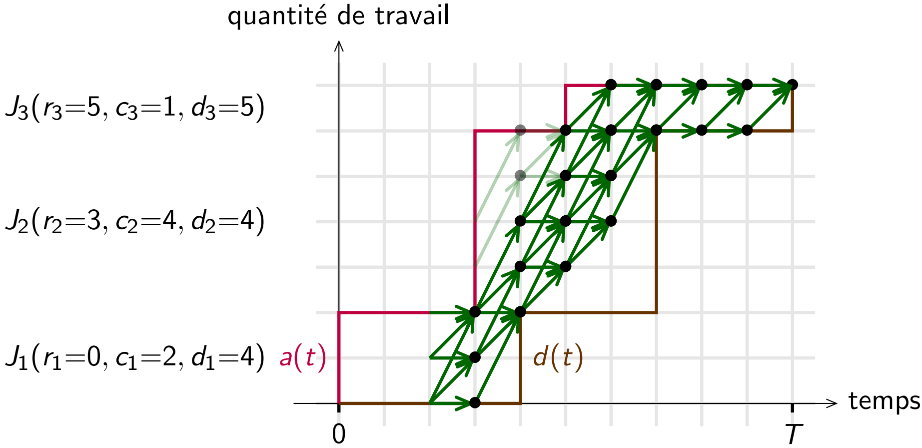
Vitesses disponibles: $\mathcal{S} = \{0, 1, 2\}$



Energie pour w en t-1: $E^*_{t-1}(w) \quad = \quad \min_{s \in \mathcal{S}} \left(P_{ower}(s) + E^*_t(w') \right)$

Programmation Dynamique

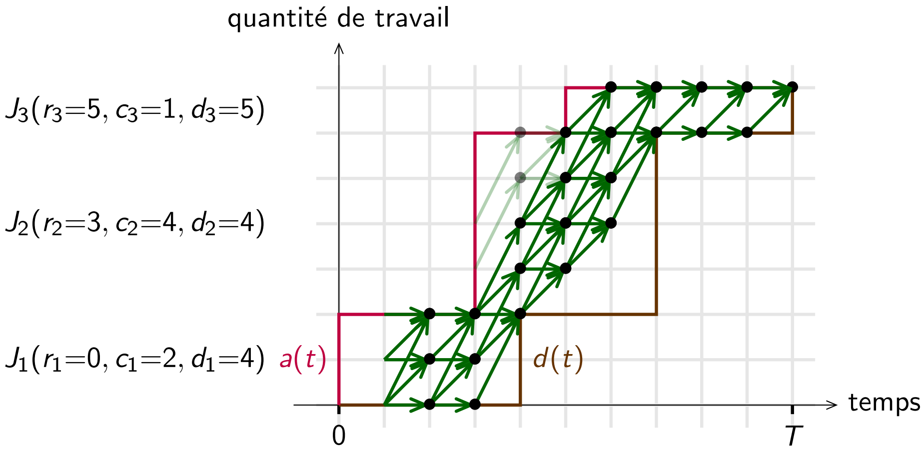
Vitesses disponibles: $\mathcal{S} = \{0, 1, 2\}$



Energie pour w en t-1: $E^*_{t-1}(w) \quad = \quad \min_{s \in \mathcal{S}} \left(P_{ower}(s) + E^*_t(w') \right)$

Programmation Dynamique

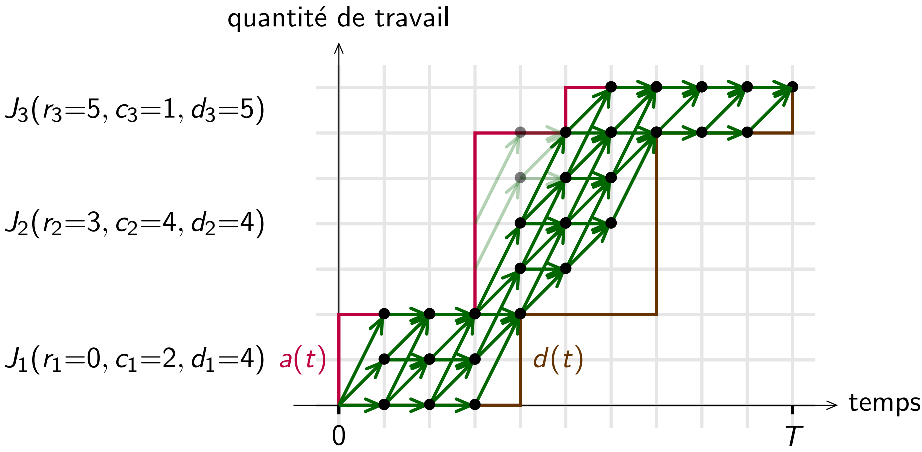
Vitesses disponibles: $\mathcal{S} = \{0, 1, 2\}$



Energie pour w en t-1: $E^*_{t-1}(w) \quad = \quad \min_{s \in \mathcal{S}} \left(P_{ower}(s) + E^*_t(w') \right)$

Programmation Dynamique

Vitesses disponibles: $\mathcal{S} = \{0, 1, 2\}$

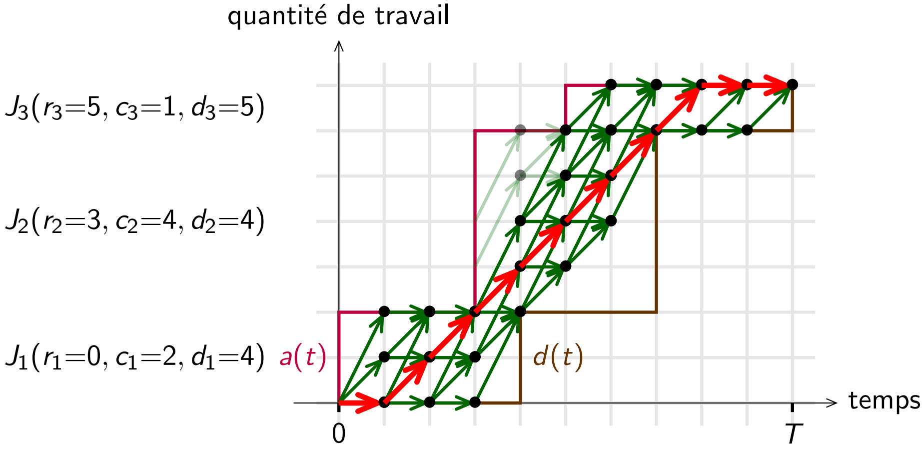


Vitesses Optimales obtenues après Programmation Dynamique:

$$\pi^* = (s_0^*, s_1^*, s_2^*, s_3^*, s_4^*, s_5^*, s_6^*, s_7^*, s_8^*, s_9^*) = (0, 1, 1, 1, 1, 1, 1, 1, 0, 0)$$

Programmation Dynamique

Vitesses disponibles: $\mathcal{S} = \{0, 1, 2\}$



Vitesses Optimales obtenues après Programmation Dynamique:

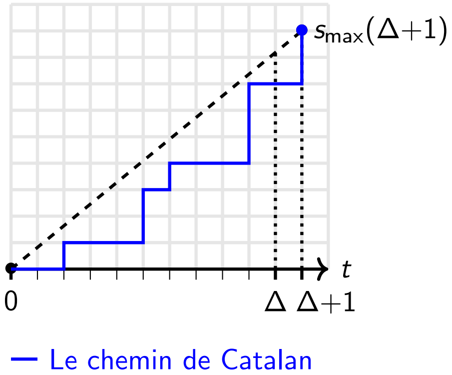
$$\pi^* = (s_0^*, s_1^*, s_2^*, s_3^*, s_4^*, s_5^*, s_6^*, s_7^*, s_8^*, s_9^*) = (0, 1, 1, 1, 1, 1, 1, 1, 0, 0)$$

Complexité

Théorème

La complexité en temps est en Kn, avec n le nombre de tâches et K, une constante, qui dépend de la vitesse maximale s_{max} et de l'échéance relative maximale Δ

L'espace d'états est en bijection avec l'espace des chemins de Catalan.



Complexité

Théorème

La complexité en temps est en Kn, avec n le nombre de tâches et K, une constante, qui dépend de la vitesse maximale $s_{\max}$ et de l'échéance relative maximale Δ

- Borne sur la taille de l'espace d'état:

$$Catalan(\Delta, s_{\max})$$

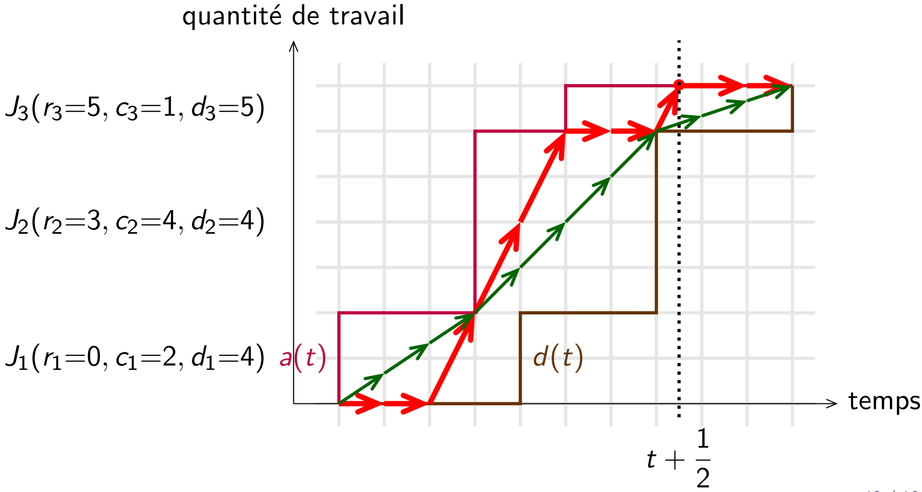
- Borne sur $T = \max_{i=1}^{n}\{r_i + d_i\}$, l'horizon de temps:

$$T \leq n\Delta$$

Cas Vitesses Non Consécutives

Vitesses disponibles: $\mathcal{S} = \{0, 2\}$

Vitesses choisies $\pi^* = (0, 0, 2, 2, 2, 0, 0, [2, 0], 0, 0)$



Cas Vitesses Non Consécutives: la solution

- Considérer des vitesses consécutives
$$s = \beta s_1 + (1-\beta)s_2, \text{ avec } \beta = \frac{s_2 - s}{s_2 - s_1}.$$
Puissance de la vitesse non disponible fixée à:
$$P_{ower}(s) = \beta P_{ower}(s_1) + (1-\beta)P_{ower}(s_2)$$

Cas Vitesses Non Consécutives: la solution

- Considérer des vitesses consécutives

$$s = \beta s_1 + (1-\beta)s_2, \text{ avec } \beta = \frac{s_2 - s}{s_2 - s_1}.$$

Puissance de la vitesse non disponible fixée à:

$$P_{ower}(s) = \beta P_{ower}(s_1) + (1-\beta) P_{ower}(s_2)$$

- Appliquer la programmation dynamique avec cet ensemble de vitesses.

Cas Vitesses Non Consécutives: la solution

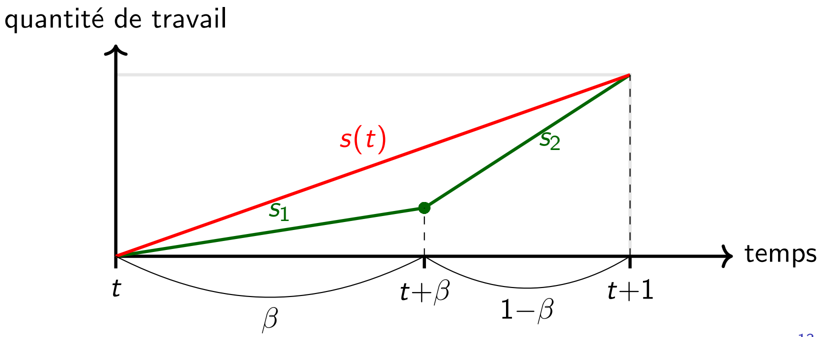
- Considérer des vitesses consécutives

$$s = \beta s_1 + (1-\beta)s_2, \text{ avec } \beta = \frac{s_2 - s}{s_2 - s_1}.$$

  Puissance de la vitesse non disponible fixée à:

$$P_{ower}(s) = \beta P_{ower}(s_1) + (1-\beta)P_{ower}(s_2)$$

- Appliquer la programmation dynamique avec cet ensemble de vitesses.
- Utilisation du Vdd-hopping pour simuler les vitesses indisponibles.



Conclusion

- Algorithme en compléxité linéaire, en $O(Kn)$
- K est potentiellement grand et dépend fortement de l'échéance relative maximale Δ.
- Généralisation: fonction de puissance non convexe, coût de changement de vitesses.
- Extension possible pour le cas en-ligne: Minimisation d'énergie avec un algorithme quasiment inchangé.

Extensions

1. Cas où la fonction de puissance est non convexe.

Algorithme de programmation dynamique valable pour une fonction de puissance non convexe.

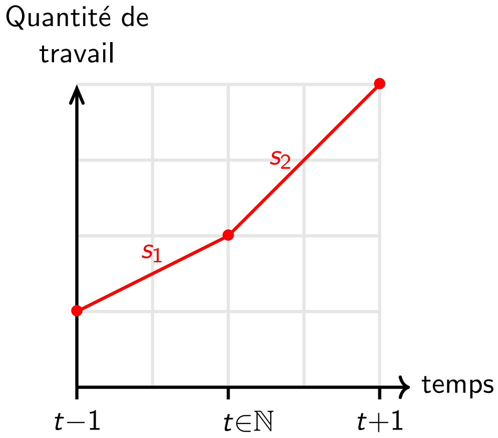
Extensions

1. Cas où la fonction de puissance est non convexe.

   Algorithme de programmation dynamique valable pour une fonction de puissance non convexe.

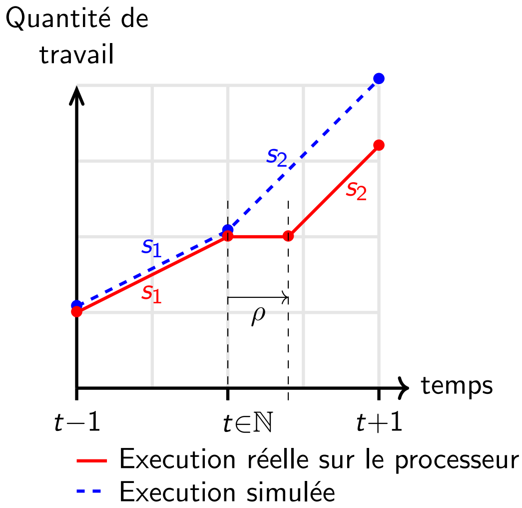
2. Coût de changement de vitesses

Coût de changement de vitesses



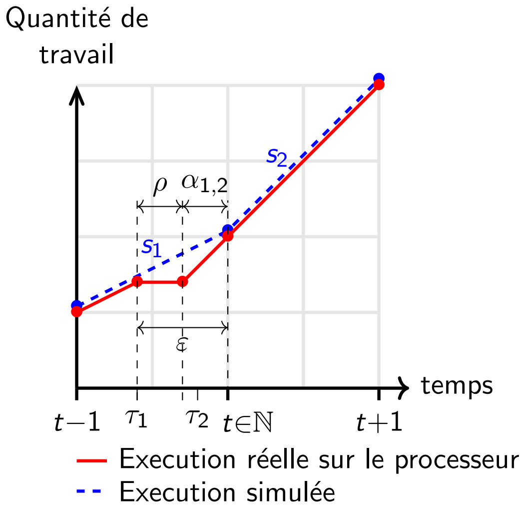
- $\rho =$ délai en temps pour changer la fréquence du processeur.

Coût de changement de vitesses



- ρ = délai en temps pour changer la fréquence du processeur.
- Quantité de travail exécuté inférieure $\Rightarrow$ problème de faisabilité
- Solution: décalage temporel du changement de vitesse

Coût de changement de vitesses



- ρ = délai en temps pour changer la fréquence du processeur.
- Quantité de travail exécuté inférieure $\Rightarrow$ problème de faisabilité
- Solution: décalage temporel du changement de vitesse
- $\varepsilon = \rho + \alpha_{1,2} = \dfrac{\rho\, s_2}{s_2 - s_1}$
- Coût additionnel en énergie:

$$\rho s_1 \left(\frac{P_{ower}(s_2) - P_{ower}(s_1)}{s_2 - s_1} \right)$$