Ordonnancement contraint par l'énergie de tâches probabilistes

Vandy Berten

vandy.berten@ulb.ac.be

Université Libre de Bruxelles

Table des matières

Section 1

Introduction

Table des matières I

Introduction

- Les systèmes embarqués ont souvent une source d'*énergie limitée*
- Les analyses "*worst-case*" donnent des *garanties*, mais *gaspillent* beaucoup de ressources
- Une connaissance plus fine des *durées d'exécution* est souvent disponible (ou calculable en ligne)
- Elle permet d'améliorer le comportement *moyen*
- But de l'exposé : voir comment on peut en tenir compte

Introduction

- Cet exposé ne va pas présenter les techniques les plus avancées de la recherche !
- On va voir les grandes idées qui sont mises en pratique
- Pour la simplicité de l'énoncé, on va faire des hypothèses simplificatrices fortes, parfois irréalistes
- La recherche actuelle est bien plus loin !
- On va se concentrer sur le processeur uniquement (pas la mémoire, les bus, les périphériques. . .)

Introduction

- En gros : aller *moins vite*, c'est *moins énergivore*
- Ce qui est souvent fait pour économiser de l'énergie :
  - Calculer la vitesse minimale pour atteindre les échéances dans le pire cas (Faire "comme si" toutes les tâches prenaient leur pire temps d'exécution)
  - Lorsque "par chance" une tâche se termine à l'avance, utiliser le *slack* pour économiser de l'énergie
- Avec les tâches probabilistes, on prend le problème *à la racine*, pour faire des économies d'énergie *moyennes*
- Idée : consommer beaucoup dans les cas très rares, pour économiser dans les cas fréquents

Analogie

- Supposons un démarcheur vendant chaque jour un article
- Il passe par *3 clients*, espacés chacun de *30 km*
- Il a *une heure* pour vendre son article. Dès qu'il l'a vendu, sa journée est terminée
- Le *1er client* a *50%* de chances d'acheter, le *2ème* (s'il y arrive) *60%*
- Il arrête sa journée après le 3ème, qu'il ait ou non acheté
- Proba de ne faire qu'*une étape* : $0,5$; *2 étapes* : $(1 - 0,5) * 0,6 = 0,3$; *3 étapes* : $1 - 0,5 - 0,3 = 0,2$
- On néglige son temps de démarchage, on ne tient pas compte du retour
- Sa voiture consomme $10\ell/100$ à 100km/h, 9 à 90km/h, ...
- À *quelle vitesse* rouler pour minimiser son carburant ?

Analogie : vitesse constante

- Stratégie simple : rouler à 90km/h.
- Consommation :

étape	1	2	3
vitesse	90	90	90
conso	2,7	2,7	2,7

- Consommation moyenne :

$$2,7 \times 0,5 + 5,4 \times 0,3 + 8,1 \times 0,2 = 4,59\ell$$

Analogie : vitesse croissante

- Stratégie de “procrastination” : on démarre lentement, on accélère si nécessaire (on espère ne pas devoir aller jusqu’au bout).
- Stratégie “75 – 90 – 112,5” :

étape	1	2	3
vitesse	75	90	112,5
conso	2,25	2,7	3,375

$\Rightarrow$ Consommation moyenne : $4,275\ell$. Gain de $6,7\%$

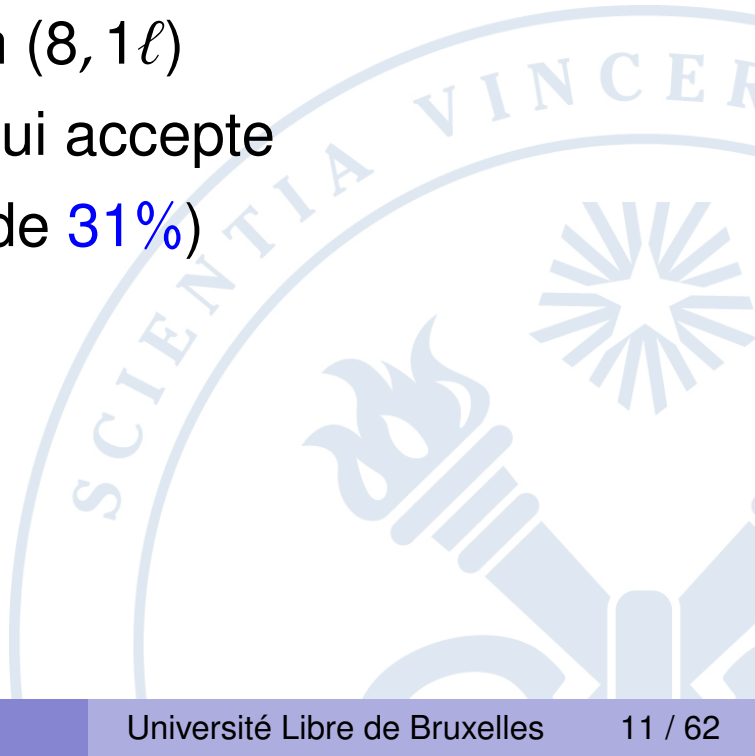
- Stratégie “65 – 90 – 146” :

étape	1	2	3
vitesse	65	90	146
conso	1,95	2,7	4,3875

$\Rightarrow$ Consommation moyenne : $4,1775\ell$. Gain de $8,9\%$

Analogie : Stratégie clairvoyante

- Avant de partir, le démarcheur téléphone
- Si le 1er client accepte, il part à 30km/h ($0,9\ell$)
- Si le 2ème client accepte, il part à 60km/h ($3,6\ell$)
- Si le 3ème client accepte, il part à 90km/h ($8,1\ell$)
- On suppose qu'il y a toujours un client qui accepte
- Consommation moyenne : $3,15\ell$ (Gain de 31%)



Section 2

Modèle

Table des matières I

Modèle déterministe

- Un travail (job) $\mathcal{J} =< r, c, d >$: processus arrivant à l'instant r, devant exécuter (maximum) c cycles, avant l'échéance d
- Une tâche $\tau =< O, C, D, T >$: un *générateur de jobs*, qui produit des jobs similaires avec une certaine périodicité :
  - Premier job en O, les suivants espacés de T
  - C cycles à exécuter
  - Échéance : D après l'arrivée
- Un système de tâche τ : ensemble de tâches

Modèle probabiliste

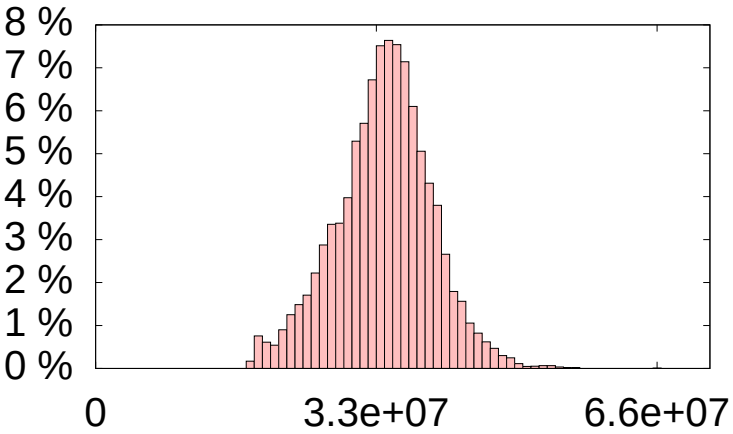
- Au moins une caractéristique est donnée par une *variable aléatoire*
- On va uniquement considérer le *temps d'exécution*
- $c(x)$: probabilité de demander x cycles (densité de probabilité)
- $C(x) \stackrel{\text{def}}{=} \sum_{y \leq x} c(y)$: probabilité de demander x cycles ou moins (probabilité cumulée)
- $\mathcal{W} \stackrel{\text{def}}{=} \max x : c(x) > 0$ (WCEC : *Worst Case Execution Cycles*)
- $C(\mathcal{W}) = 1$
- $\sum_x c(x) = 1$
- On peut adapter la granularité : au lieu de parler de *cycles*, on aura des *groupes de cycles*

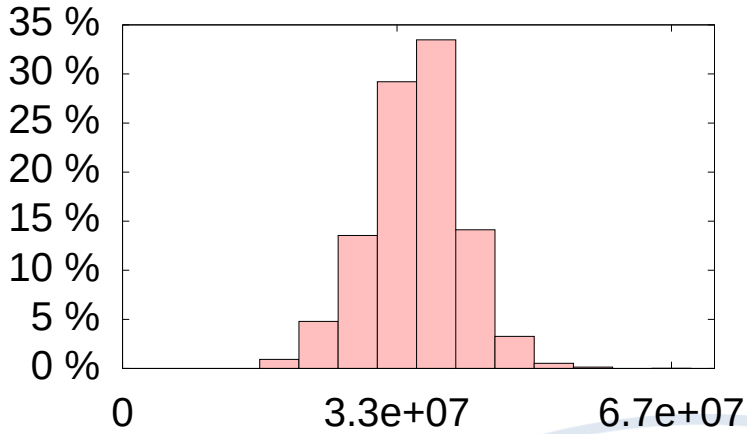
Distribution

50 groupes

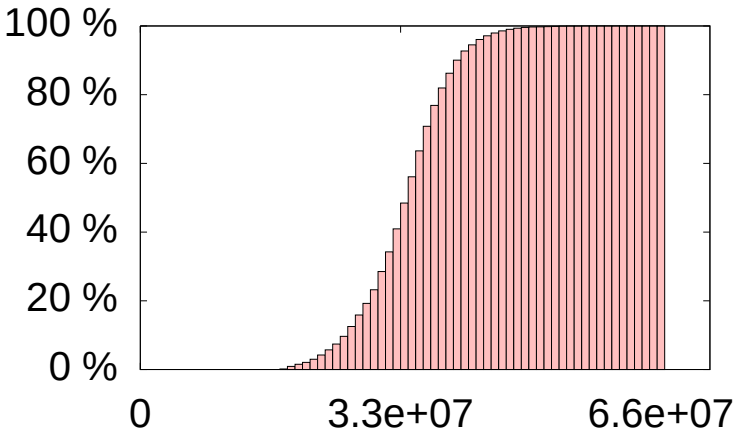
10 groupes

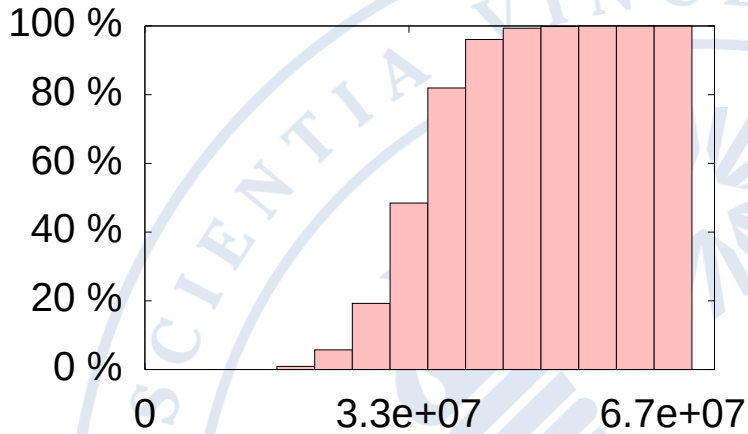
$c(x)$





$C(x)$





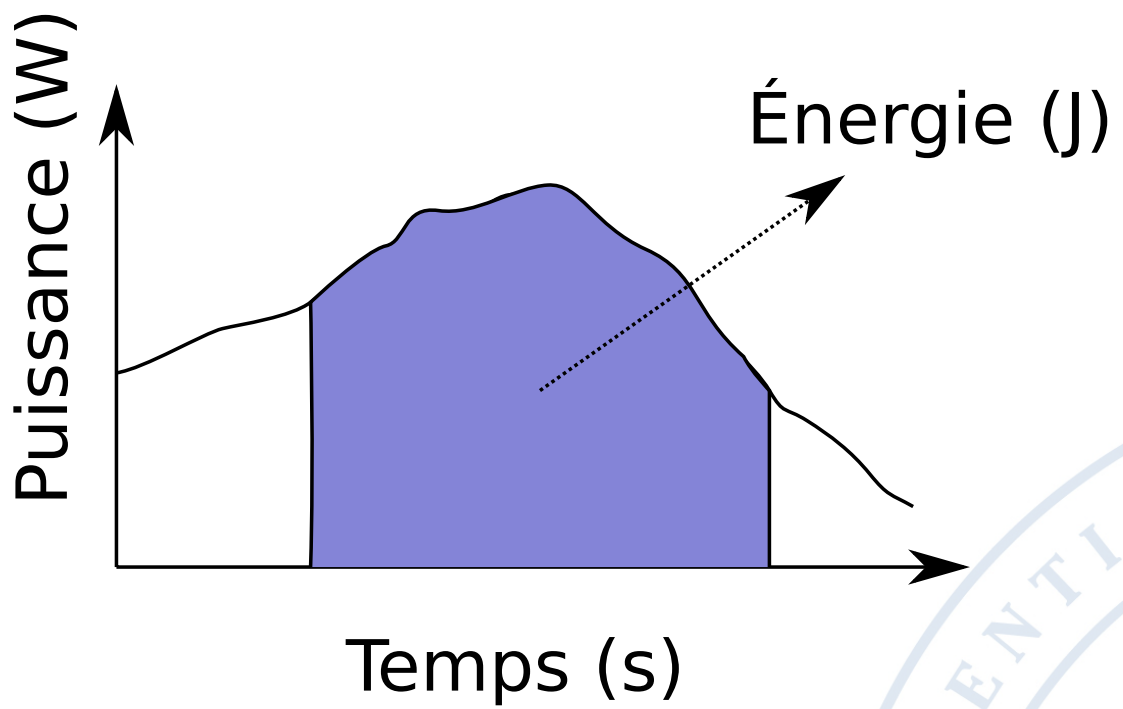
Puissance et Énergie

- Puissance :
  - Un processeur est un appareil travaillant à une *puissance* donnée
  - Unité de puissance : *Watt (W)*
  - Correspond à un appareil consommant un courant d'un ampère (A) lorsqu'il est soumis à une tension d'un volt (V)
  - Puissance : mesure *instantanée* de la consommation
- Énergie :
  - Énergie : Puissance $\times$ temps
  - Mesure *totale* de la consommation sur un *intervalle de temps donné*
  - Unité d'énergie : Joule (J)
  - Correspond à une puissance d'un watt pendant une seconde :

$$1J = 1W \times 1s$$

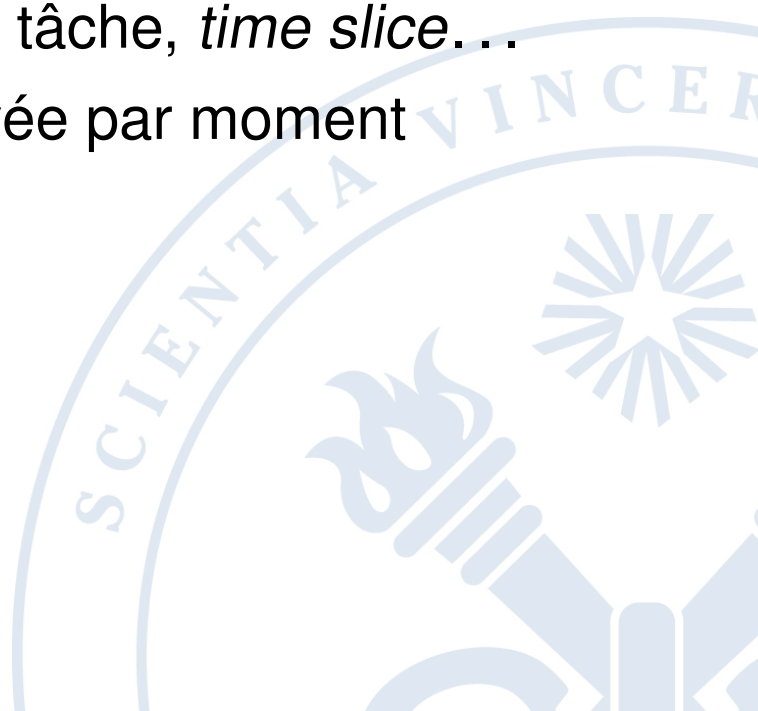
  - Grandes quantités : 1kWh (1000 watts pendant 1 heure) = $3.6e^6$ J

Puissance et Énergie



Objectif énergétique

- Notre objectif : sur un *intervalle donné*, réduire l'*énergie* consommée
- Intervalle : hyper-période, période d'une tâche, *time slice*...
- Il se peut que la *puissance* soit très élevée par moment



Processeurs DVFS

- Les processeurs DVFS (Dynamic Voltage and Frequency Scaling) peuvent adapter leur vitesse et puissance
- Une *fréquence* σ (en Hz) nécessite une *puissance* $p(\sigma)$ (en W)
- En théorie :

$$p(\sigma) \simeq \sigma^{\alpha}$$

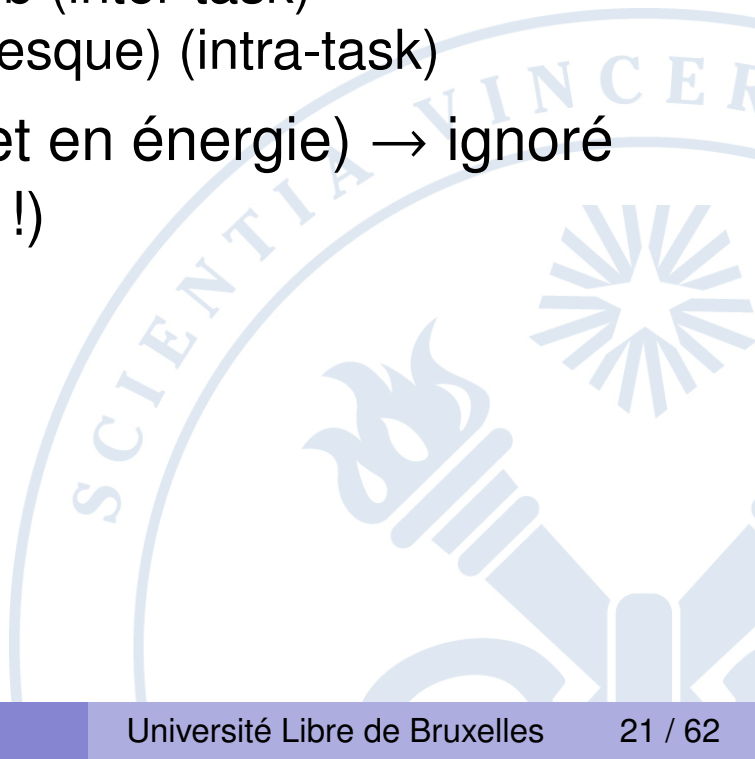
avec $\alpha \in [2, 3]$ (caractéristique du CPU)

- En pratique : on a une liste de M modes

$$\{< \sigma_1, p_1 >, \ldots, < \sigma_M, p_M >\}$$

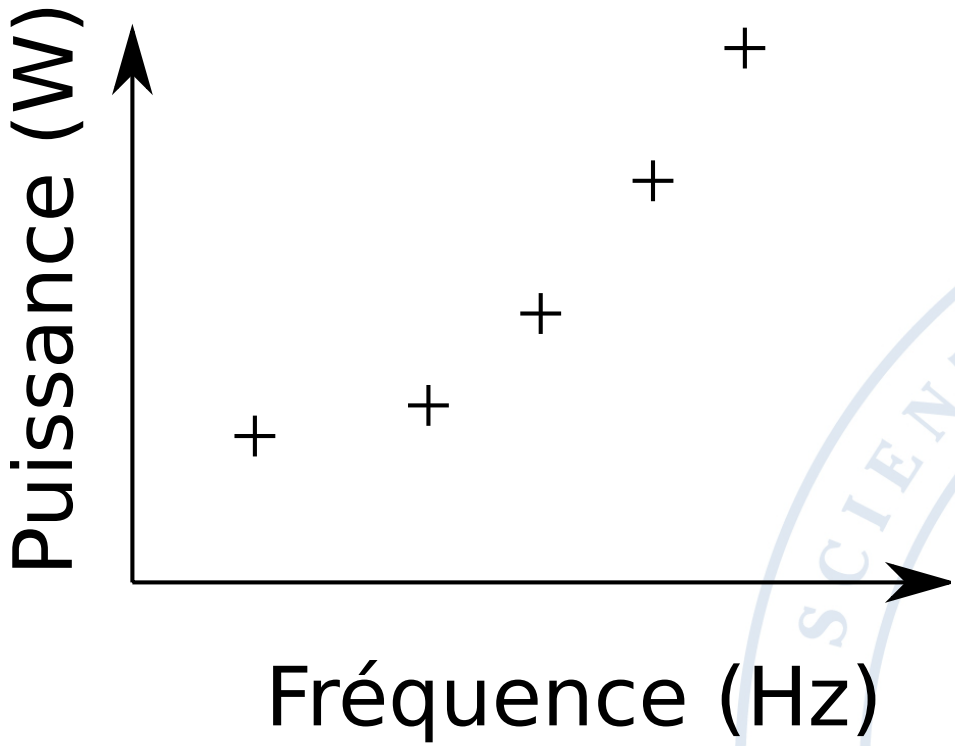
Modes de fonctionnement

- En fonction du modèle/de l'architecture, un changement de mode peut se faire :
  - Statiquement, au démarrage du système
  - Dynamiquement, au démarrage d'un job (inter-task)
  - Dynamiquement, à tout moment (ou presque) (intra-task)
- Changer de mode a un coût (en temps et en énergie) → ignoré dans cet exposé (pas dans la recherche !)



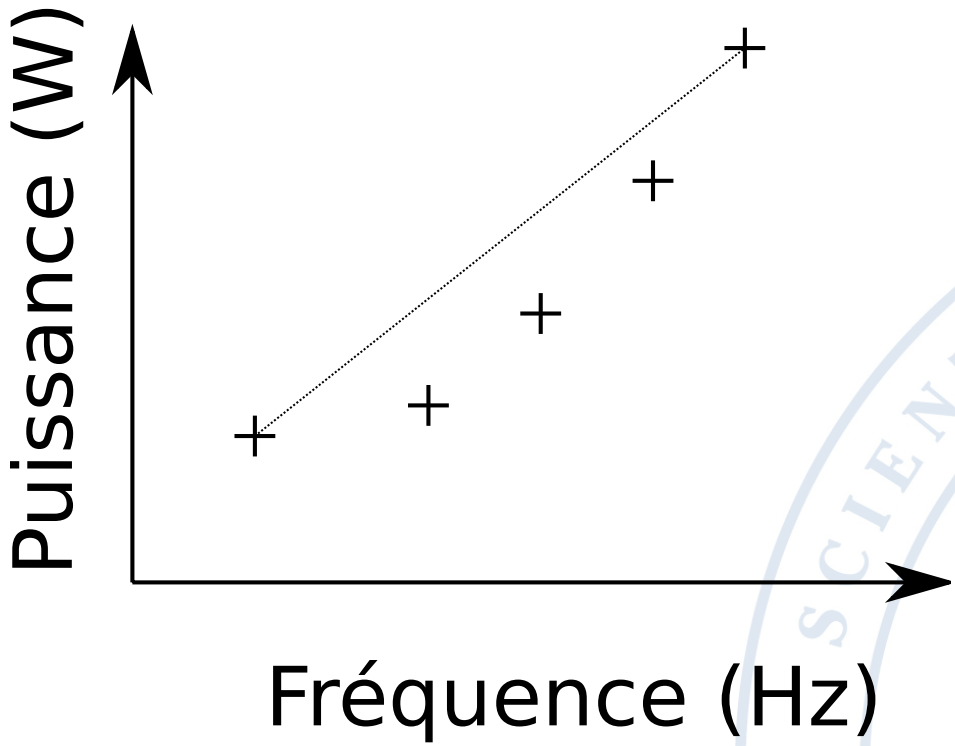
Convexité

- En théorie, la fonction $p(\sigma)$ est *convexe* en σ
- En pratique : souvent presque vrai, avec quelques écarts ...
- Intuition : puissance augmente plus vite que la fréquence.
- Fréquence multipliée par $A \Rightarrow$ Puissance multipliée par $B > A$



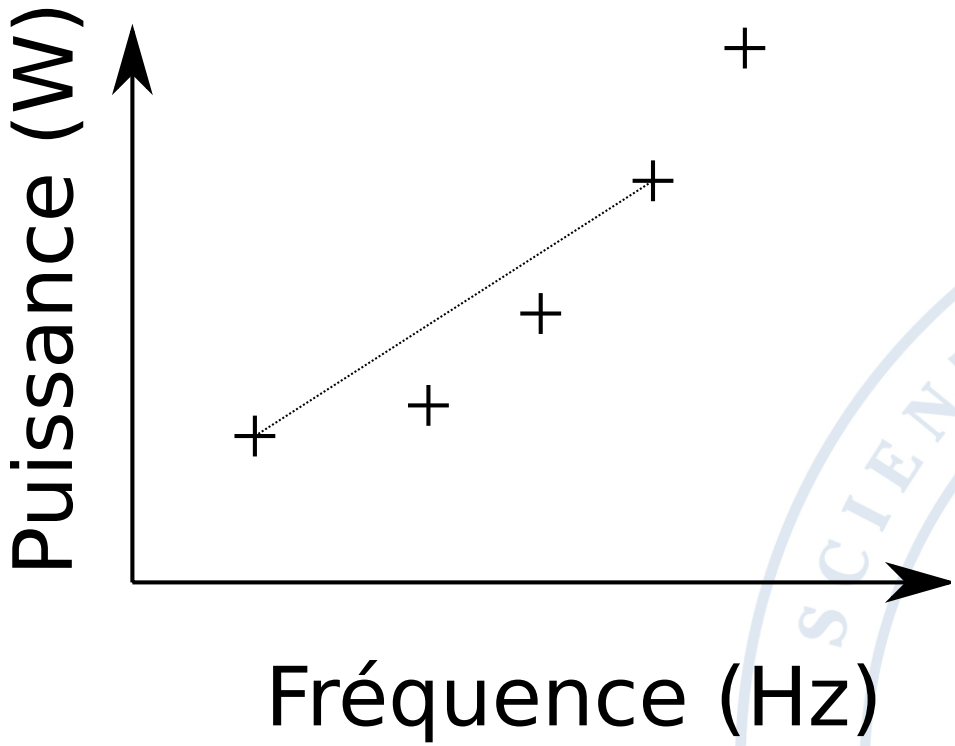
Convexité

- En théorie, la fonction $p(\sigma)$ est *convexe* en σ
- En pratique : souvent presque vrai, avec quelques écarts ...
- Intuition : puissance augmente plus vite que la fréquence.
- Fréquence multipliée par $A \Rightarrow$ Puissance multipliée par $B > A$



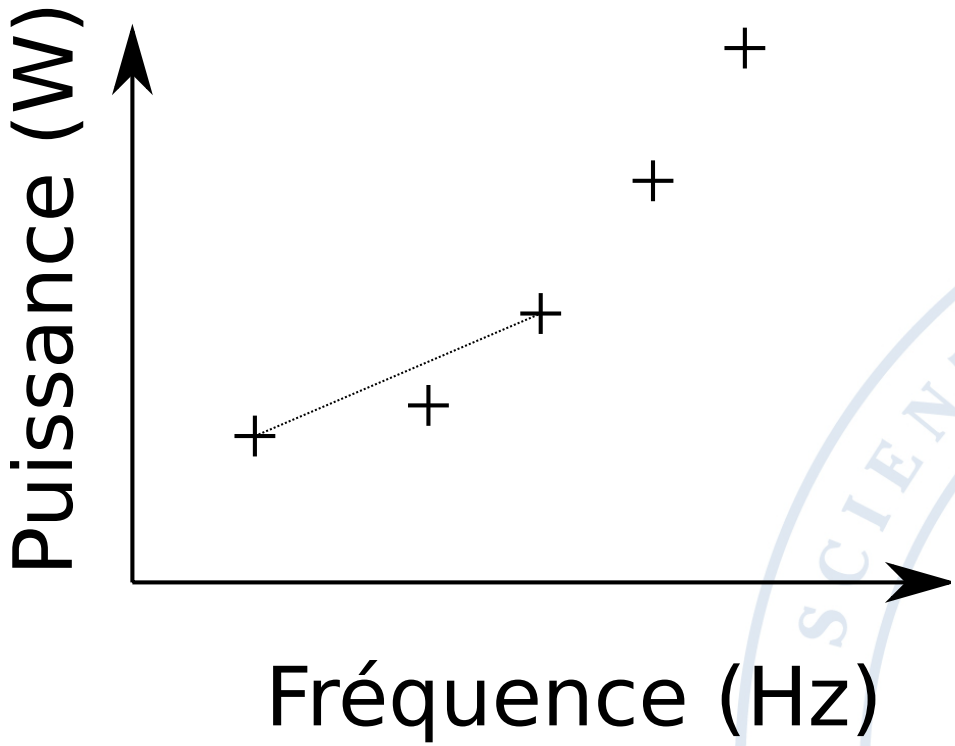
Convexité

- En théorie, la fonction $p(\sigma)$ est *convexe* en σ
- En pratique : souvent presque vrai, avec quelques écarts ...
- Intuition : puissance augmente plus vite que la fréquence.
- Fréquence multipliée par $A \Rightarrow$ Puissance multipliée par $B > A$



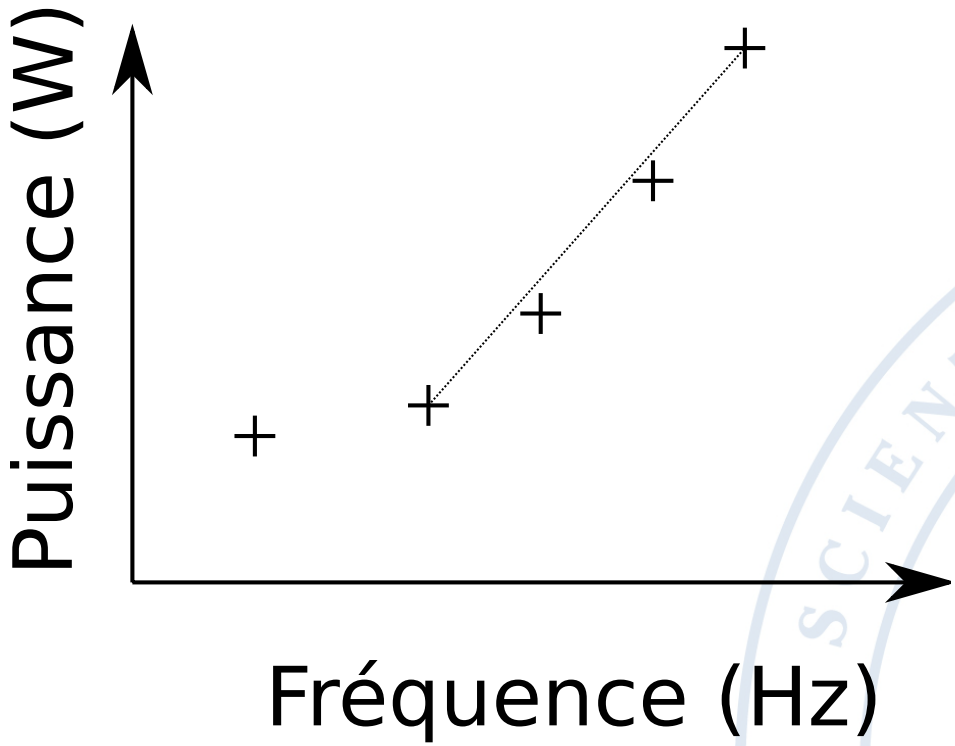
Convexité

- En théorie, la fonction $p(\sigma)$ est *convexe* en σ
- En pratique : souvent presque vrai, avec quelques écarts ...
- Intuition : puissance augmente plus vite que la fréquence.
- Fréquence multipliée par $A \Rightarrow$ Puissance multipliée par $B > A$



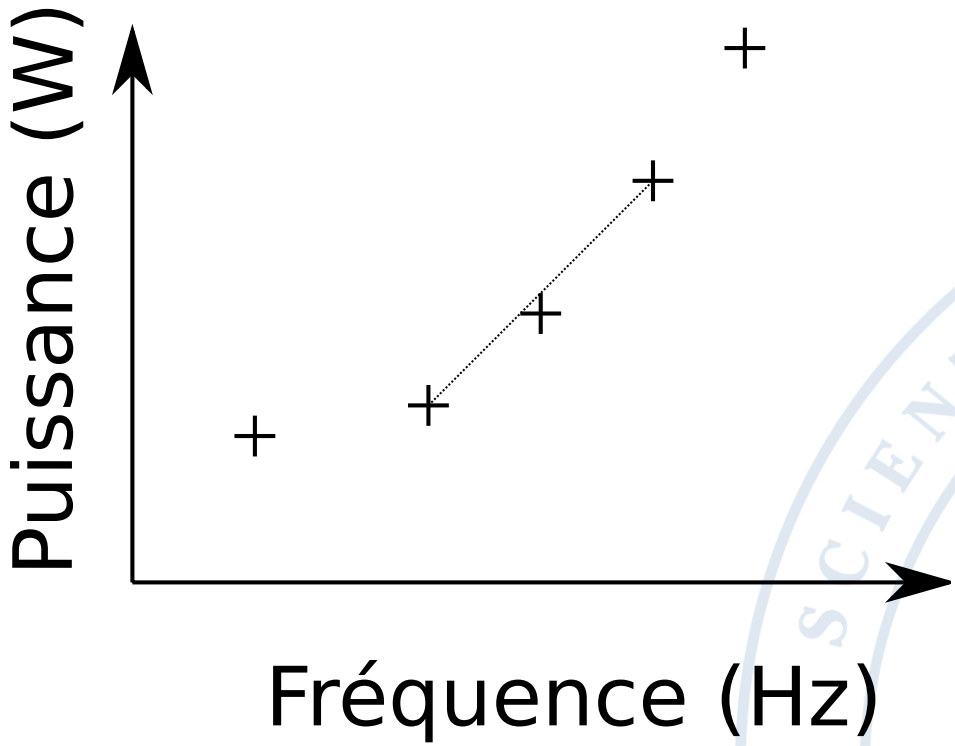
Convexité

- En théorie, la fonction $p(\sigma)$ est *convexe* en σ
- En pratique : souvent presque vrai, avec quelques écarts ...
- Intuition : puissance augmente plus vite que la fréquence.
- Fréquence multipliée par $A \Rightarrow$ Puissance multipliée par $B > A$



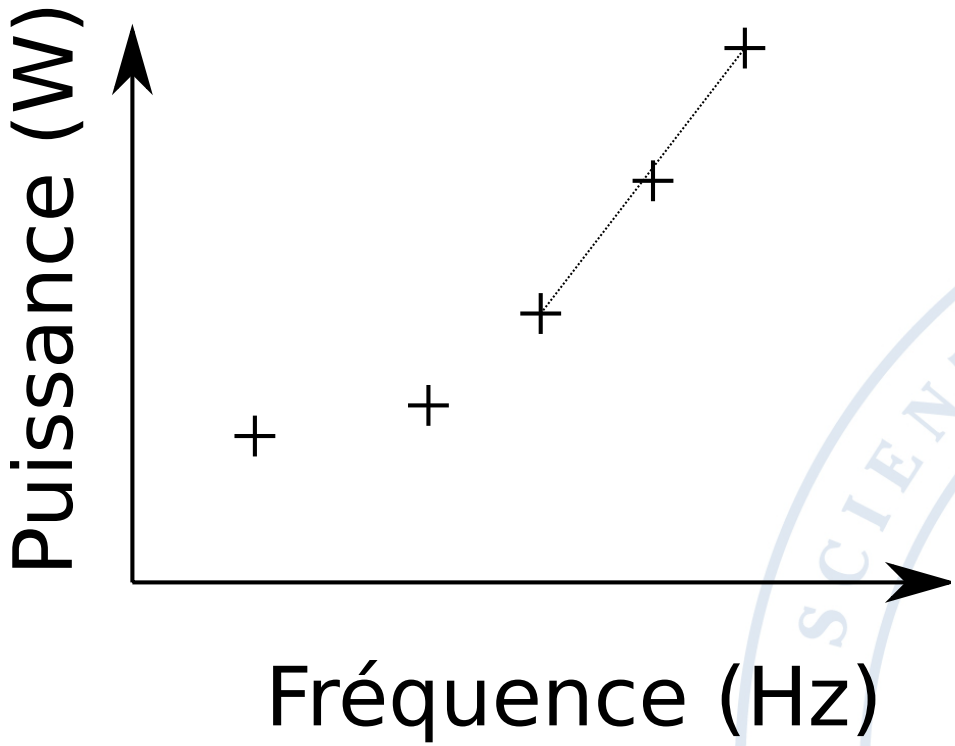
Convexité

- En théorie, la fonction $p(\sigma)$ est *convexe* en σ
- En pratique : souvent presque vrai, avec quelques écarts ...
- Intuition : puissance augmente plus vite que la fréquence.
- Fréquence multipliée par $A \Rightarrow$ Puissance multipliée par $B > A$



Convexité

- En théorie, la fonction $p(\sigma)$ est *convexe* en σ
- En pratique : souvent presque vrai, avec quelques écarts ...
- Intuition : puissance augmente plus vite que la fréquence.
- Fréquence multipliée par $A \Rightarrow$ Puissance multipliée par $B > A$



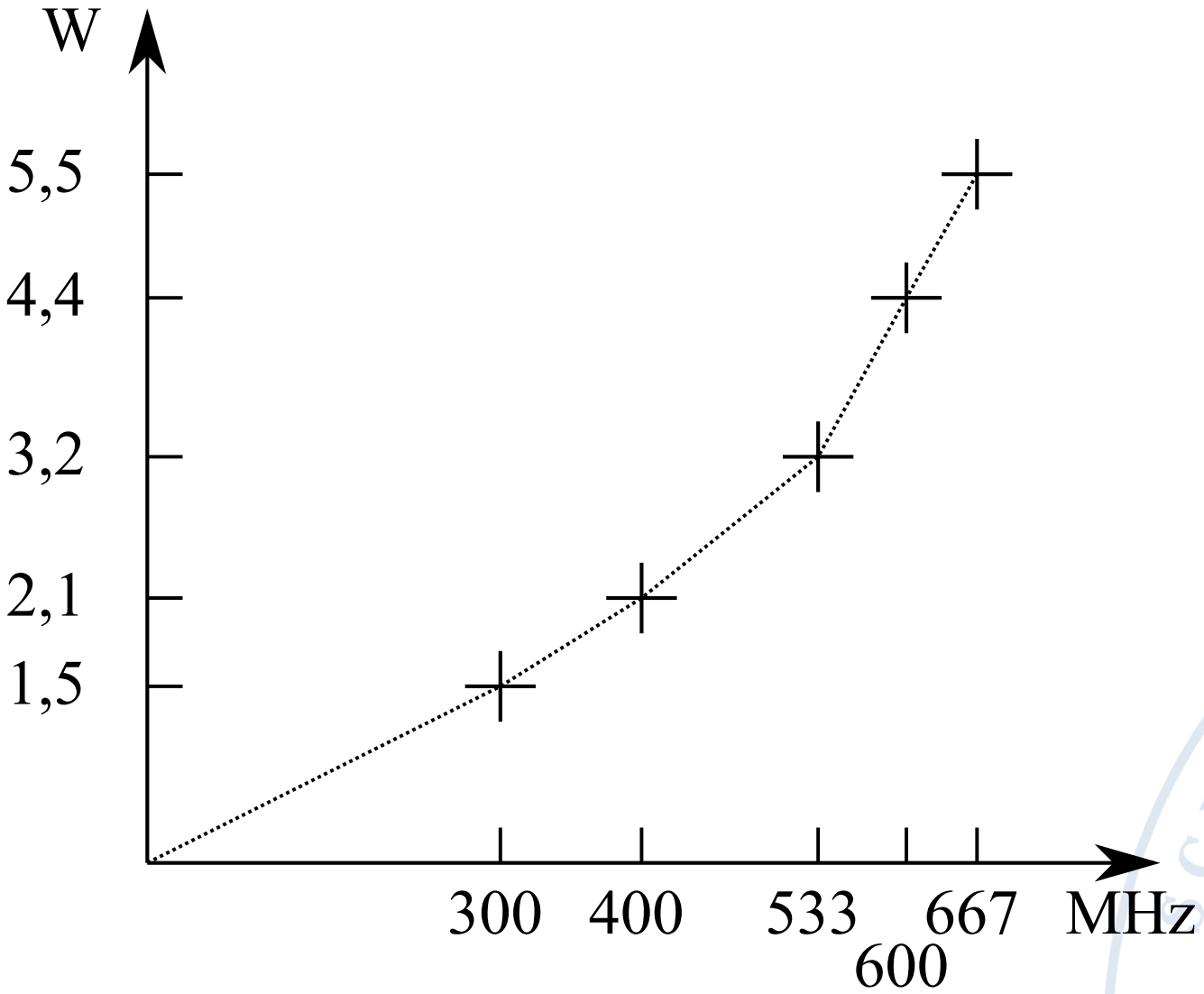
Convexité

- Si l'ensemble $\{(0,0),(\sigma_1,p_1),...,(\sigma_M,p_M)\}$ est (strictement) convexe, on a :

$$\frac{p_i}{\sigma_i} < \frac{p_{i+1}}{\sigma_{i+1}} \quad \forall i$$

- Or, $\frac{p_i}{\sigma_i}$ = l'énergie nécessaire pour exécuter un (bloc de) cycle(s) dans le mode i
- Donc basse fréquence $\Rightarrow$ moins d'énergie dépensée pour le job

Exemple : processeur Crusoe LongRun

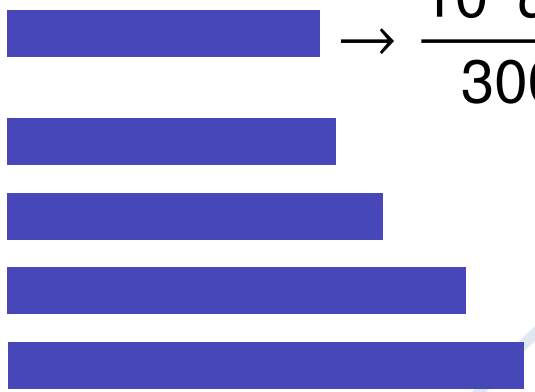


	MHz	W
1	300	1,5
2	400	2,1
3	533	3,2
4	600	4,4
5	667	5,5

Exemple : processeur Crusoe LongRun

Énergie nécessaire pour exécuter 10^9 cycles :

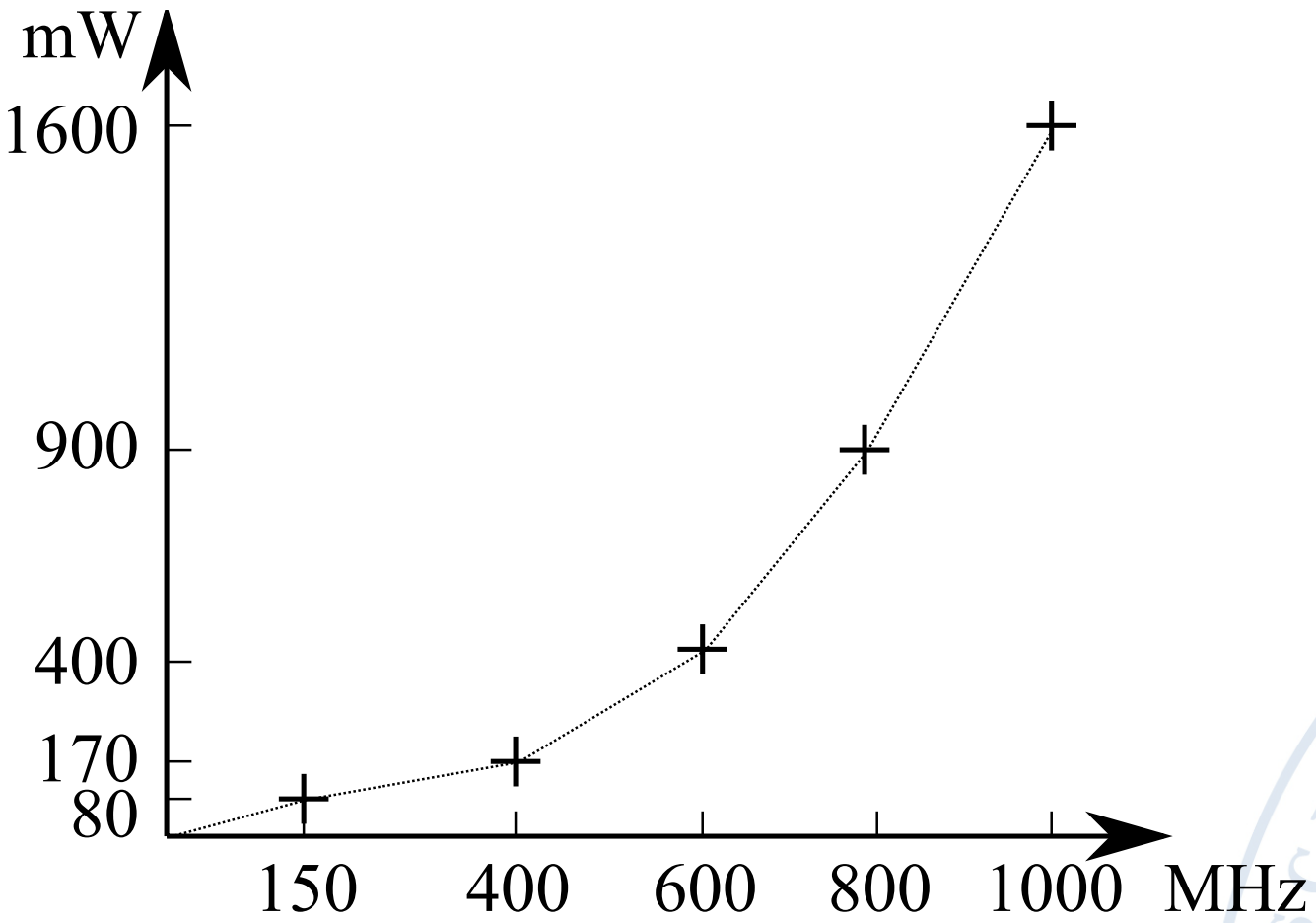
	MHz	W	J
1	300	1,5	5,000
2	400	2,1	5,250
3	533	3,2	6,004
4	600	4,4	7,333
5	667	5,5	8,246



$$\rightarrow \frac{10^9 c \times 1,5W}{300.10^6 c/s}$$

Hypothèse : x sec. à fréquence $\sigma_1 \rightarrow x\frac{\sigma_1}{\sigma_2}$ sec. à fréquence σ_2

Exemple : processeur Intel XScale

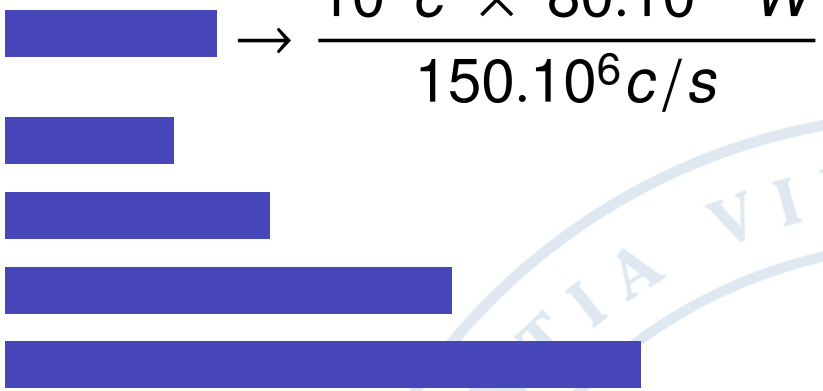


	MHz	mW
1	150	80
2	400	170
3	600	400
4	800	900
5	1000	1600

Exemple : processeur Intel XScale

Énergie nécessaire pour exécuter 10^9 cycles :

	MHz	mW	J
1	150	80	0,533
2	400	170	0,425
3	600	400	0,666
4	800	900	1,125
5	1000	1600	1,600



$$\rightarrow \frac{10^9 c \times 80.10^{-3} W}{150.10^6 c/s}$$

Section 3

Durées connues

Table des matières I

Contexte

- On considère un système avec durée d'exécution variable, mais connue à l'arrivée du travail
- Exemples :
  - Décodage d'un flux vidéo déjà joué
  - Les données d'entrée permettent de connaître à l'avance le temps de calcul
- Si on est capable d'utiliser cette info, on ne doit pas tenir compte de l'aspect probabiliste
- Sinon, on verra plus tard...
- On s'intéresse à un travail :
  - de durée (connue) C
  - à qui l'ordonnanceur a alloué une durée $A > \frac{C}{\sigma_M}$ pour s'exécuter

Cas convexe

- Dans le cas de modes convexes, on peut montrer [IY98] que :
  - Il faut s'exécuter le plus constamment possible
  - Il faut utiliser tout l'espace A (sauf si $A < \frac{C}{\sigma_1}$)
- Idéalement, on utilise la fréquence :

$$\sigma_{ideal} \stackrel{\text{def}}{=} \frac{C}{A}$$

- En général : pas possible/disponible !
- Si fréquence non modifiable au milieu d'un travail (*Inter-task DVFS*) :

$$\lceil \sigma_{ideal} \rceil_{\sigma}$$

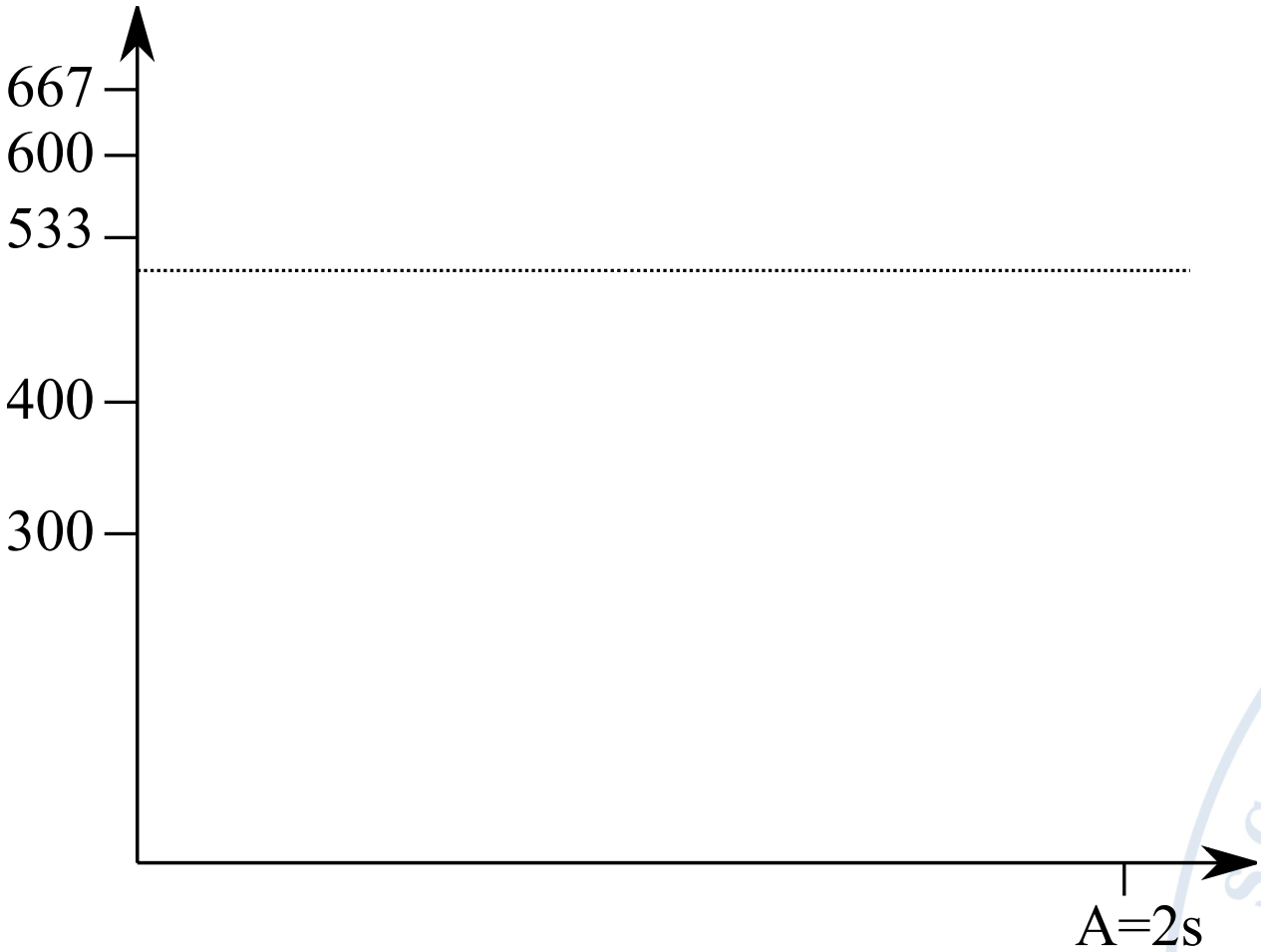
où $\lceil x \rceil_{\sigma} \stackrel{\text{def}}{=}$ la plus petite fréquence disponible supérieure à x

Cas convexe

- Si on peut changer de fréquence, on sait [IY98] que :
  - Il ne faut qu'un changement
  - Une phase à $\sigma_L = \lfloor \sigma_{ideal} \rfloor_\sigma$, puis une phase à $\sigma_H = \lceil \sigma_{ideal} \rceil_\sigma$
  - On exécute la première (resp. seconde) phase durant ℓ (resp. $A - \ell$), tel que
$$\ell \times \sigma_L + (A - \ell) \times \sigma_H = C$$
- On consomme donc :
$$\ell \times p_L + (A - \ell) \times p_H$$
- "*DPM*" (Dynamic Power Management) pas intéressant si $\sigma_{ideal} \geq \sigma_1$

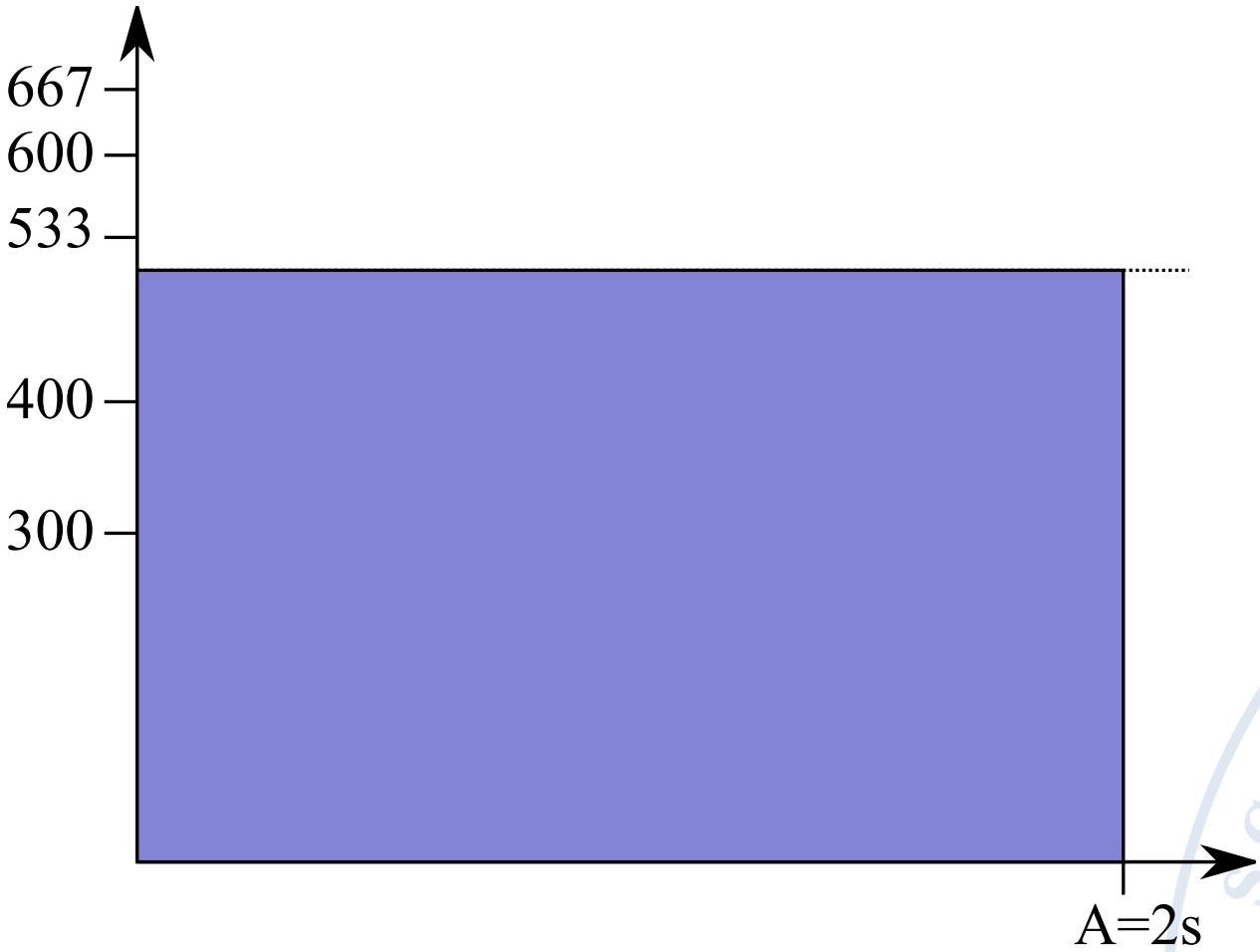
Modes convexes : exemple

Exemple : on veut exécuter 10^9 cycles en $A = 2$ secondes



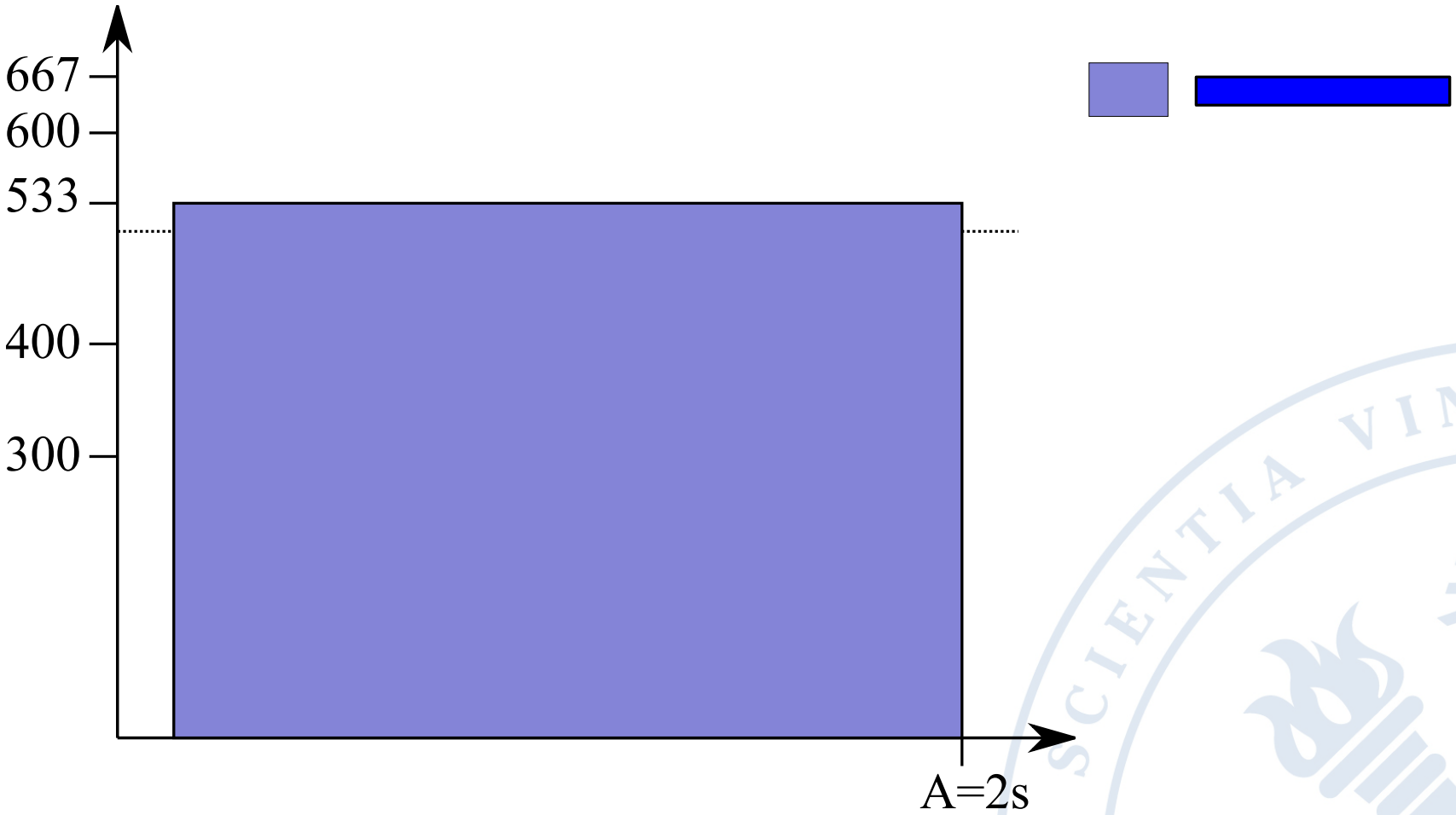
Modes convexes : exemple

Exemple : on veut exécuter 10^9 cycles en $A = 2$ secondes



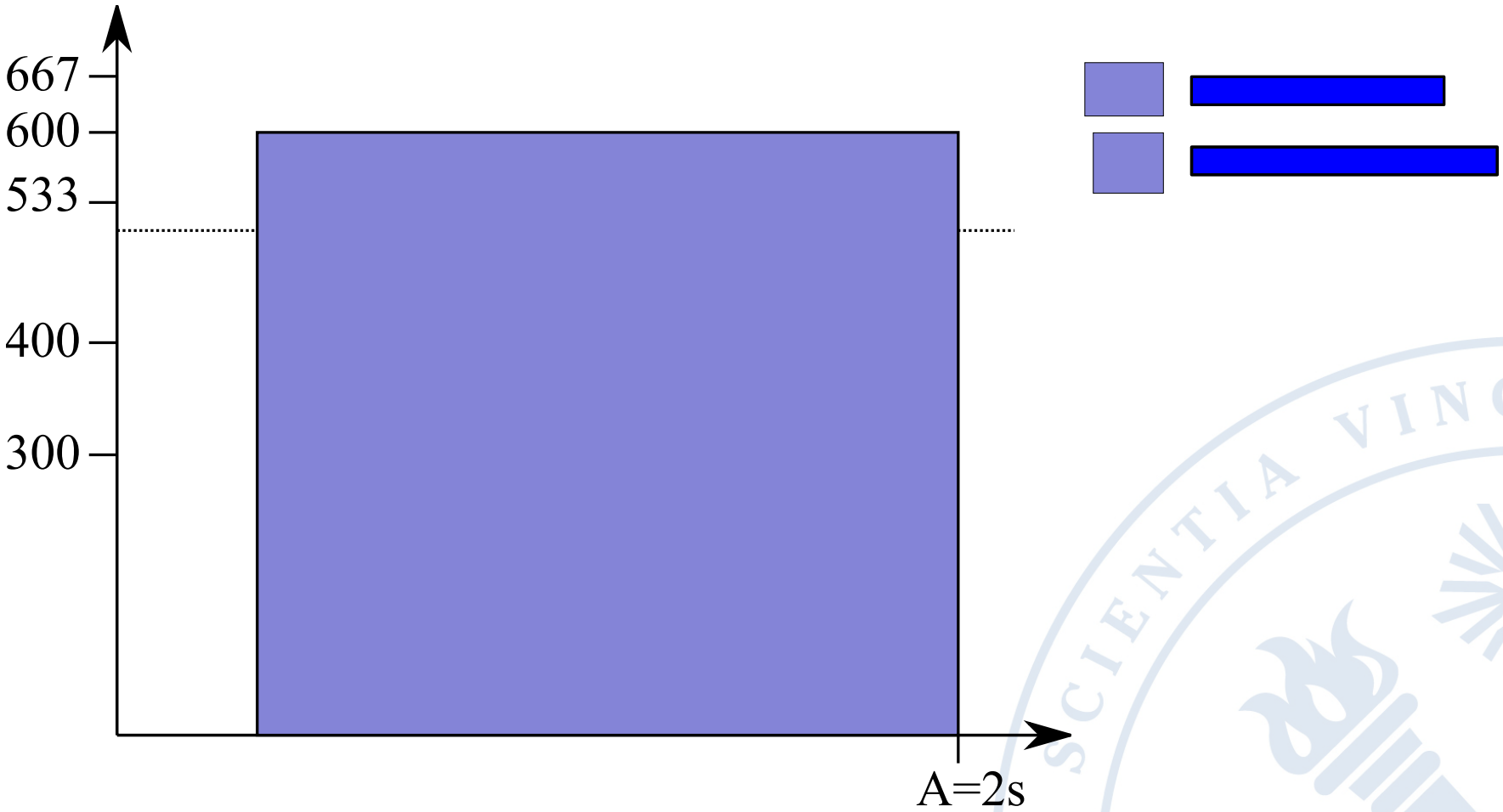
Modes convexes : exemple

Exemple : on veut exécuter 10^9 cycles en $A = 2$ secondes



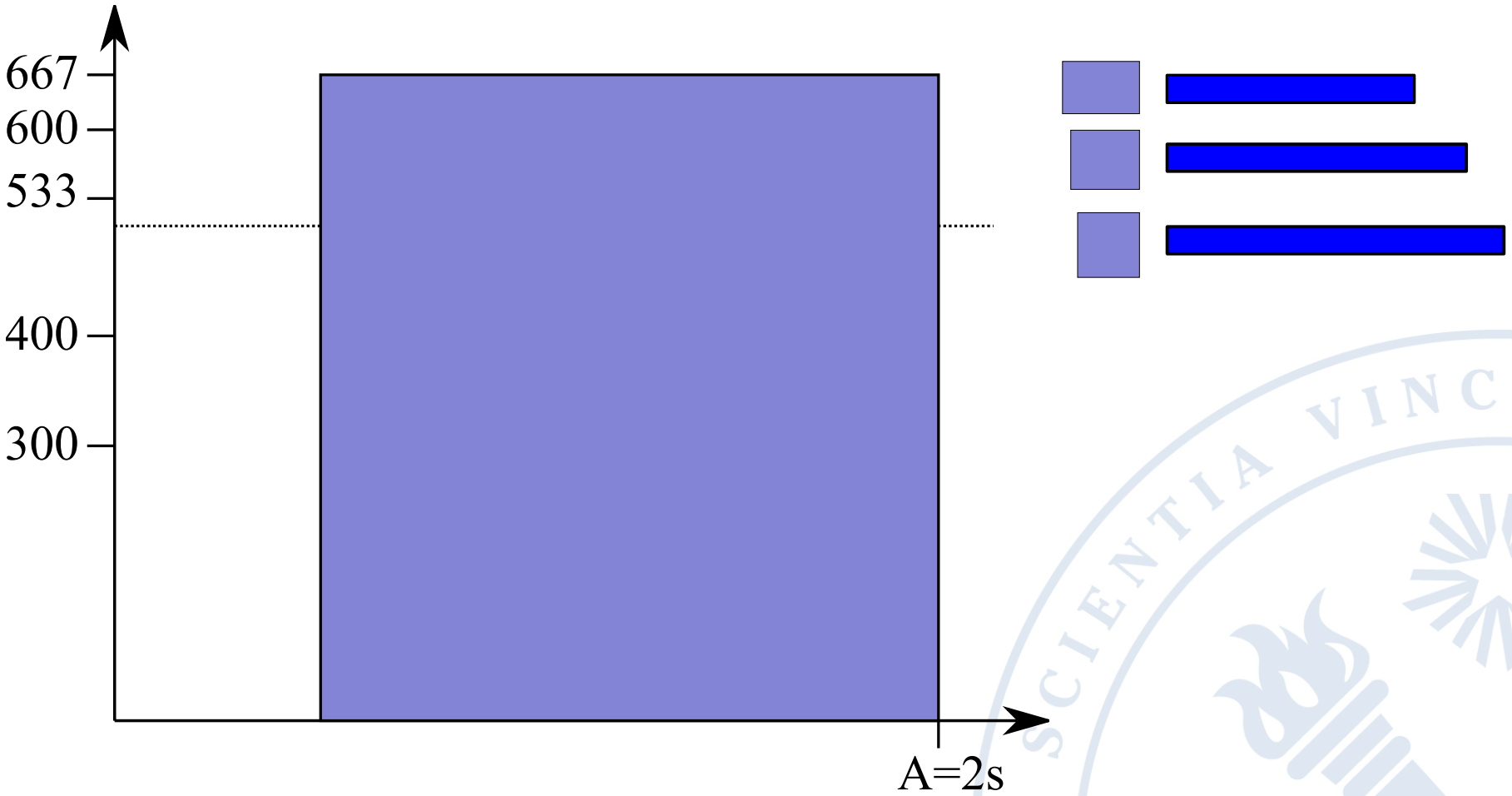
Modes convexes : exemple

Exemple : on veut exécuter 10^9 cycles en $A = 2$ secondes



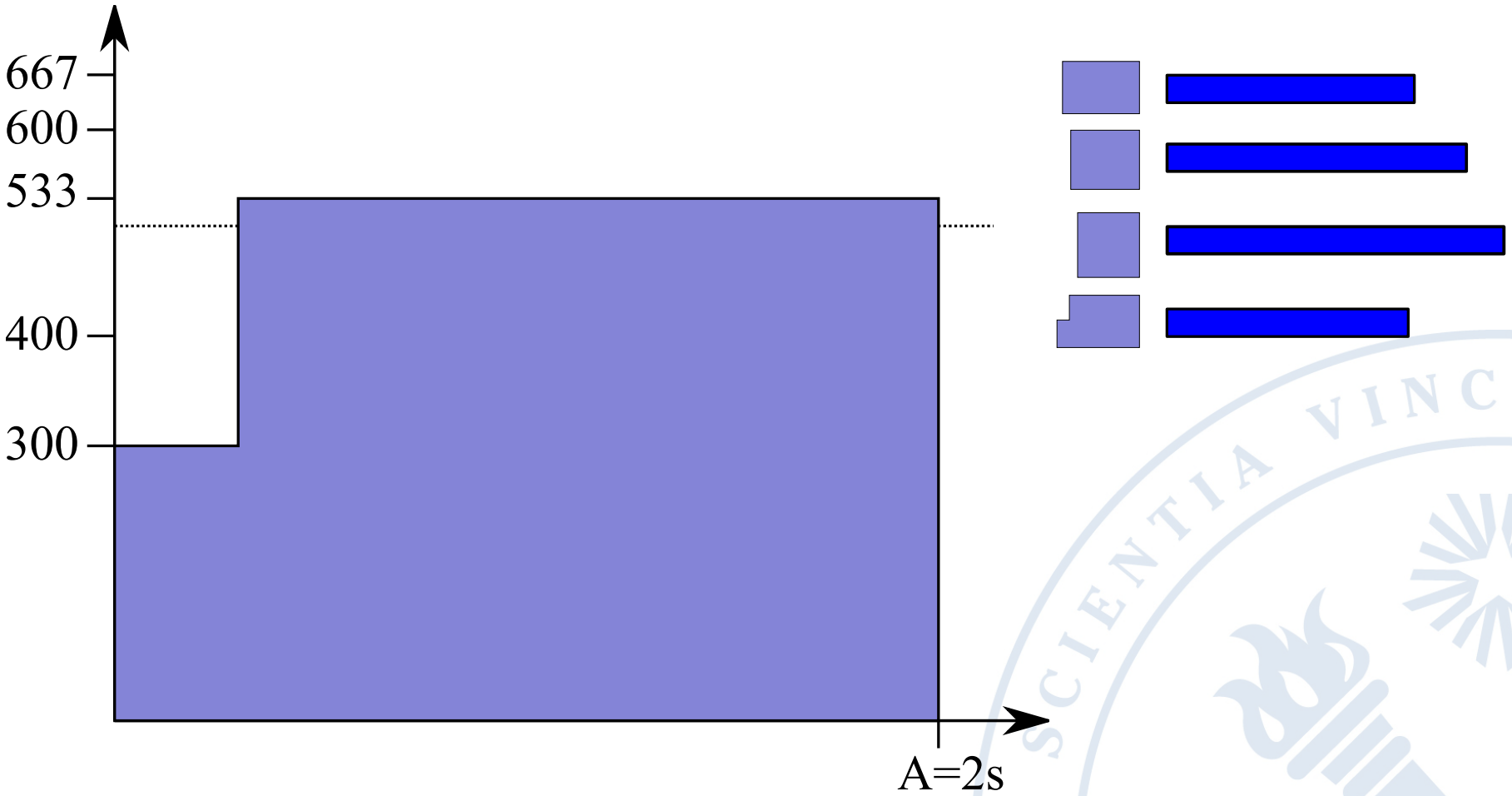
Modes convexes : exemple

Exemple : on veut exécuter 10^9 cycles en $A = 2$ secondes



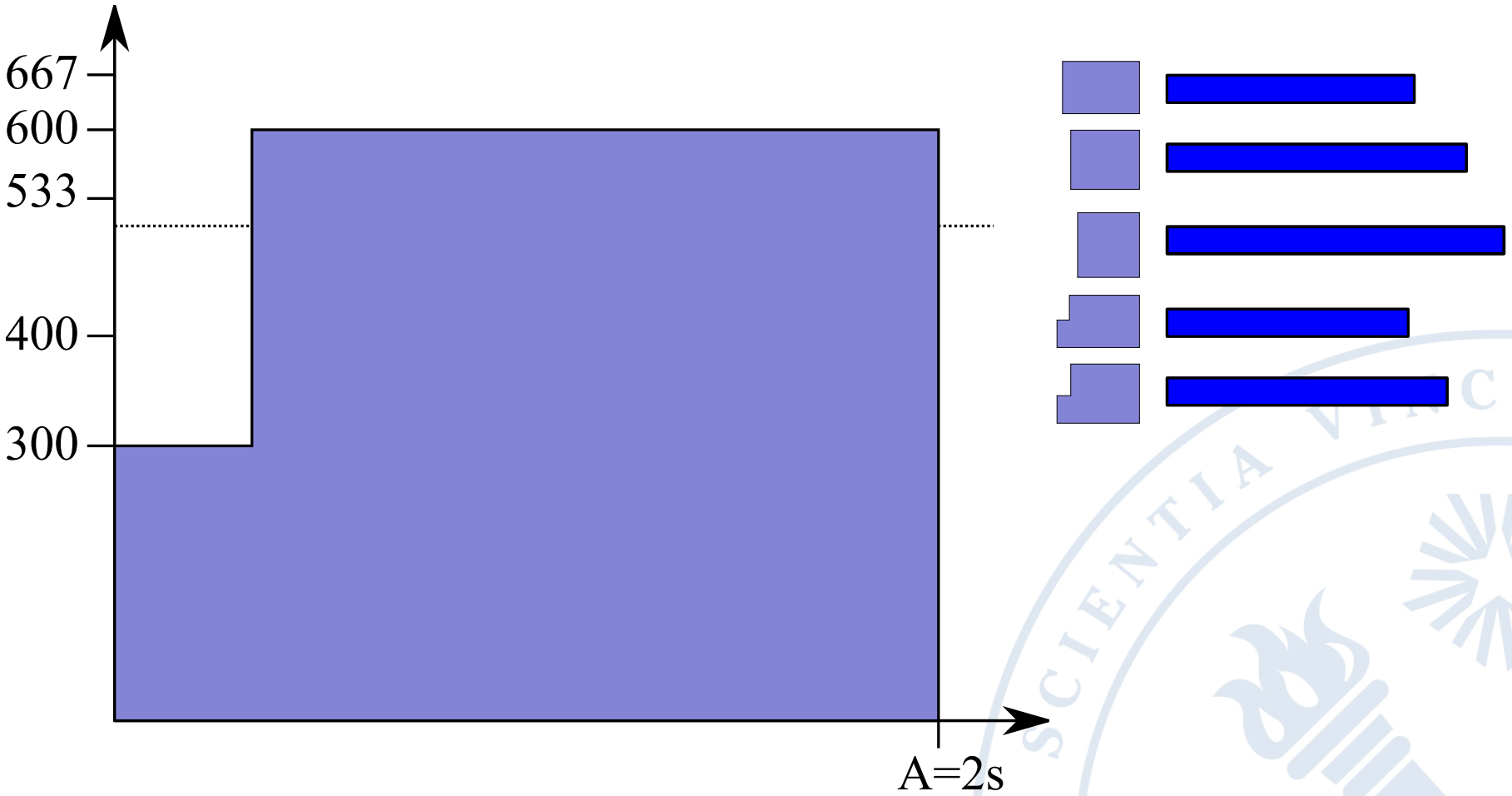
Modes convexes : exemple

Exemple : on veut exécuter 10^9 cycles en $A = 2$ secondes



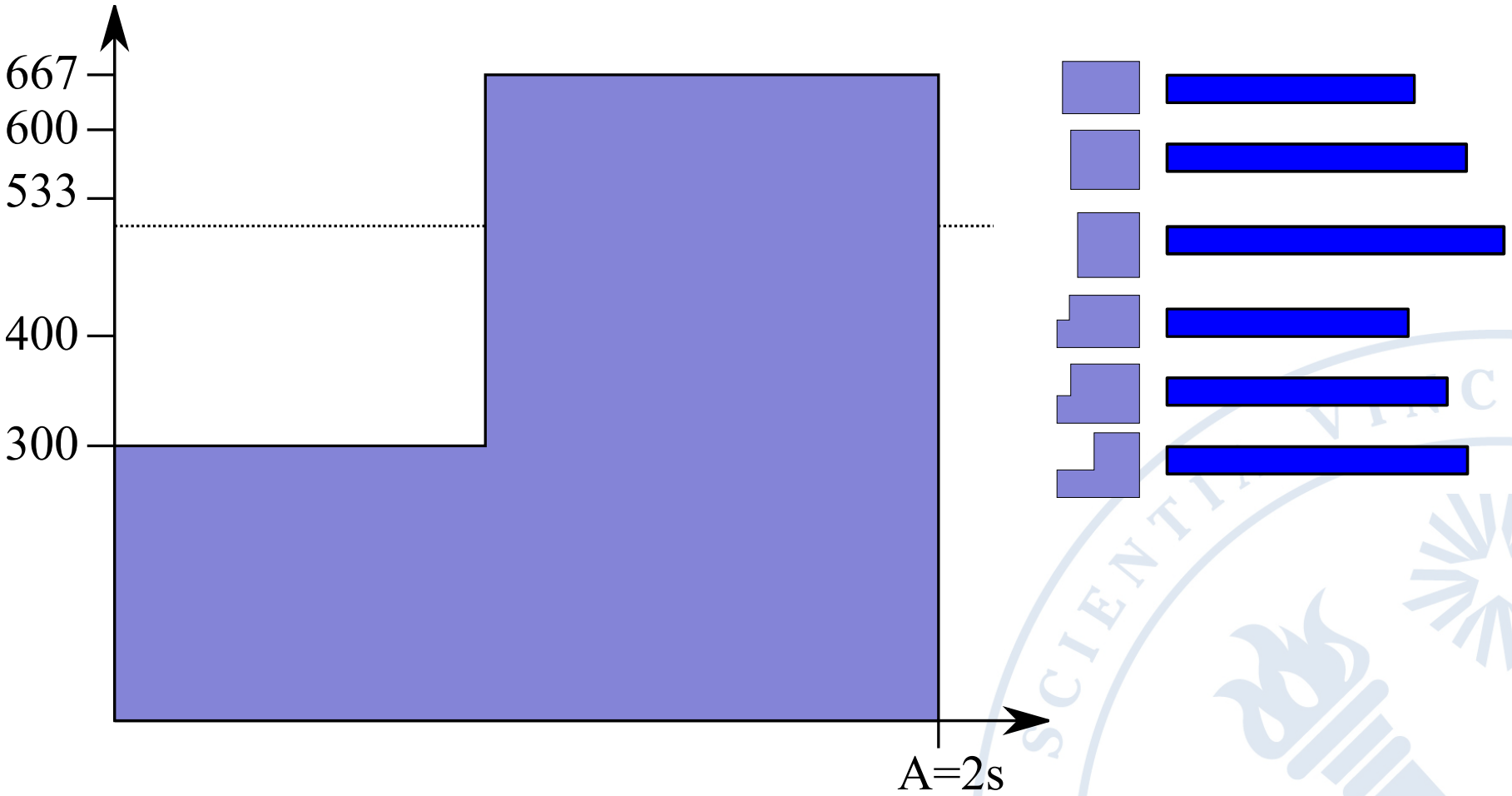
Modes convexes : exemple

Exemple : on veut exécuter 10^9 cycles en $A = 2$ secondes



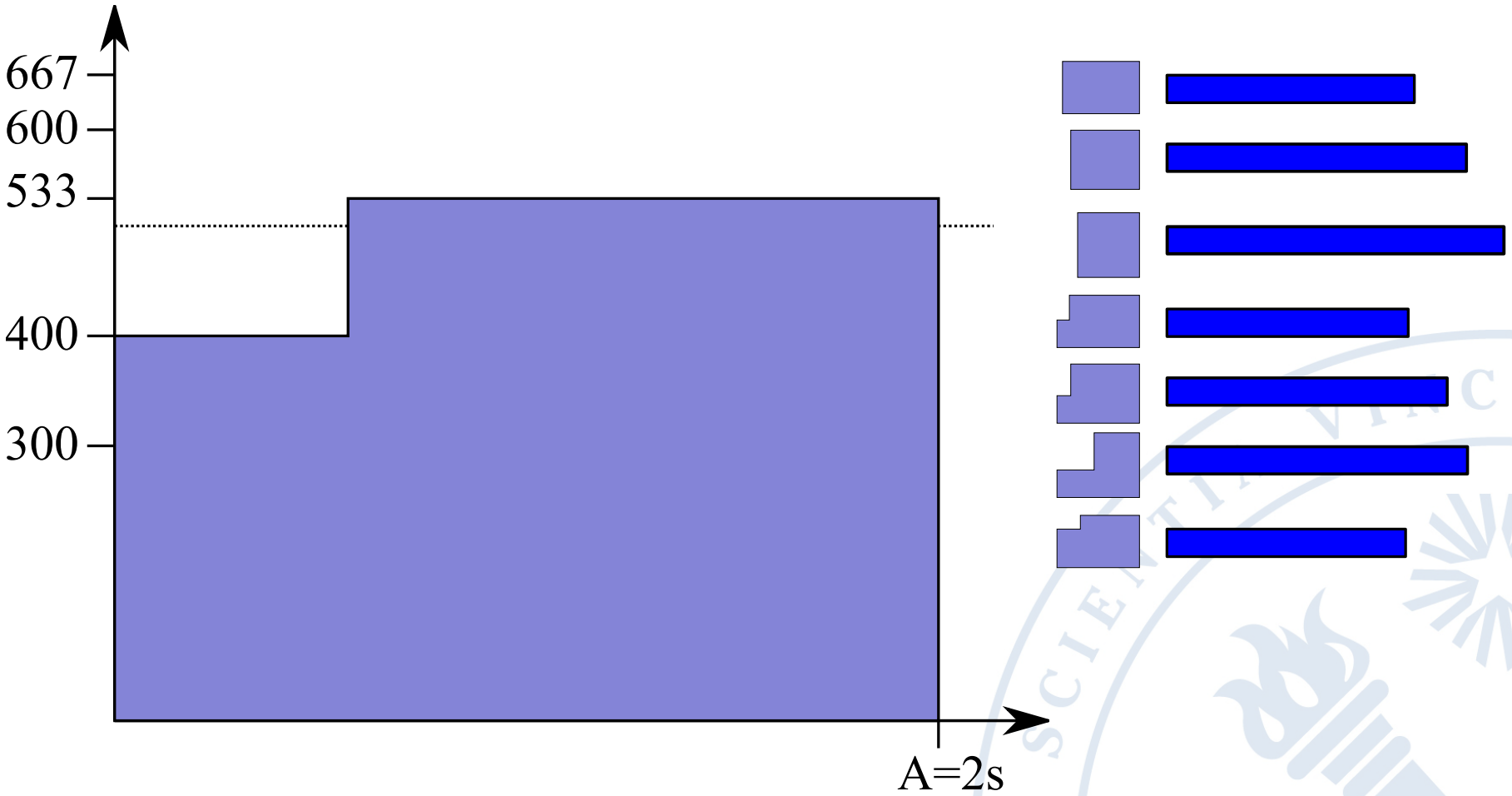
Modes convexes : exemple

Exemple : on veut exécuter 10^9 cycles en $A = 2$ secondes



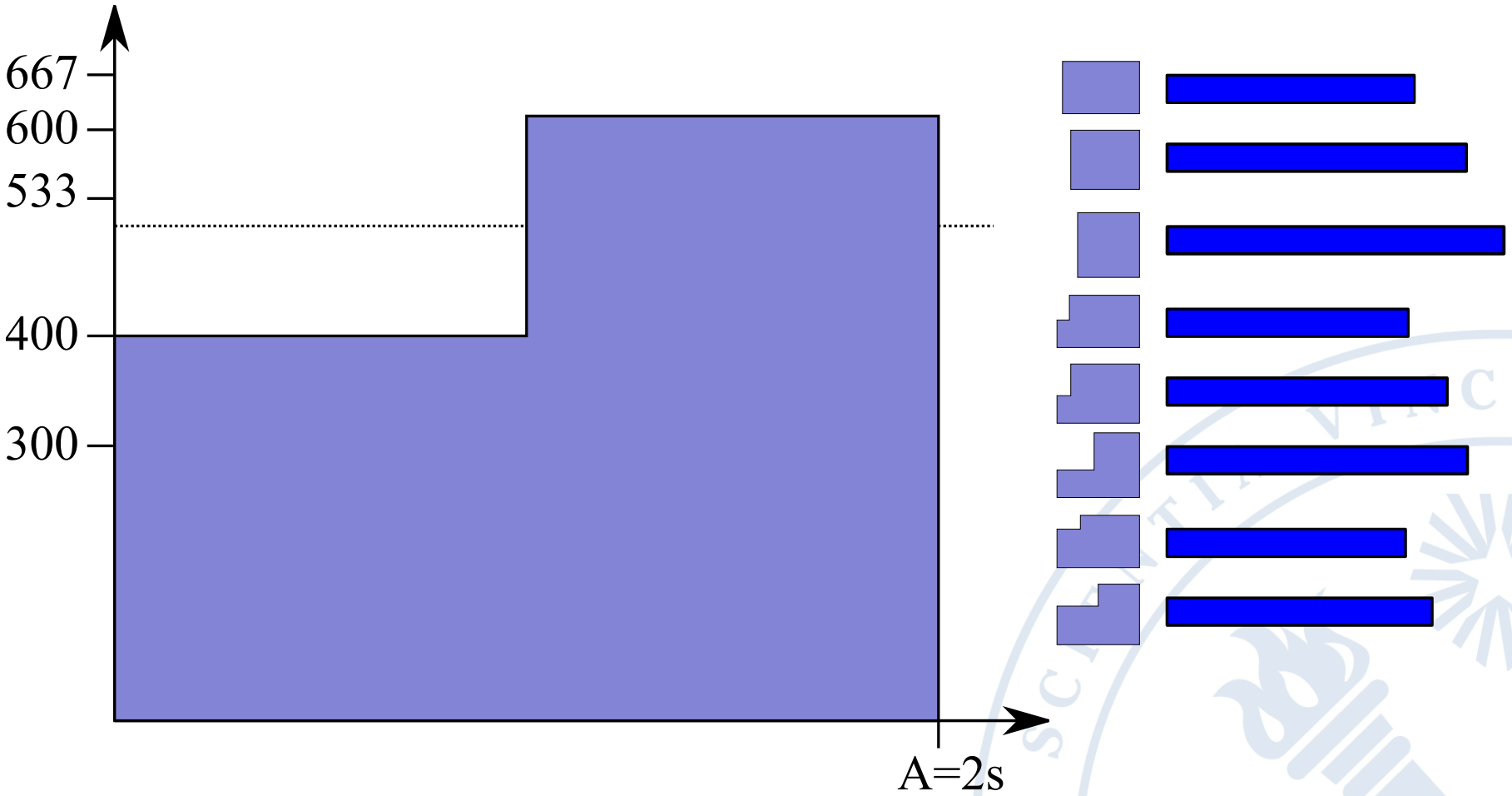
Modes convexes : exemple

Exemple : on veut exécuter 10^9 cycles en $A = 2$ secondes



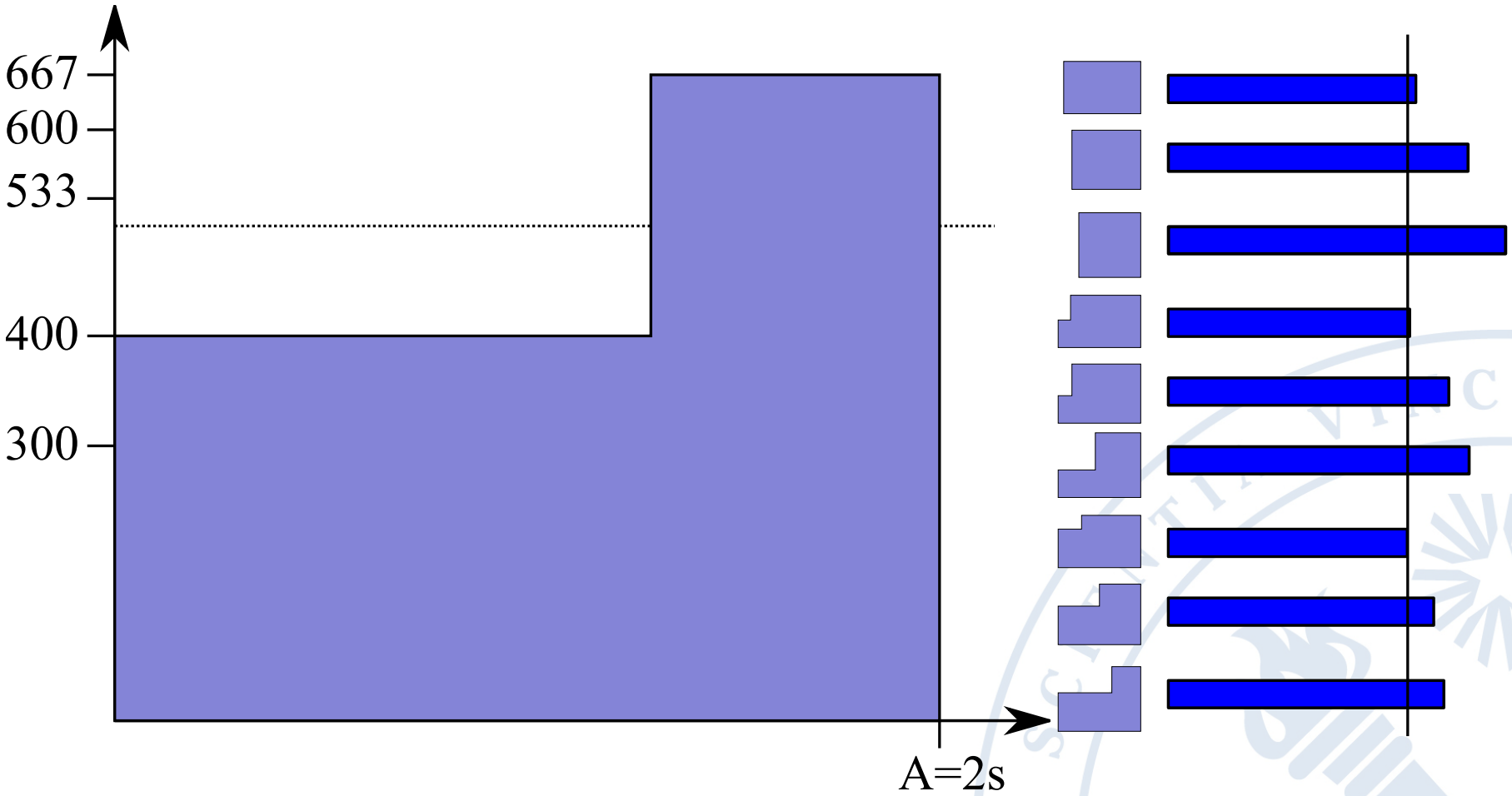
Modes convexes : exemple

Exemple : on veut exécuter 10^9 cycles en $A = 2$ secondes



Modes convexes : exemple

Exemple : on veut exécuter 10^9 cycles en $A = 2$ secondes



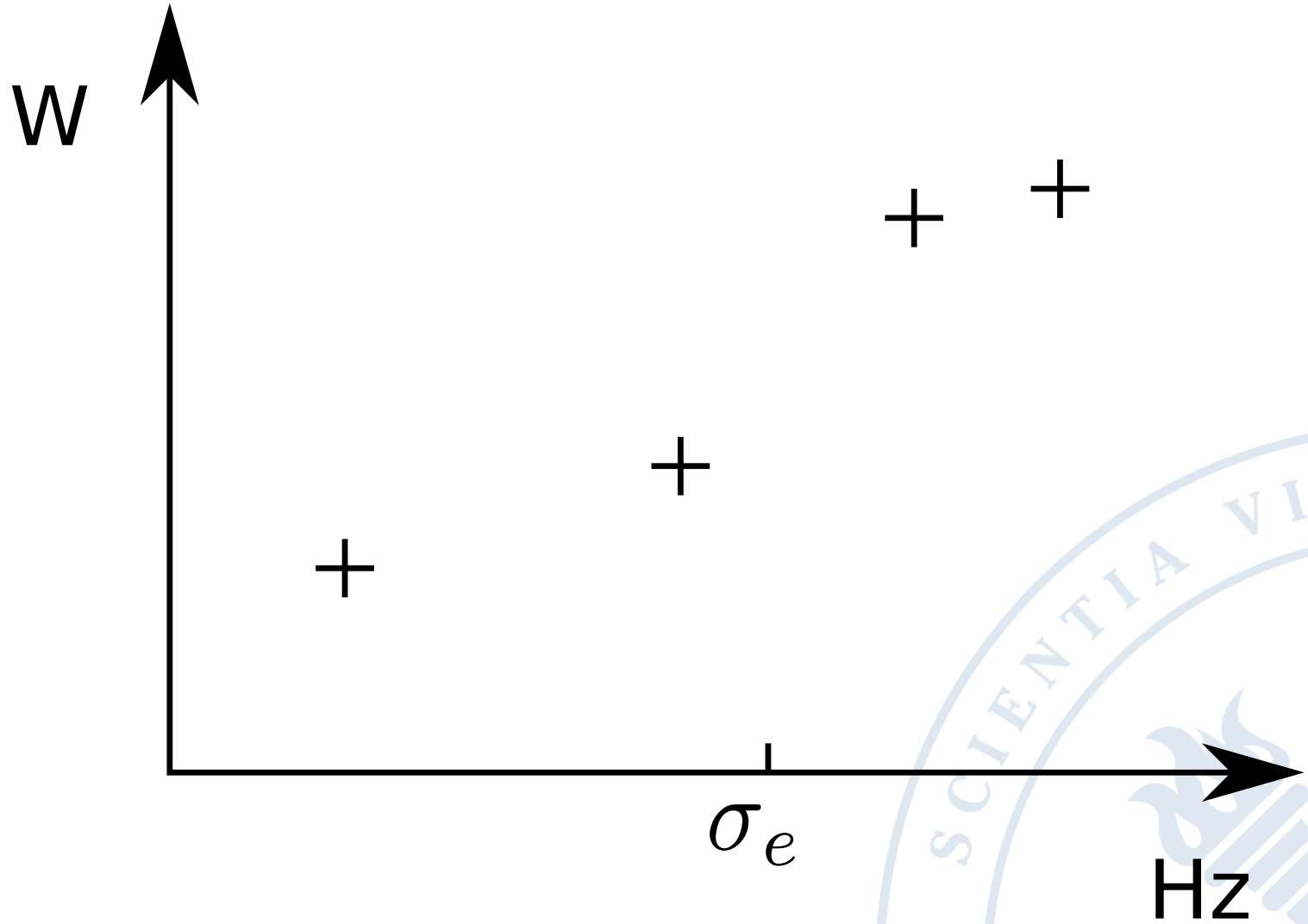
Cas non convexe

- Si les modes ne sont pas convexes, il faut essayer toutes les combinaisons de deux modes [BBL09]
- On peut le faire “intelligemment” : il faut σ_H et σ_L, avec $\sigma_H \geq \sigma_{ideal}$ et $\sigma_L \leq \sigma_{ideal}$, et

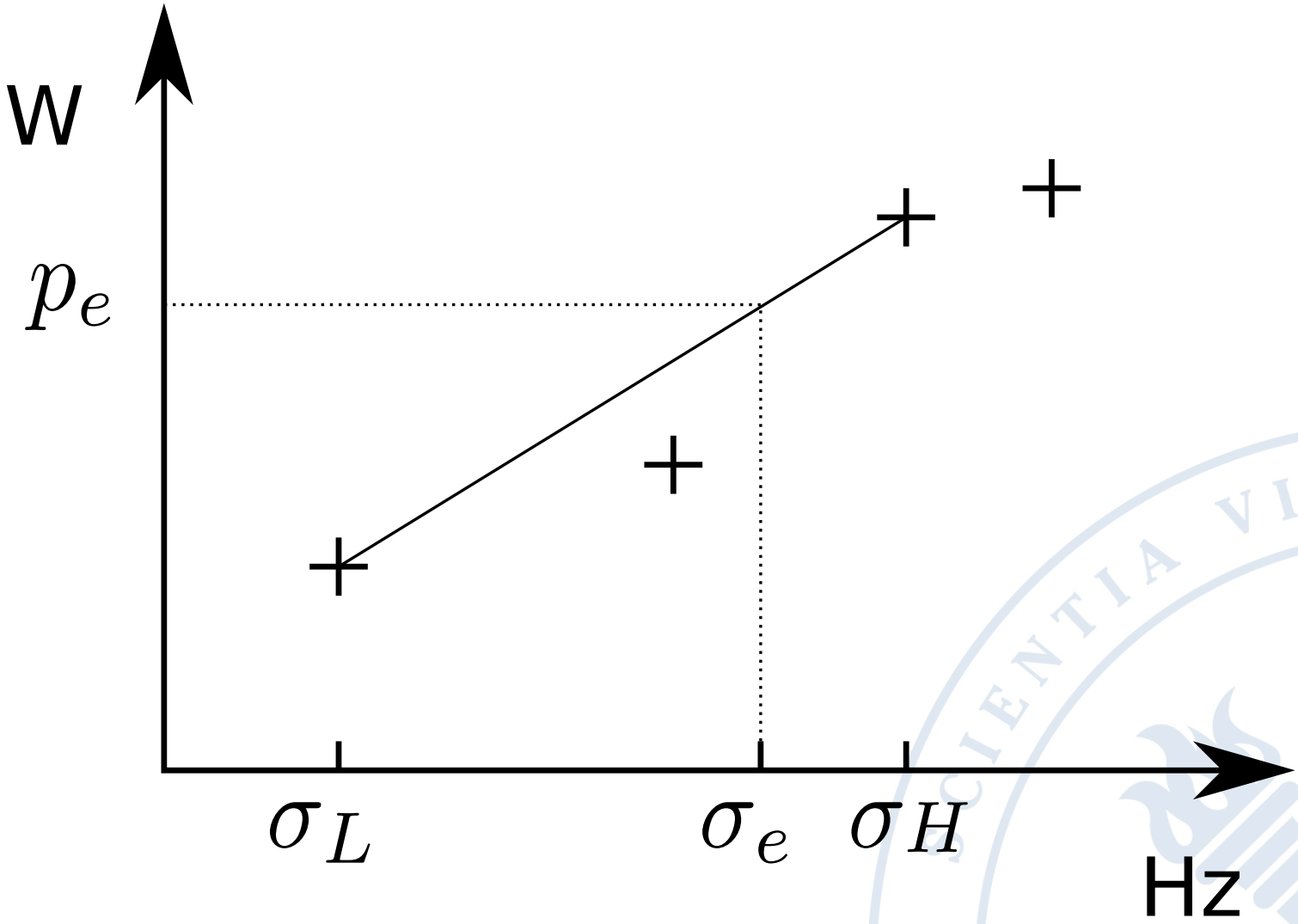
$$\ell \times \sigma_L + (A - \ell) \times \sigma_H = C$$

- Très facile d’intégrer des contraintes plus réalistes (*power overhead, switching time*, . . .)
- Calculs simples, peuvent être faits en ligne (A peut donc varier pour chaque instance)
- “*DPM*” (Dynamic Power Management) peut être intéressant (intégrer le(s) mode(s) *idle*/veille/off)

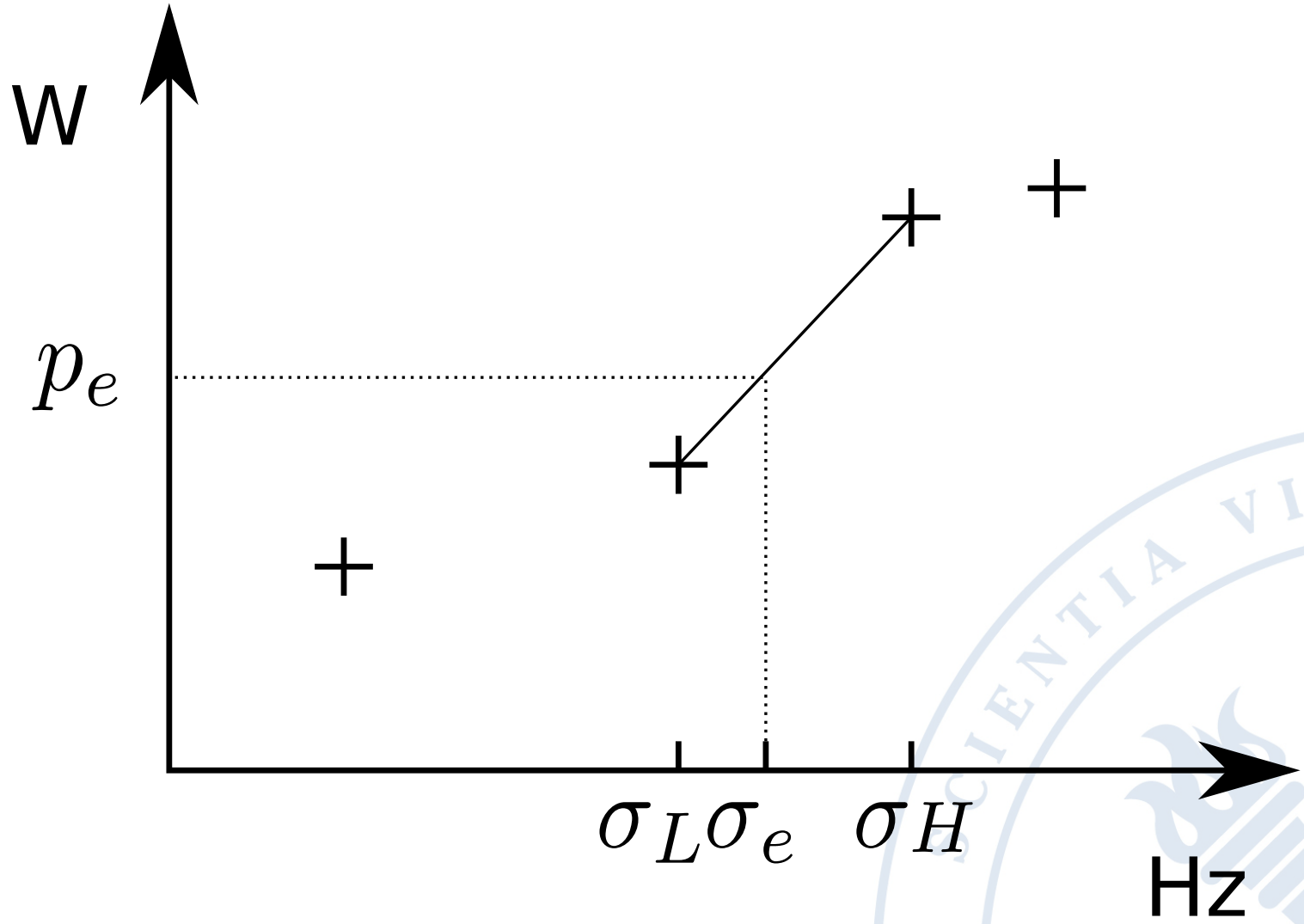
Modes non-convexes, choix du couple σ_L, σ_H



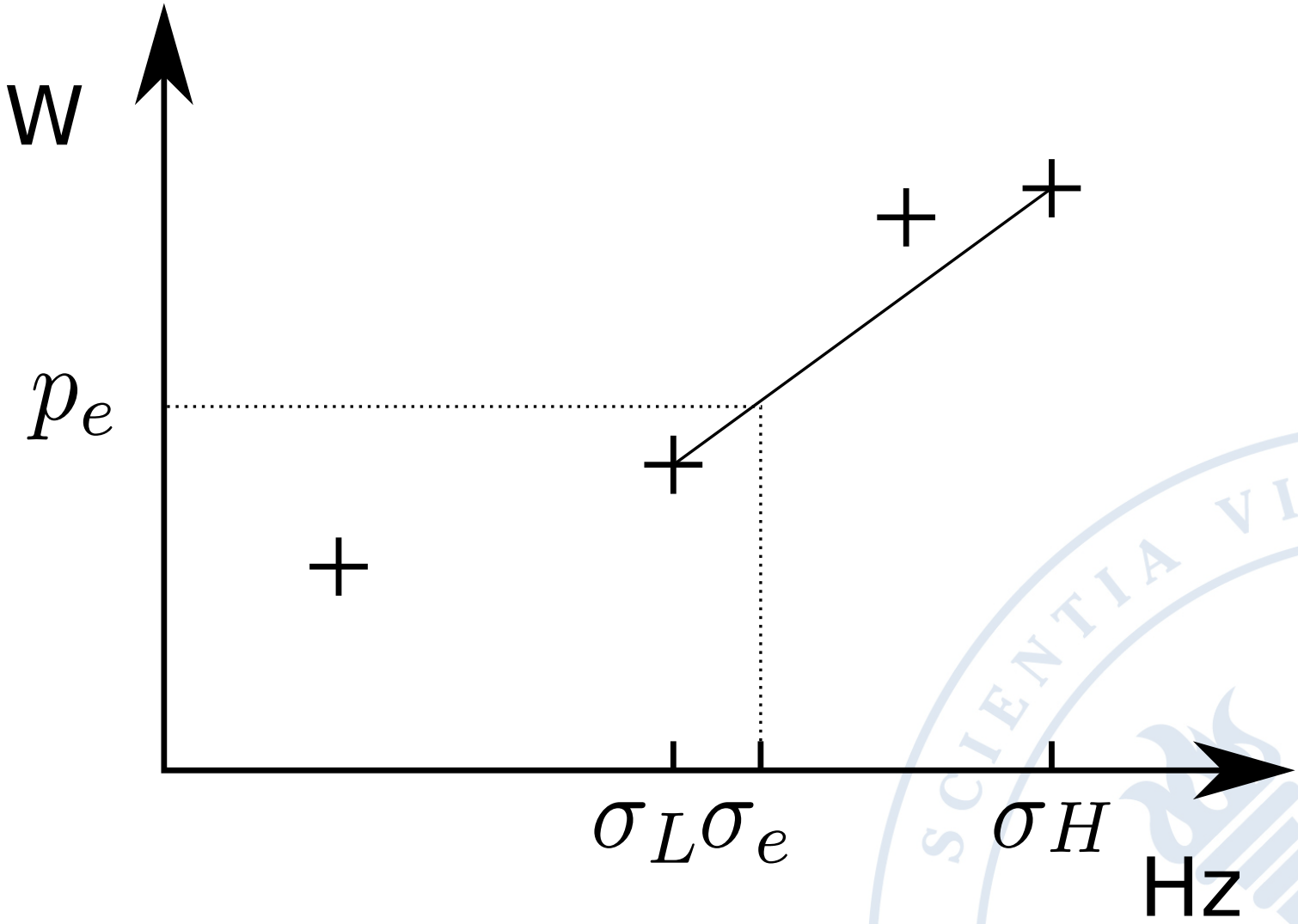
Modes non-convexes, choix du couple σ_L, σ_H



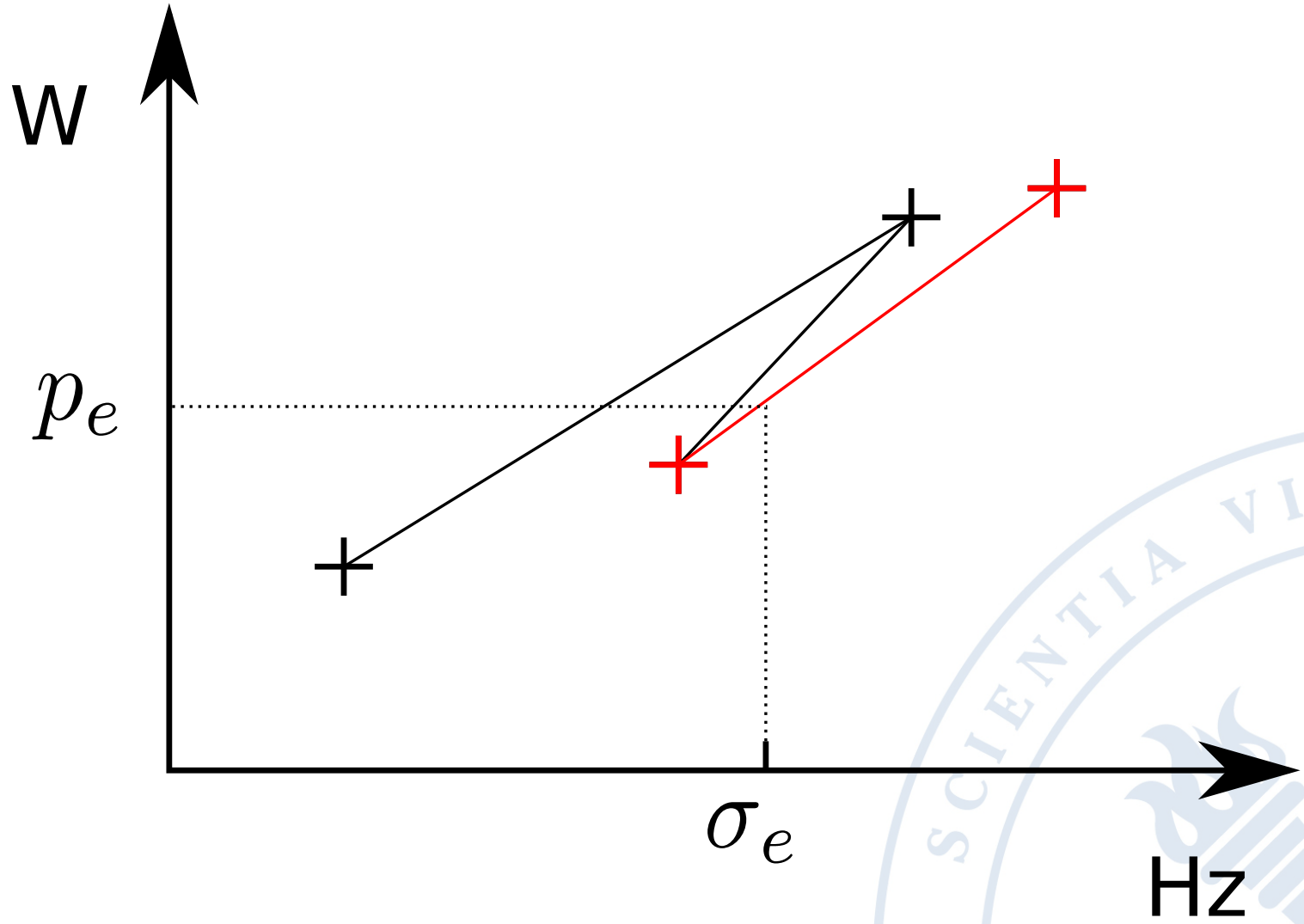
Modes non-convexes, choix du couple σ_L, σ_H



Modes non-convexes, choix du couple σ_L, σ_H



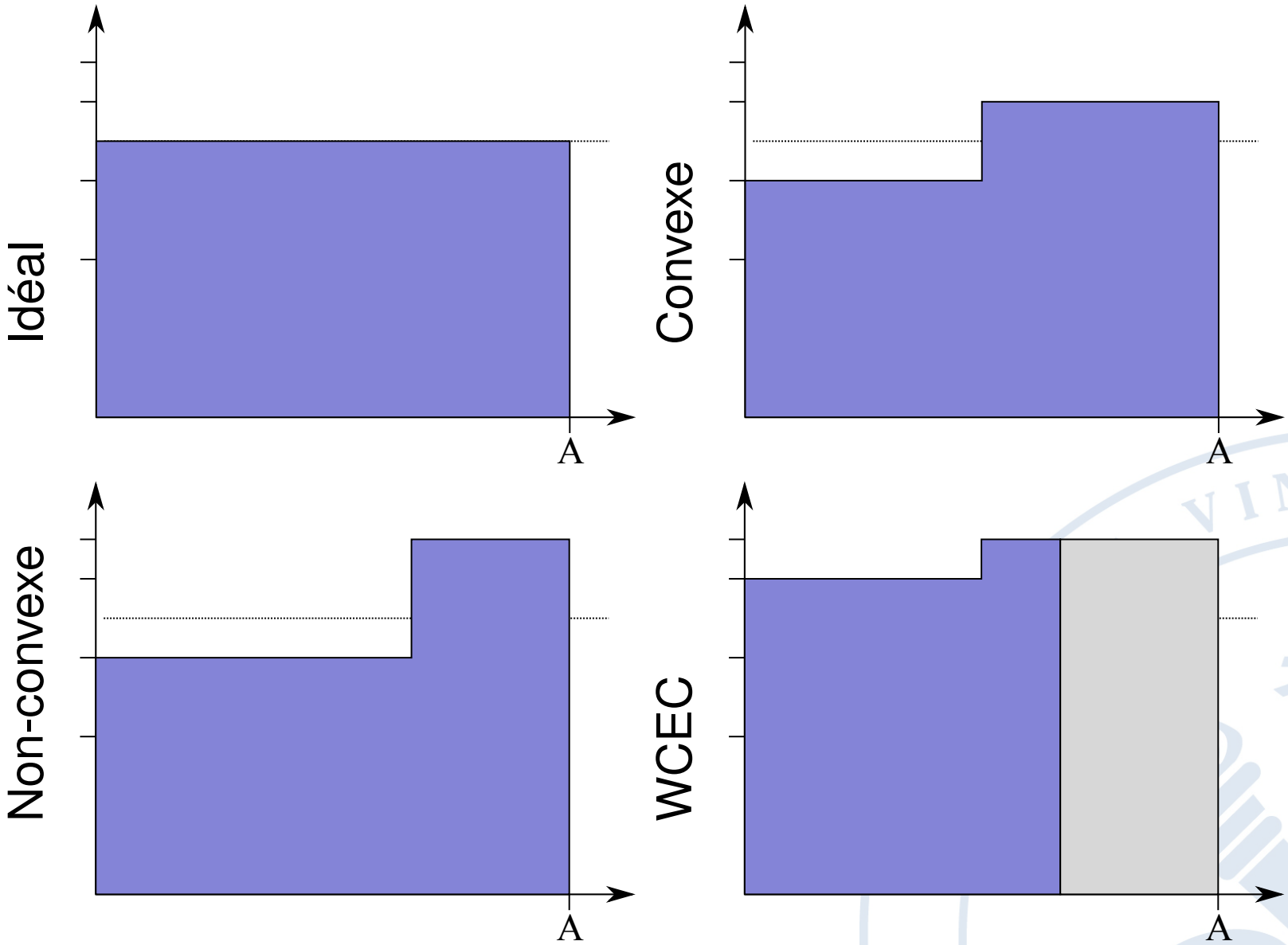
Modes non-convexes, choix du couple σ_L, σ_H



WCEC

- Les techniques présentées peuvent être utilisées sur base du WCEC au lieu de la durée exacte
- On aura alors souvent du *slack*, qui pourra être distribué
- Il existe de très nombreux travaux sur le *slack reclaiming*, mais ce n'est pas l'objet de cet exposé !
- Si on ne connait pas la durée exacte, mais uniquement le WCEC et quelques informations sur la distribution, on peut faire beaucoup mieux
- C'est l'objet de la section suivante !

Résumé



Section 4

Durées probabilistes

Table des matières I

Contexte

- Si on connait les durées d'exécution, on peut optimiser l'énergie *exacte* dépensée par un job
- Si on ne les connait pas, on s'intéressera à l'énergie *moyenne*
- Les premiers cycles s'exécutent très probablement
- Les derniers cycles (près du WCEC) ont peu de chance de s'exécuter → s'ils consomment beaucoup d'énergie, ça a moins d'impact que les premiers
- Donc : on commence lentement pour accélérer plus tard *si nécessaire*
- Le plus constant possible n'est plus optimal !

Énergie moyenne

- Soit $e(x) \stackrel{\text{def}}{=} \frac{p(x)}{\sigma(x)}$ l'énergie du cycle x
- Énergie si x cycles : $\sum_{y \leq x} e(y)$
- Énergie moyenne :

$$\begin{aligned} \overline{E} &= \sum_{x \leq \mathcal{W}} c(x) \sum_{y \leq x} e(y) \\ &= \sum_{x \leq \mathcal{W}} e(x) \sum_{y \geq x} c(y) \\ &= \sum_{x \leq \mathcal{W}} e(x)(1 - C(x)) \end{aligned}$$

Objectif

- On veut choisir les $\mathcal{W}$ variables $e(x)$ (M valeurs possibles),
- minimisant
$$\sum_{x \leq \mathcal{W}} e(x)(1 - C(x))$$
- sous la contrainte
$$\sum_{x \leq \mathcal{W}} \frac{1}{\sigma(x)} \leq A$$
- On peut facilement ajouter des contraintes réalistes (*power overhead, switching time*, ...)
- On peut simplifier le problème :
  - Pour certaines formes particulières de $e(x)$ et/ou $C(x)$
  - En augmentant la taille des blocs de cycles
  - En limitant le nombre de changements autorisés

Modes polynomiaux

- Dans le cas où $e(x) = \mathcal{K} \cdot \sigma(x)^{\beta}$ pour des constantes $\mathcal{K}$ et β, on a une forme clause [Gru02]
- Cas $\beta = 2$:

$$\sigma(x) = \frac{1}{A} \times \frac{\sum_y \sqrt[3]{1 - C(y)}}{\sqrt[3]{1 - C(x)}}$$

- Si la fréquence n'est pas disponible, elle peut être *convertie*
- La partie droite peut être calculée *hors ligne*

Changements limités

- Si on impose un seul changement par job (k cycles à σ_L puis $\mathcal{W} - k$ cycles à σ_H) :

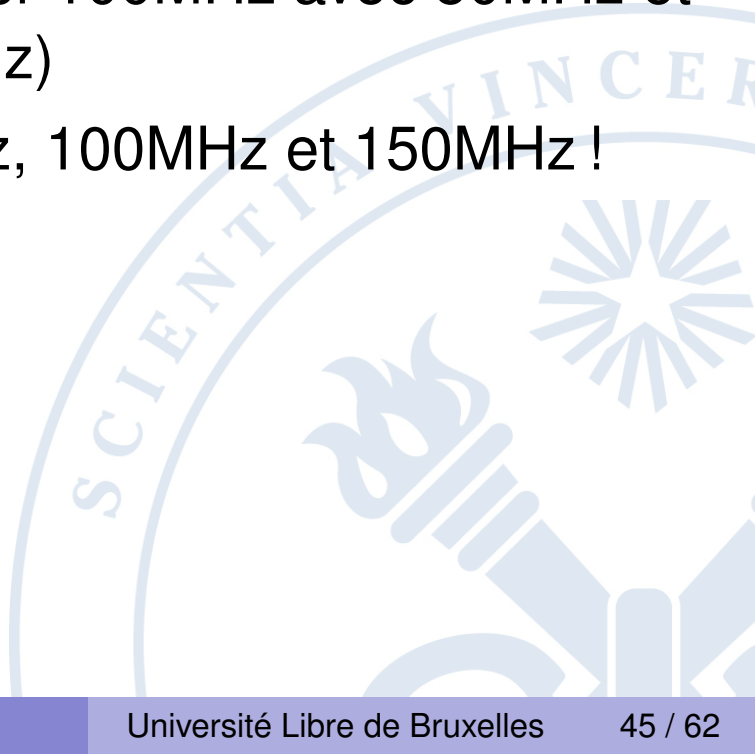
$$\overline{E} = e_L \sum_{x \leq k} (1 - C(x)) + e_H \sum_{x > k} (1 - C(x))$$

  sous la contrainte $\frac{k}{\sigma_L} + \frac{\mathcal{W} - k}{\sigma_H} \leq A$

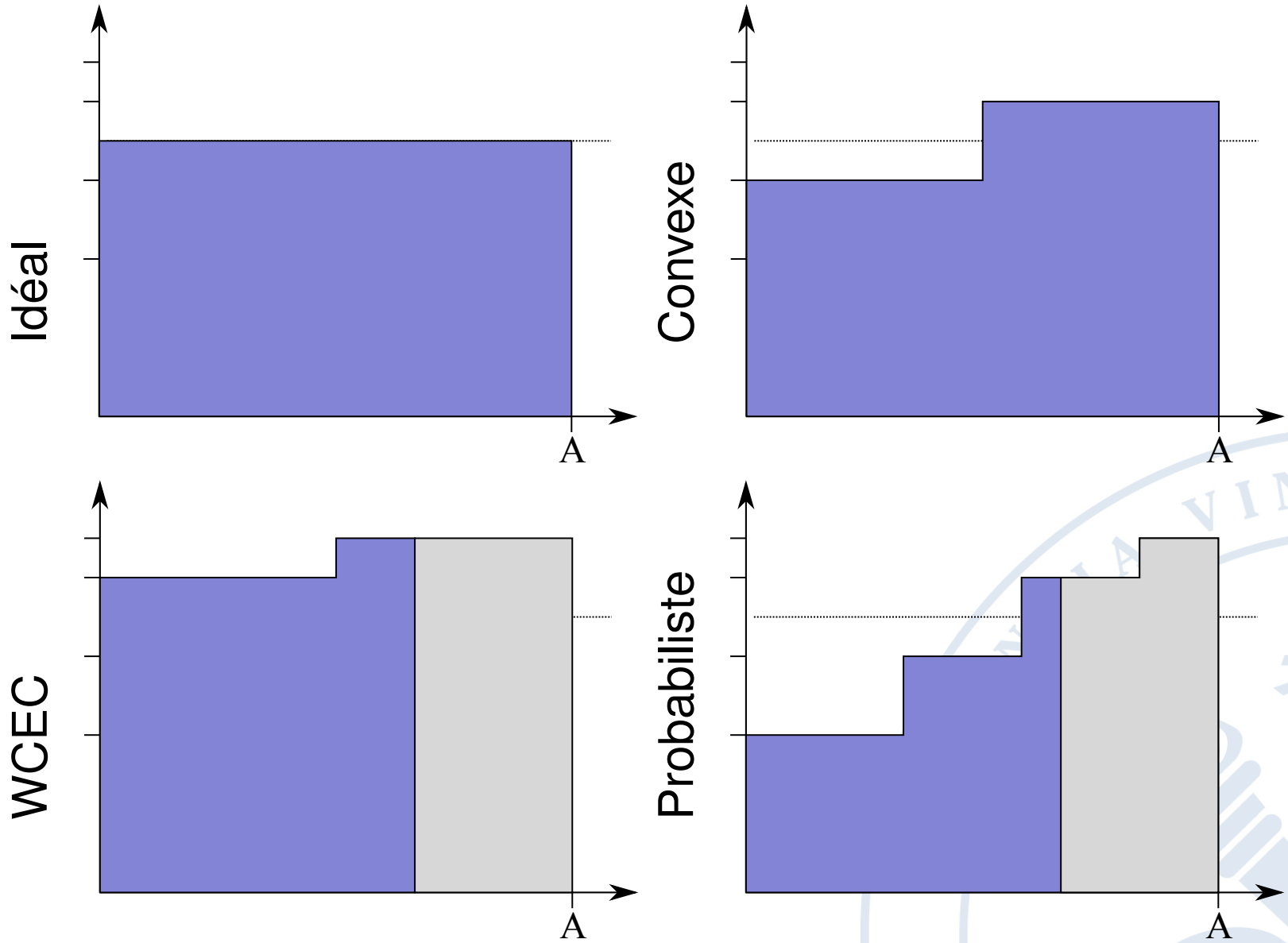
- On peut essayer toutes les combinaisons où $\sigma_L \leq \frac{\mathcal{W}}{A} \leq \sigma_H$
- Au maximum : $\left(\frac{M}{2}\right)^2$ combinaisons

Changements limités

- On peut généraliser à plus de changements, mais ça devient plus complexe
- Un seul découpage possible pour simuler 100MHz avec 50MHz et 150MHz (50% à 50MHz, 50 % à 150MHz)
- Une infinité de découpages avec 50MHz, 100MHz et 150MHz !



Résumé



Section 5

Tâches multi-segment

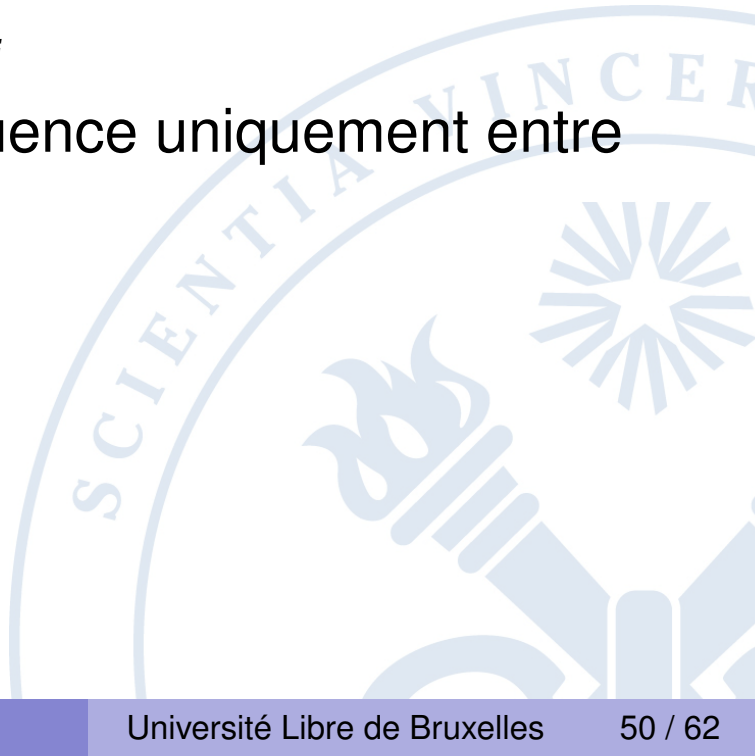
Table des matières I

Contexte

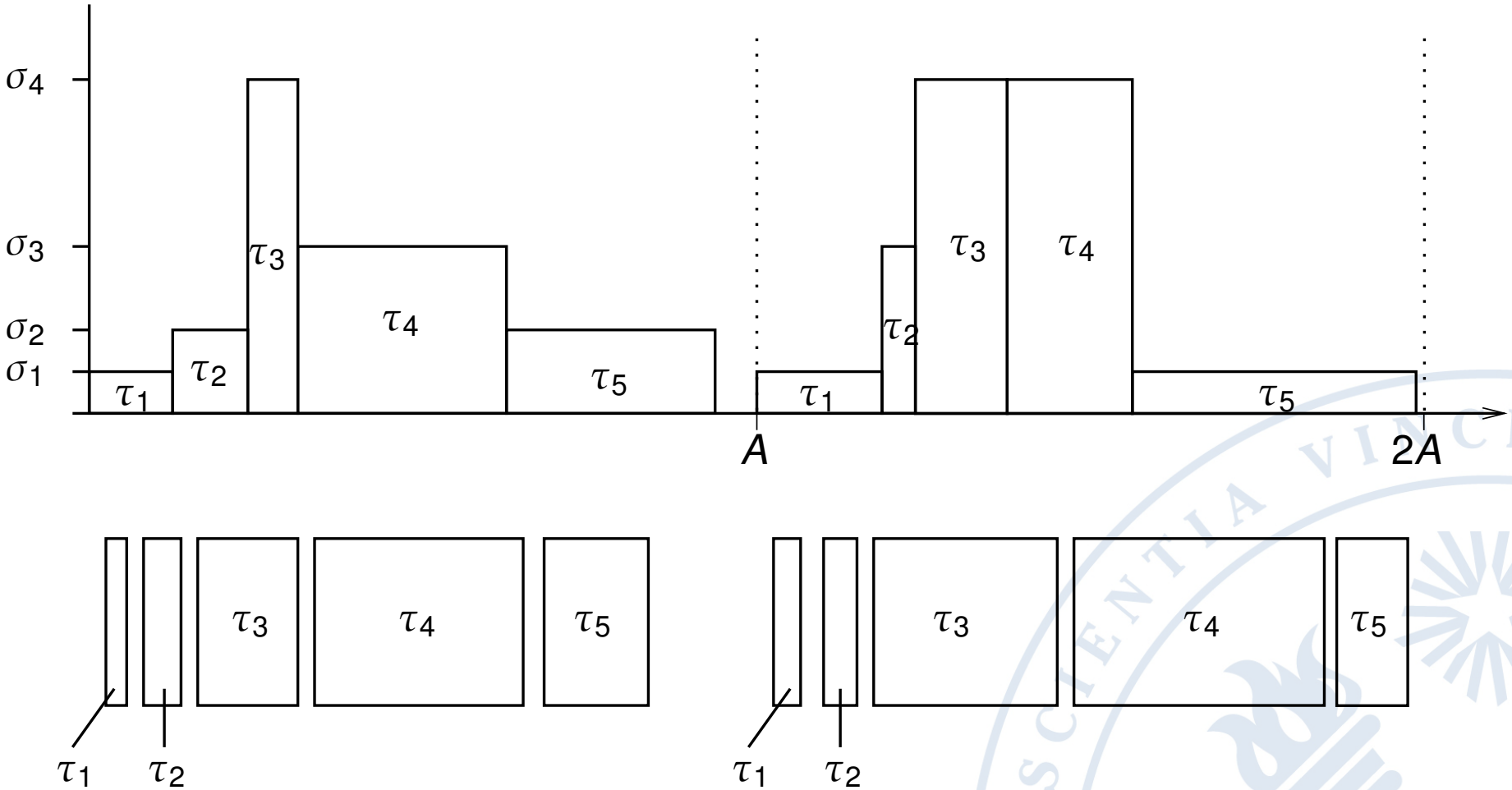
- Parfois, on a une connaissance plus fine que la distribution pour l'ensemble d'un job
- Par exemple :
  - une phase d'initialisation : temps constant
  - les calculs : temps très variable
  - la finalisation : temps peu variable
- On peut alors s'approcher de l'objectif "temps constant"
- Nécessite une assistance du compilateur/programmeur
- Peut aussi s'appliquer aux modèles "Frame based"
- Stratégie différente de la section précédente :
  - Démarrer rapidement (pour être sûr d'atteindre la DL)
  - Ralentir progressivement si possible

Modèle

- On considère N tâches indépendantes mais successives $\tau_1, \tau_2, \ldots, \tau_N$
- Toutes partagent la même échéance A (*allotted time*)
- Pour chaque tâche, on a $c_i()$ et $C_i()$, $\mathcal{W}_i$
- On suppose qu'on peut changer la fréquence uniquement entre deux jobs



Modèle



Zone de danger

- Considérons le démarrage de τ_N
- Démarre après $A - \frac{\mathcal{W}_N}{\sigma_M} \Rightarrow$ risque d'échec en A !
- $A - \frac{\mathcal{W}_N}{\sigma_M}$ est donc l'échéance de τ_{N-1}
- En général, l'échéance de τ_i est :

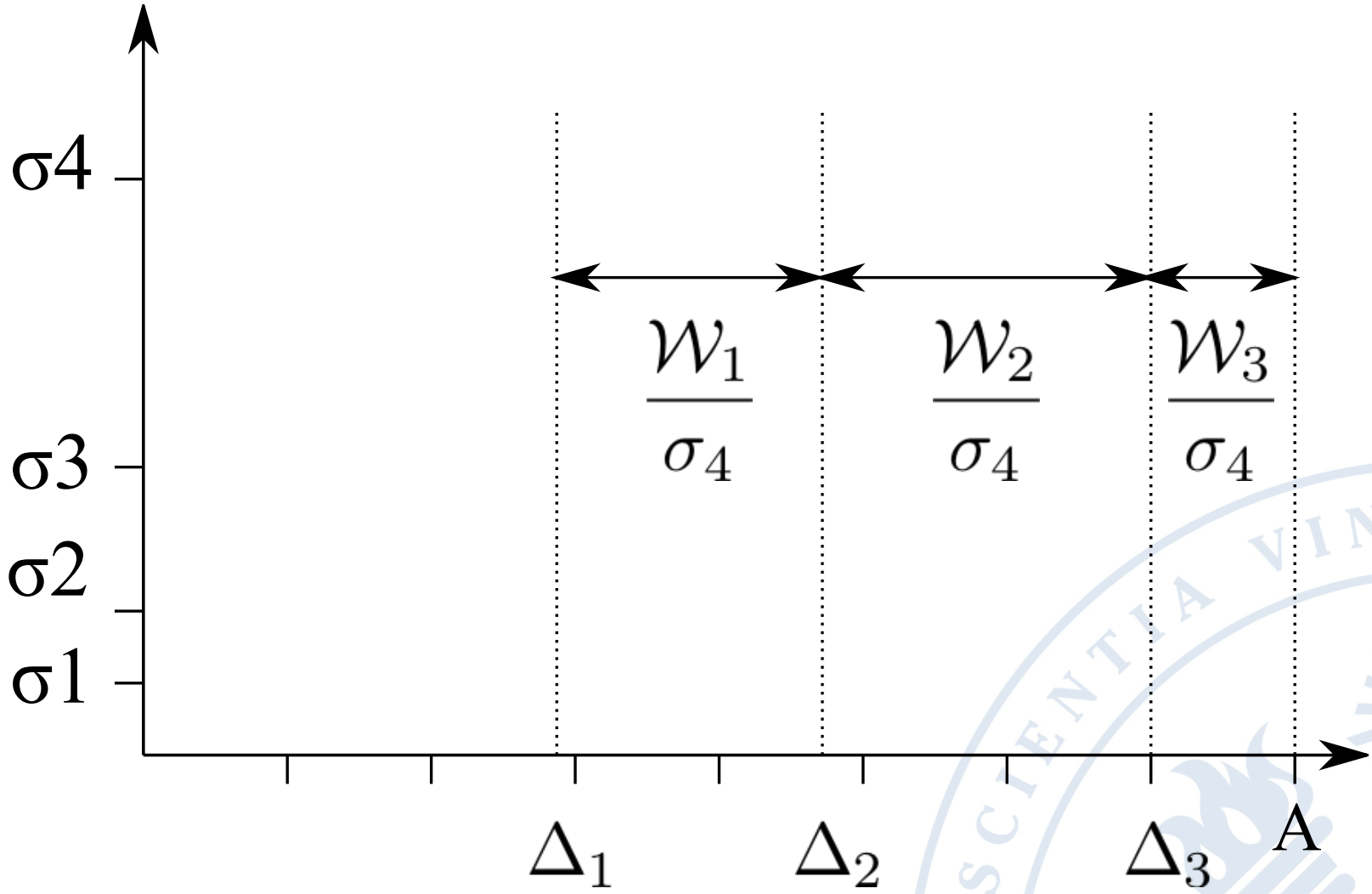
$$A - \frac{1}{\sigma_M} \sum_{k=i+1}^{N} \mathcal{W}_i$$

- Tout job démarrant après

$$\Delta_i \stackrel{\text{def}}{=} A - \frac{1}{\sigma_M} \sum_{k=i}^{N} \mathcal{W}_i$$

  est donc "en danger" !
- Un job de τ_i démarrant en Δ_i DOIT utiliser σ_M !

Échéances intermédiaires



Fréquence minimale

- Si τ_N démarre en t ($\leq \Delta_i$), sa fréquence *minimale* est de :

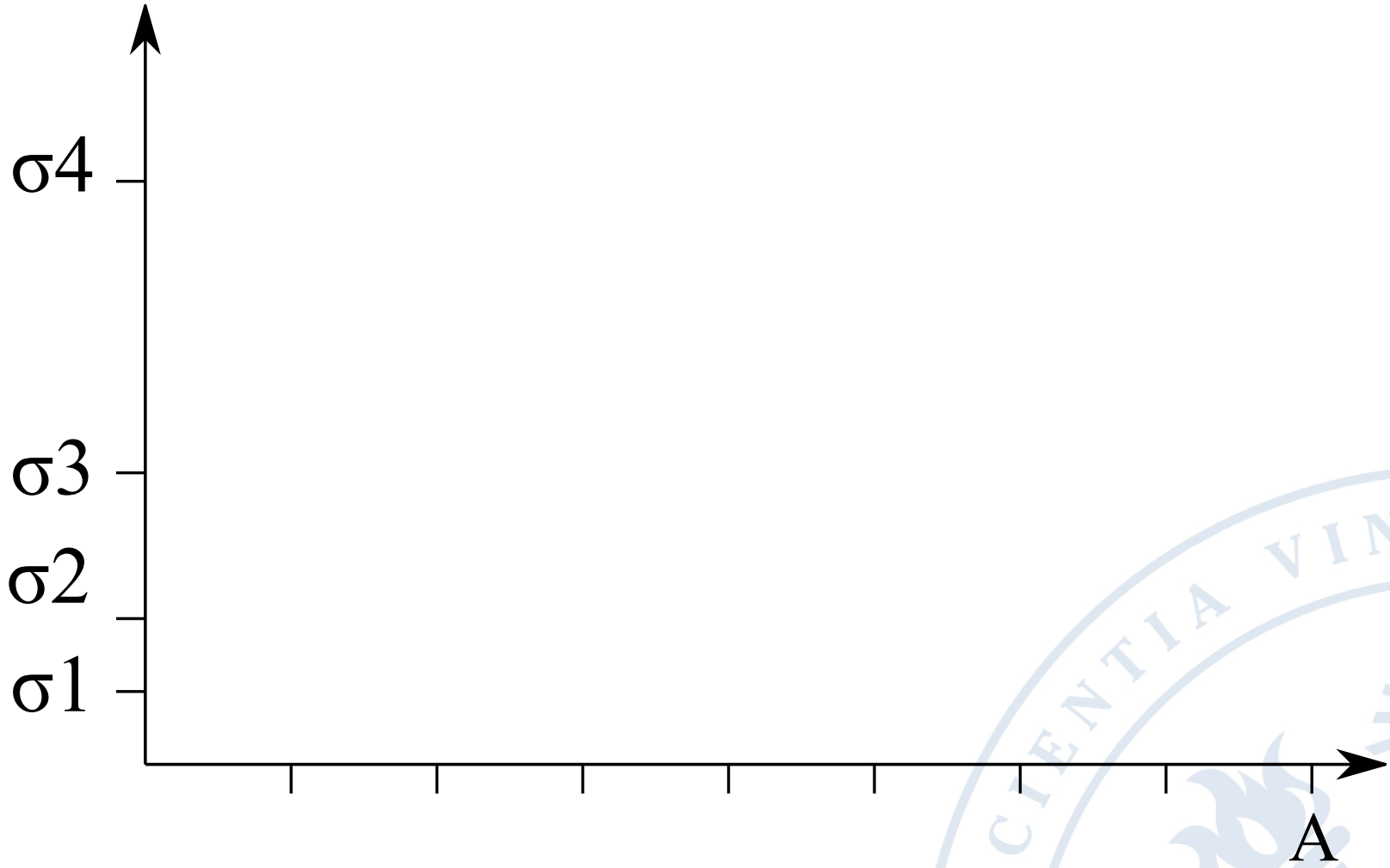
$$\left\lceil \frac{\mathcal{W}_i}{A - t} \right\rceil_\sigma$$

- En général, on doit s'arrêter avant la "zone de danger" suivante. La fréquence minimale pour τ_i est donc de

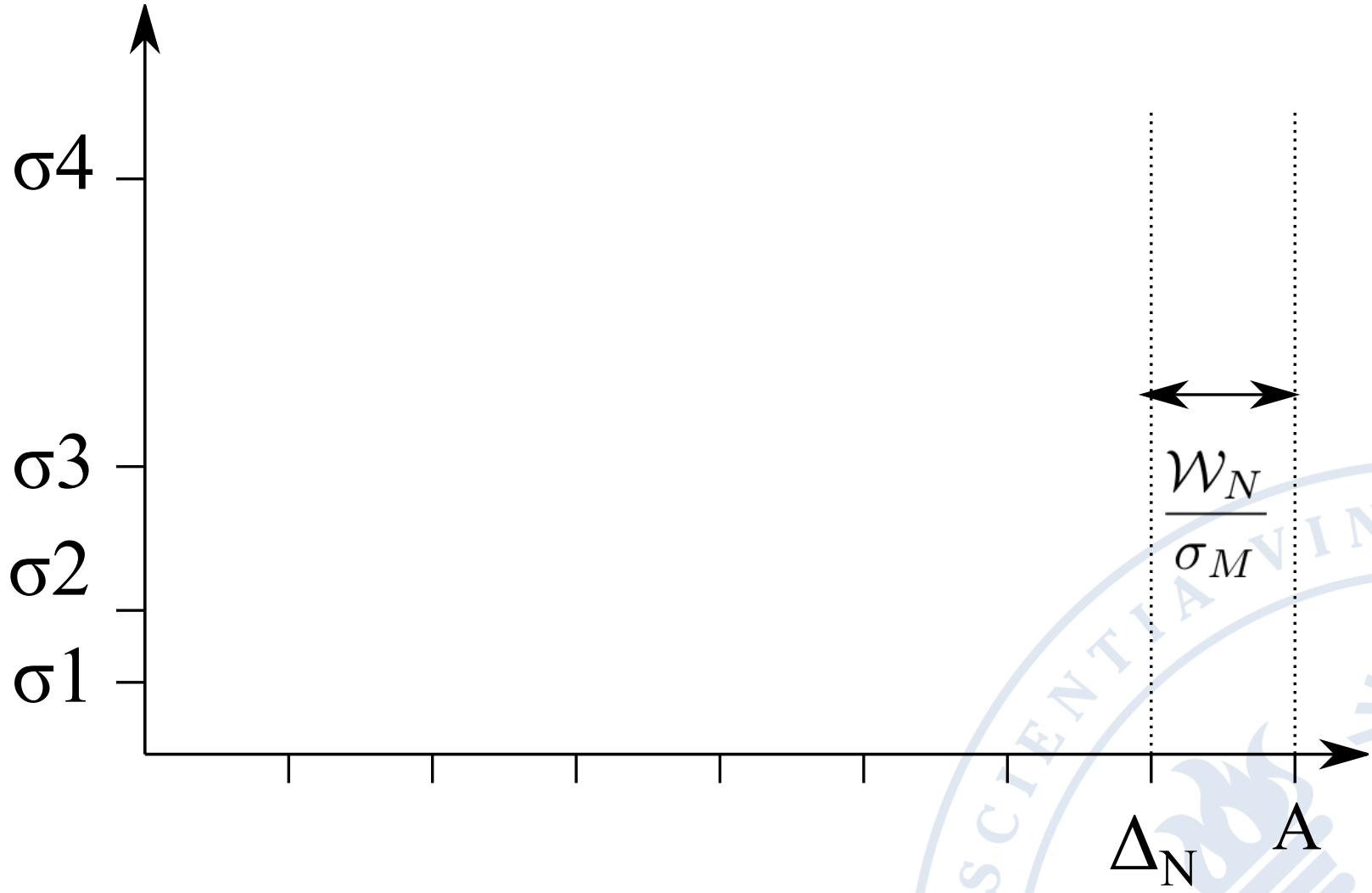
$$S_i(t) \stackrel{\text{def}}{=} \left\lceil \frac{\mathcal{W}_i}{\Delta_{i+1} - t} \right\rceil_\sigma$$

- On a donc un sélecteur de fréquence "agressif" : démarrer τ_i à la fréquence $S_i(t)$

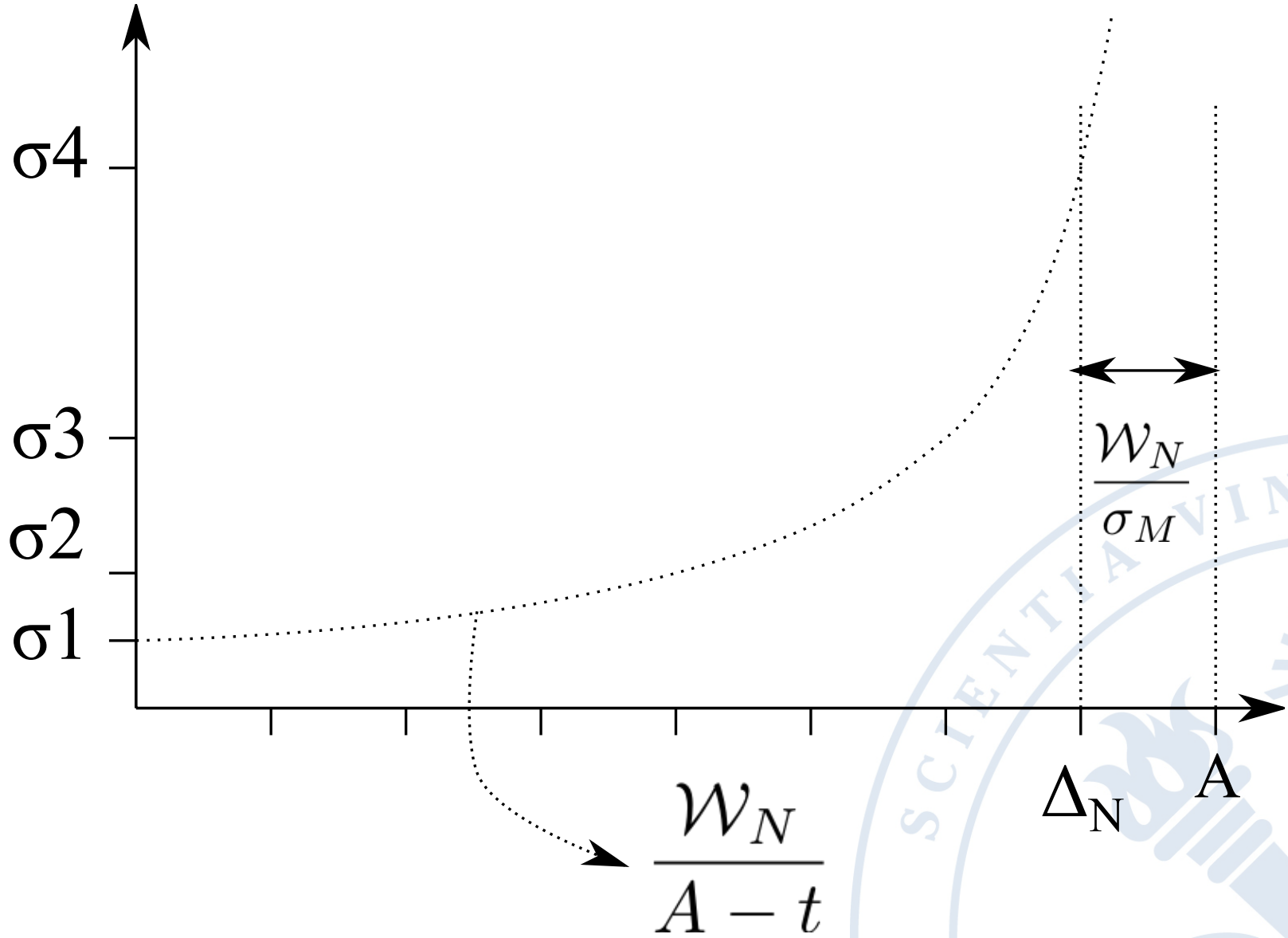
Exemple



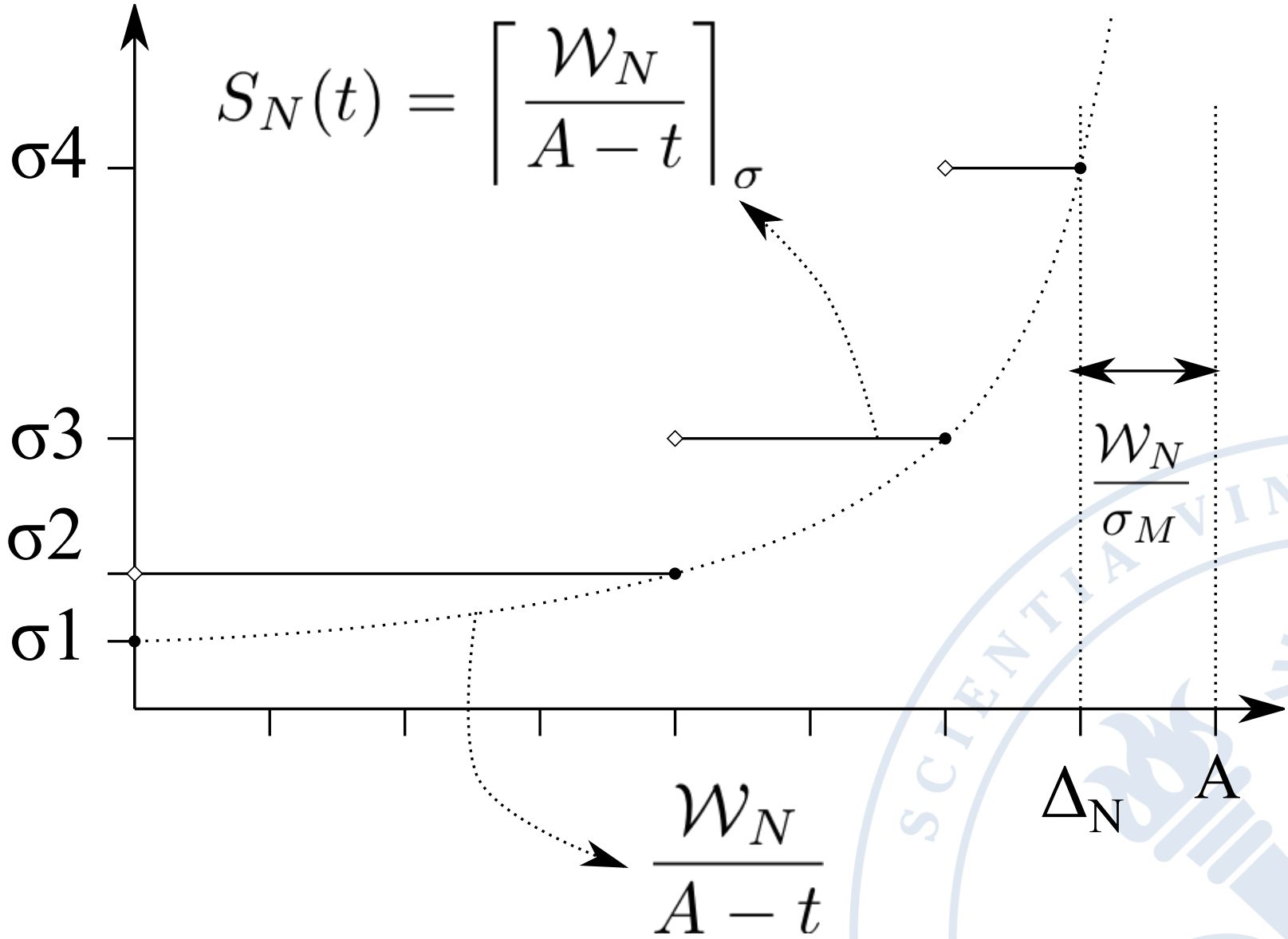
Exemple



Exemple

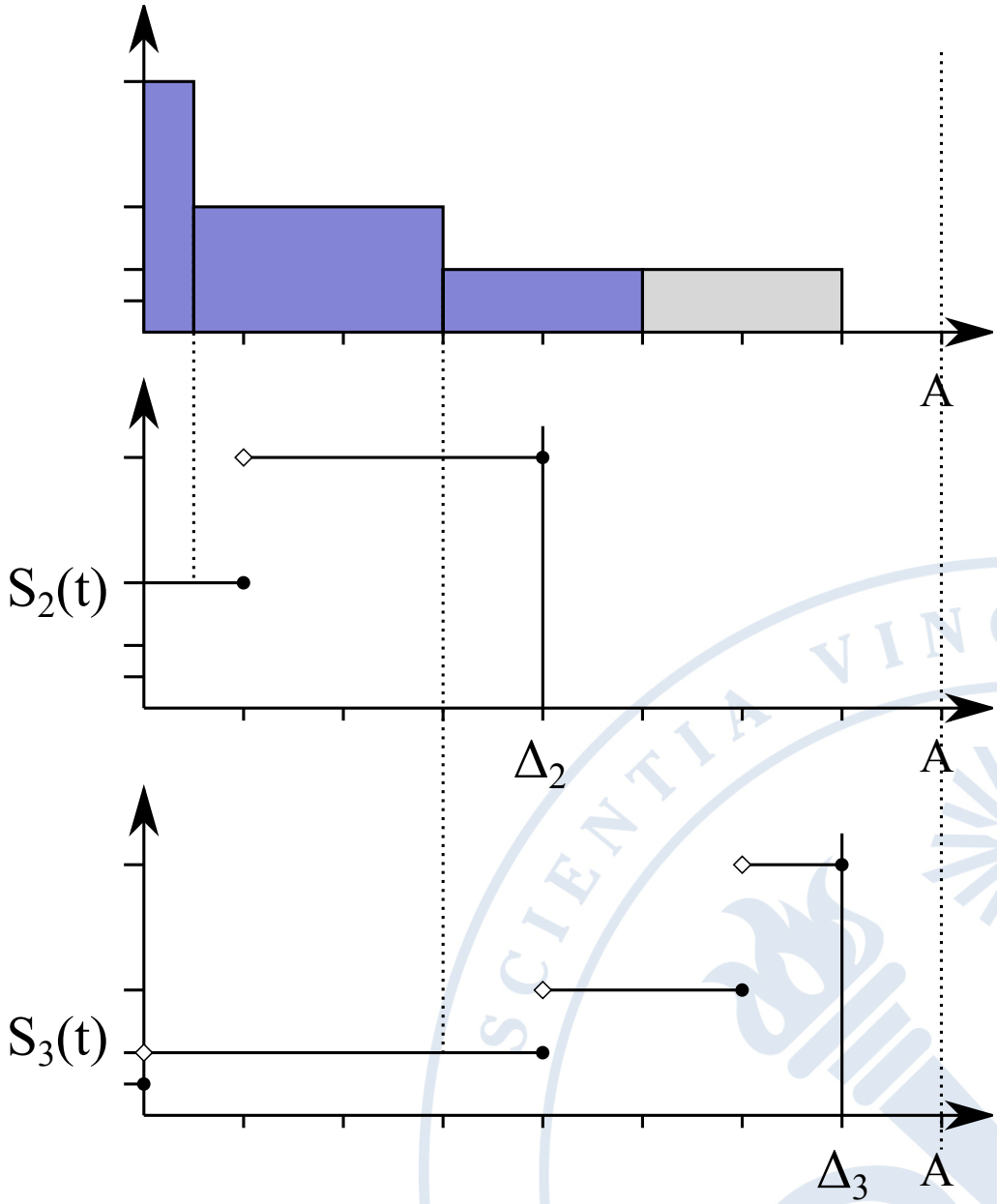


Exemple


$S_N(t) = \left\lceil \frac{\mathcal{W}_N}{A-t} \right\rceil_\sigma$
σ4
σ3
σ2
σ1
$\frac{\mathcal{W}_N}{\sigma_M}$
$\frac{\mathcal{W}_N}{A-t}$
Δ_N
A

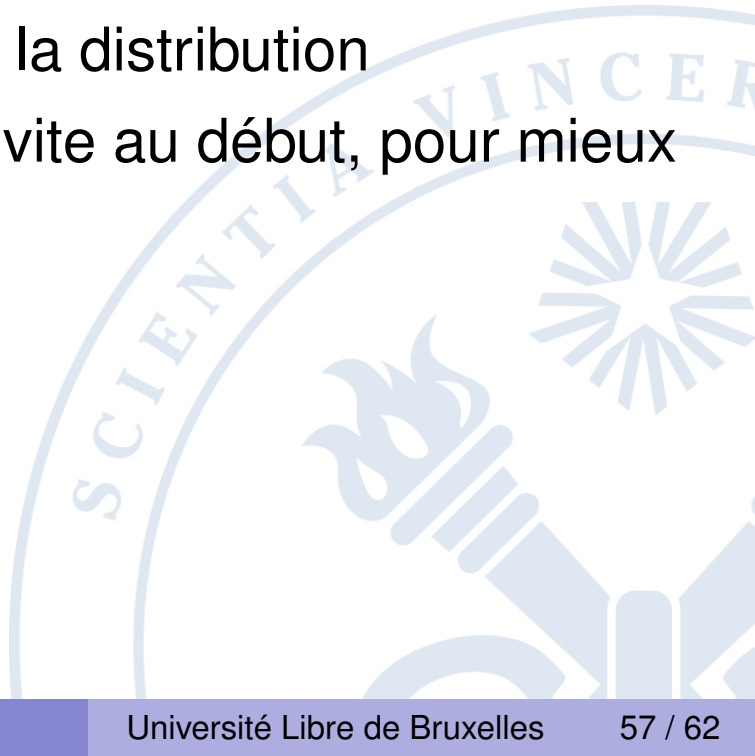
Exemple

- $\mathcal{W}_1 = 32, \mathcal{W}_2 = 24, \mathcal{W}_3 = 8$
- $A = 8$
- $\sigma : 1, 2, 4, 8$



Variantes

- Utiliser $S_i(t)$ tel quel est agressif : démarrage le plus lent possible
- “Pousse” un maximum de travail vers la fin
- Tendance à devoir accélérer le processeur vers la fin
- On tient juste compte du WCEC, pas de la distribution
- $S_i(t)$ est une borne ... on peut aller plus vite au début, pour mieux équilibrer



Utilisation de la moyenne

- Hypothèse plus équilibrée : tous les jobs vont prendre leur nombre de cycles moyen avg_i
- La vitesse moyenne idéale est de

$$\frac{\sum_i avg_i}{A}$$

- Idéalement, ou pourrait vouloir

$$S_i^{avg}(t) = \frac{\sum_{j \geq i} avg_j}{A - t}$$

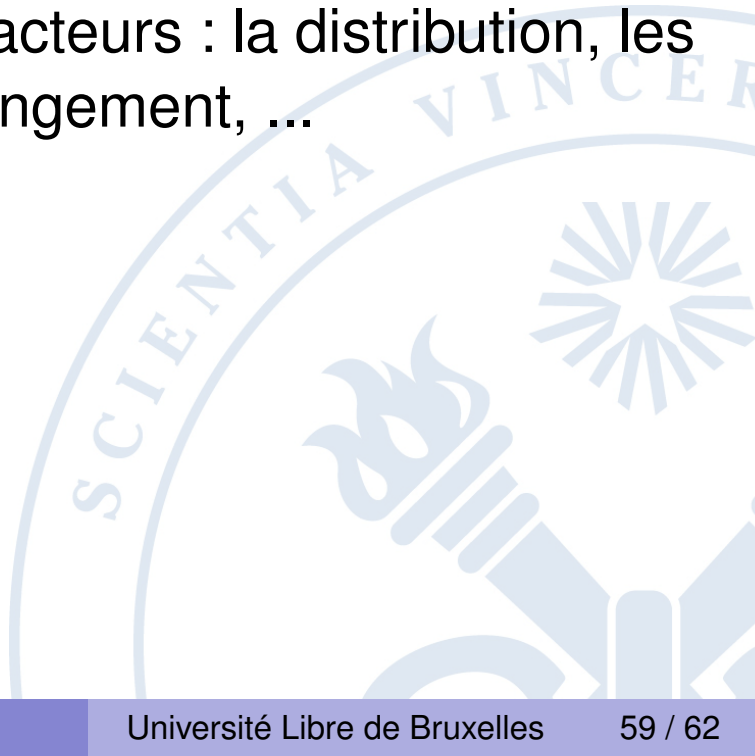
- Ne garantit pas l'échéance !

$$S_i^{avg}(t) = \max\left\{\frac{\sum_{j \geq i} avg_j}{A - t}, S_i(t)\right\}$$

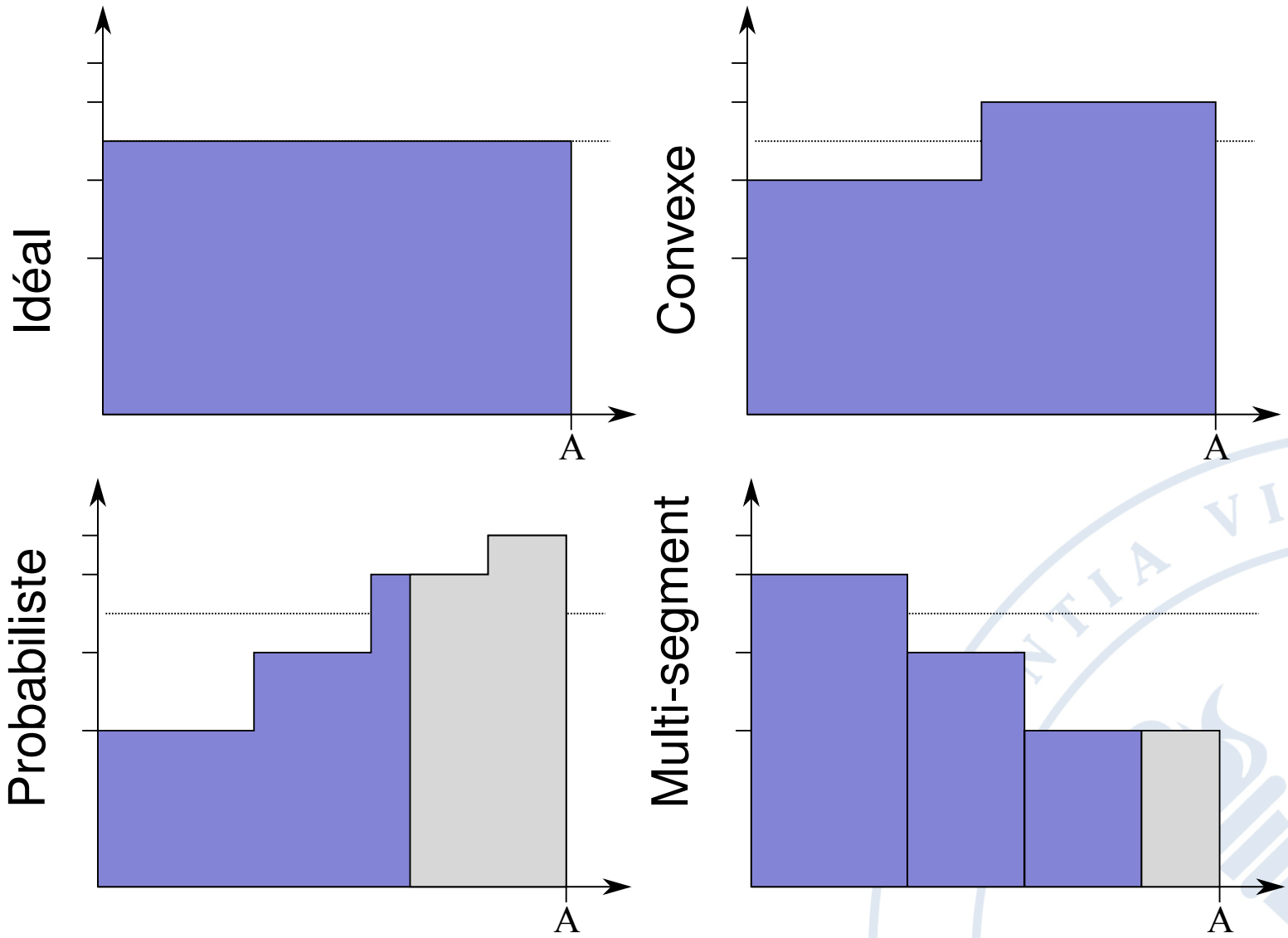
- Il faut encore "arrondir" (pas forcément vers le haut)

Variantes

- Il existe plein de méthodes dans la littérature !
- DPM-S, PITDVS, OITDVS, PITDVS^{clos}
- Elles tiennent compte de beaucoup de facteurs : la distribution, les coûts de changement, les temps de changement, ...



Résumé



Questions ?

Bibliographie


Enrico Bini, Giorgio Buttazzo, and Giuseppe Lipari.
Minimizing cpu energy in real-time systems with discrete speed management.
ACM Trans. Embed. Comput. Syst., 8 :31 :1–31 :23, July 2009.


Flavius Gruian.
Energy-Centric Scheduling for Real-Time Systems,.
PhD thesis, Lund University, 2002.


Tohru Ishihara and Hiroto Yasuura.
Voltage scheduling problem for dynamically variable voltage processors.
In *ISLPED '98 : Proceedings of the 1998 international symposium on Low power electronics and design*, pages 197–202, New York, NY, USA, 1998. ACM.