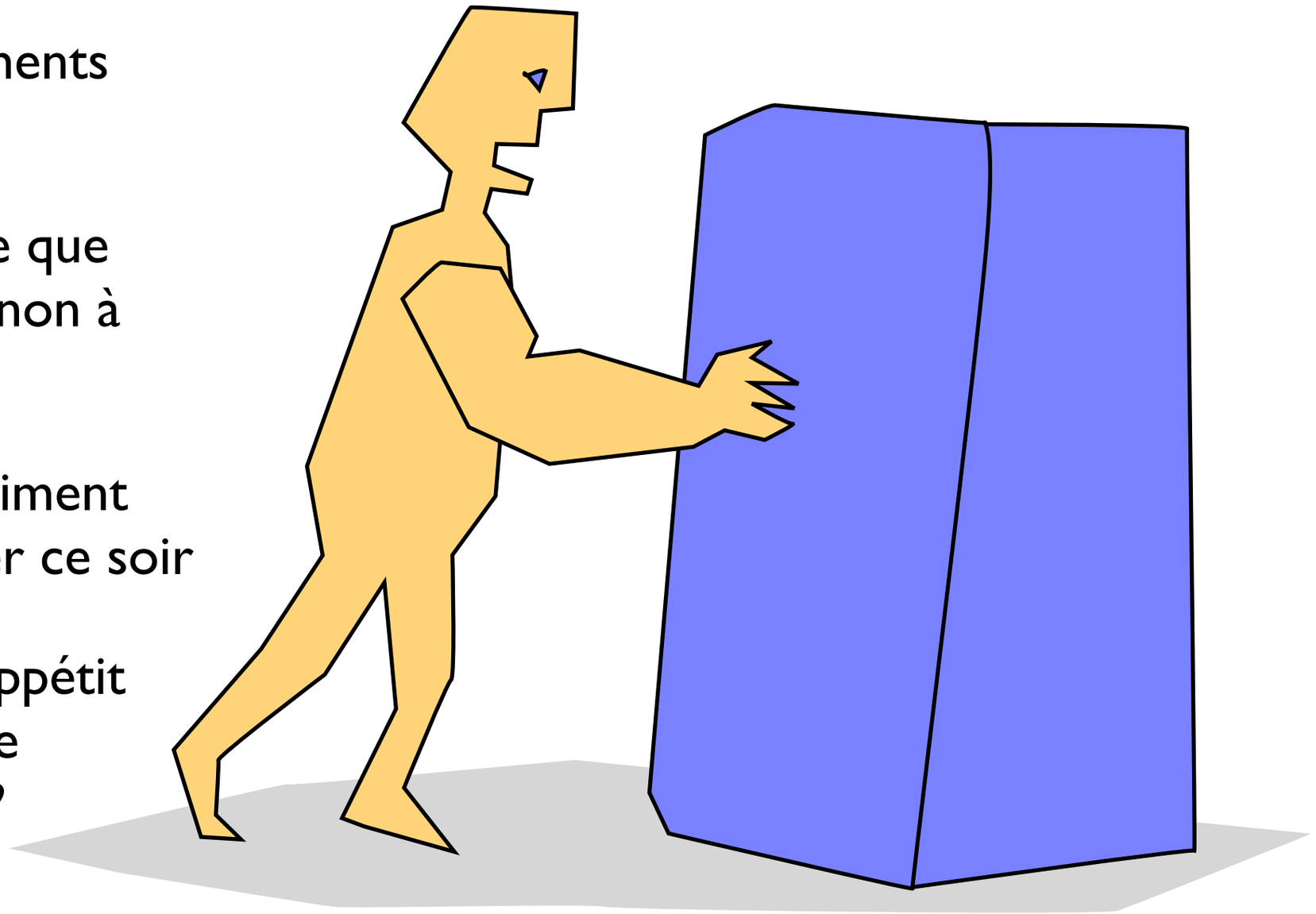
Gestion de tampon en ligne

pour illustrer des techniques centrales en algorithmique en ligne

C. Dürr — CNRS, LIP6 — roadef'2014

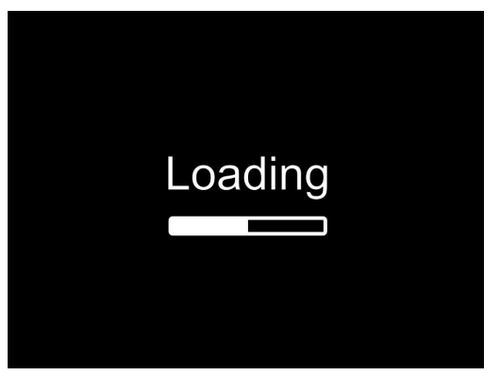
La scène devant le frigo

- jeter les aliments périmés
- découvrir ce que son compagnon à acheté
- choisir un aliment pour cuisiner ce soir
- suivre son appétit ou la date de péremption?



La scène dans un routeur

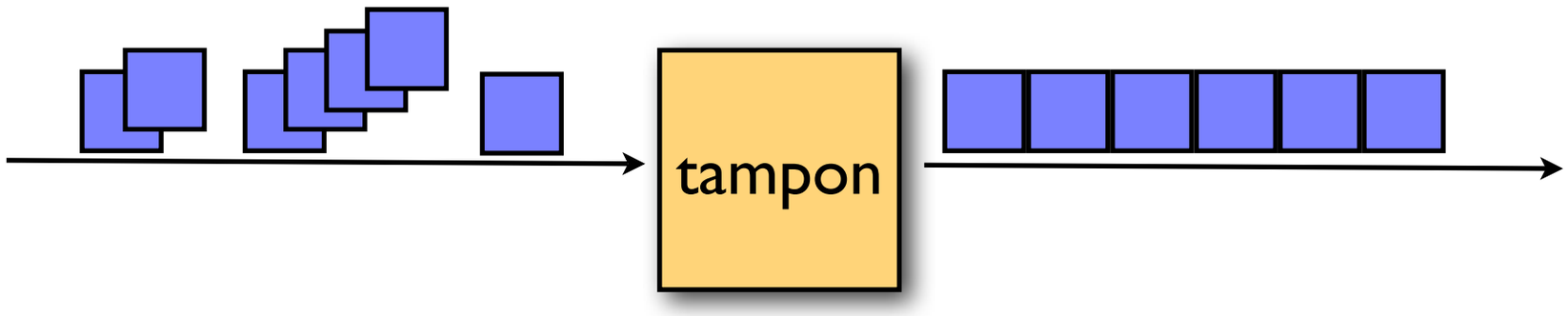
- Les pertes de paquets dans les réseaux sont dues au congestion.
- congestion: les routeurs doivent jeter des paquets
- améliorer politique de gestion de paquets dans les routeurs
  = améliorer qualité de service dans les réseaux



Plan

- le problème
- les algorithmes en ligne et le ratio de compétitivité
- construire des bornes inférieures
- analyser le ratio d'un algorithme avec un schéma de facturation
- les algorithmes en ligne randomisés
- un algorithme randomisé
- tableau des résultats principaux

Le modèle



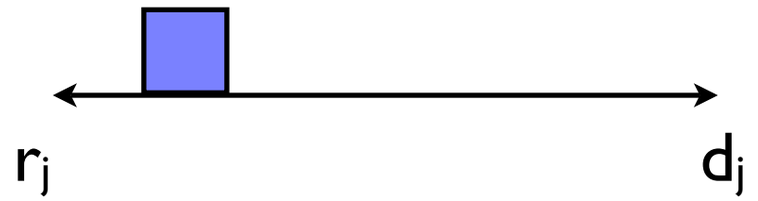
- des paquets de taille unitaire arrivent en ligne dans un tampon
- à chaque unité de temps un de ces paquet peut être envoyé
- on ne peut pas garder tous les paquets car…
  - il y a un tampon de capacité limité (modèle capacité bornée)
  - les paquets ont une date limite d'expiration (modèle délai borné)

Le modèle de délai borné

- **chaque paquet j**
  - arrive à un temps de relâchement r_j
  - une valeur w_j
  - et une date limite d_j

tous entier

- **à tout moment**
  - des paquets arrivent
  - des *anciens* paquets expirent
  - on peut envoyer exactement un paquet
  - **but:** maximiser la valeur totale des paquets envoyés

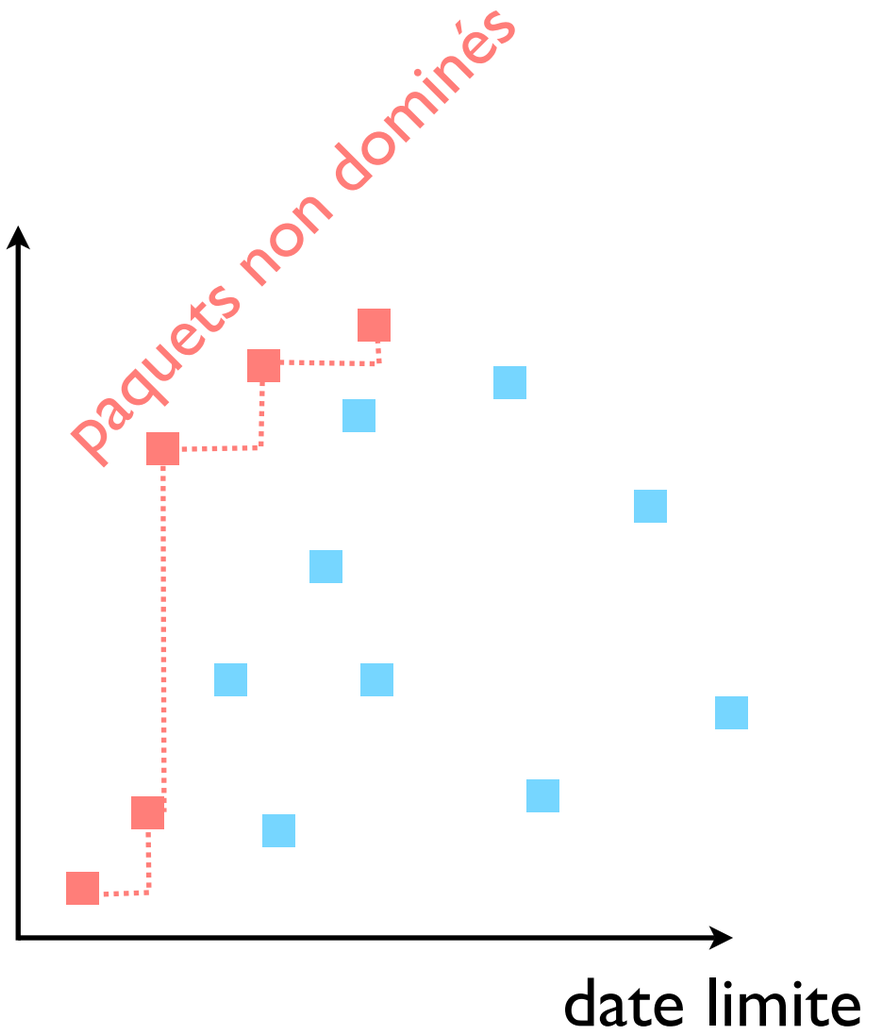


lié à

- ordonnancement en ligne pour tâches de taille unitaire
- couplage bi-parti en ligne

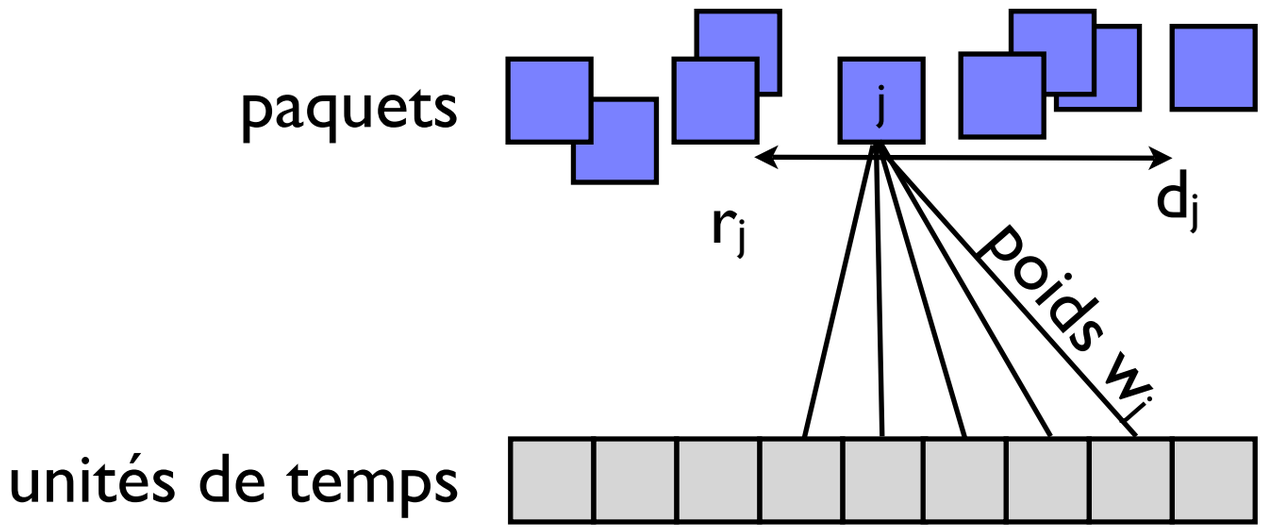
Le dilemme

- Quel paquet envoyer ?
- dominance : clairement si i est moins urgent et de valeur moindre que j, il vaut mieux envoyer j que i.
- quel compromis trouver entre valeur et urgence ?



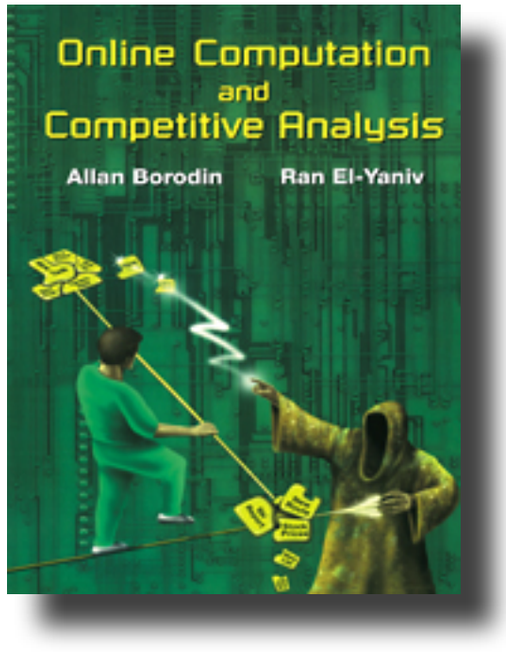
Le problème hors ligne

- Si on connaissait les paquets à venir, le problème serait facile :
- modélisation par graphe bi-parti paquets-unités de temps
- couplage de poids maximum = solution optimale



Les algorithmes en ligne

- les données arrivent en ligne
- à chaque pas de temps l'algorithme doit **prendre une décision sans connaître le futur**, sans pouvoir revenir sur sa décision plus tard
- on peut **voir comme un jeu** entre
  - **l'algorithme,** qui prend des décision pour produire une bonne solution (grande valeur objective)
  - et **l'adversaire,** qui génère l'entrée pour forcer l'algorithme a produire une mauvaise solution



Le ratio de compétitivité

- ratio d'un algorithme A
  = ratio entre valeur d'une solution produite par A
  et de la solution optimale
  := max OPT(I)/A(I) sur toutes les instances I

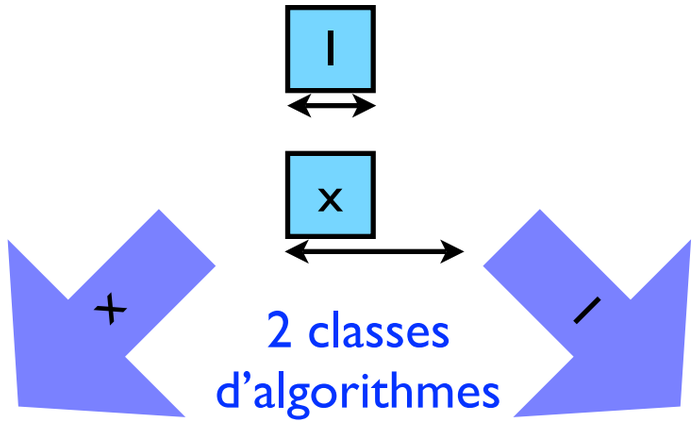
- ratio d'un problème
  := min ratio(A) sur tous les algorithmes A
  = prix a payer pour ne pas connaître le futur

- instance := séquence finie de paquets

- algorithme := fonction qui détermine un paquet à envoyer en fonction de l'historique.
  On ne demande pas à la fonction à être calculable en temps polynomial, mais en général c'est une simple règle de priorité

$\sqrt{2}$, une borne inférieure

- Au temps 0 l'adversaire relâche deux paquets,
  - valeur 1 d'écart 1
  - valeur x d'écart 2



ADV | 1 | x

ALG | x |

- si l'algorithme envoie x, l'adversaire envoie 1, puis x.
- ratio = (1+x)/x

x

ADV | x | x |

ALG | 1 | x |

- si l'algorithme envoie 1, l'adversaire envoie x, relâche un autre paquet d'écart 1 et de valeur x
- ratio = 2x/(1+x)

- choisir x pour maximiser le min des deux ratios: (1+x)/x=2x/(1+x)
  solution: x=1+$\sqrt{2}$, ce qui donne le ratio $\sqrt{2}$=1.414…

Φ, une borne inférieure

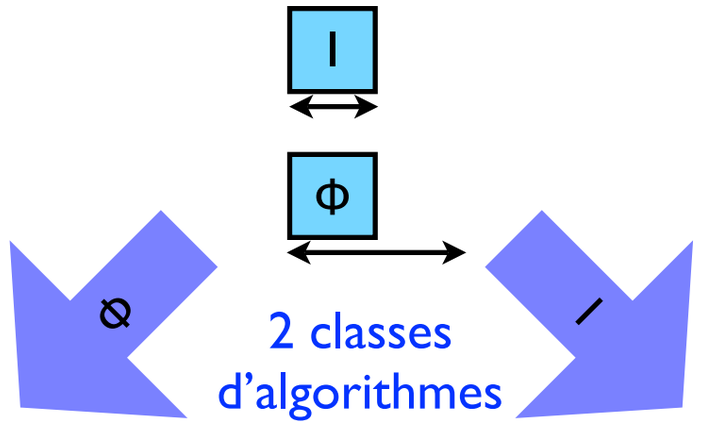
- **Φ=le nombre d'or** = $(1+\sqrt{5})/2=1.618...$
- solution à $1+x=x^2$
- suite de Fibonacci $1,1,2,3,5,8,\ldots,F_i,F_{i+1},\ldots$

ratio tend vers Φ

- rapport largeur/hauteur d'un livre de poche (p.ex.)

Φ, une borne inférieure

- Au temps 0 l'adversaire relâche deux tâches,
  - valeur 1 d'écart 1
  - valeur Φ d'écart 2



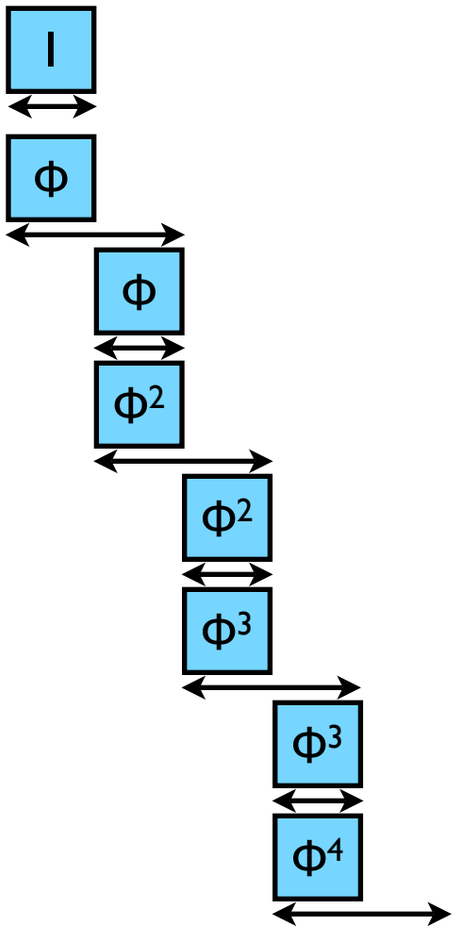
ADV | 1 | Φ |
ALG | Φ |

- si l'algorithme envoie Φ, l'adversaire envoie 1, puis Φ.
- ratio = (1+Φ)/Φ= Φ

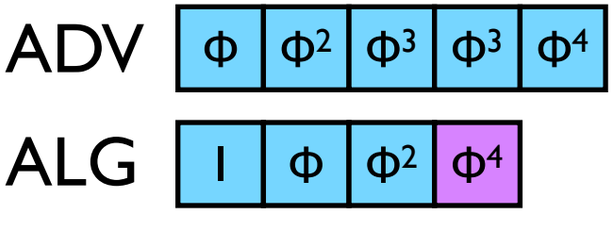
ADV | Φ |
ALG | 1 |

- si l'algorithme envoie 1, l'adversaire envoie Φ et fait disparaître Φ du tampon de l'algorithme
- ratio = Φ/1= Φ

Φ, une borne inférieure



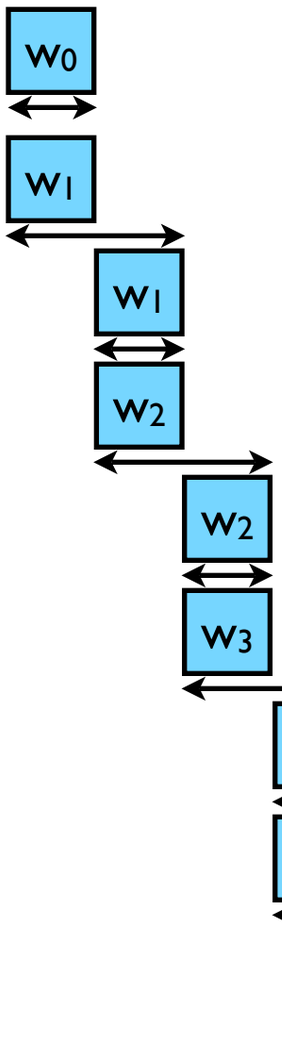
- On a triché, l'adversaire ne peut pas faire disparaître des tâches du tampon de l'algorithme
- Alors l'adversaire répète cette situation avec des valeurs de plus en plus grand, jusqu'à ce qu'à un temps t l'algorithme se décide d'exécuter la tâche de plus grande valeur

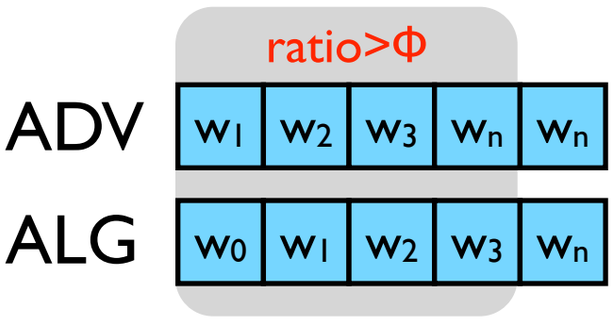


$$\text{ratio} = \frac{\Phi + \Phi^2 + \ldots + \Phi^t + \Phi^t + \Phi^{t+1}}{1 + \Phi + \ldots + \Phi^{t-1} + \Phi^{t+1}} = \Phi$$

ratio=Φ

Φ, une borne inférieure





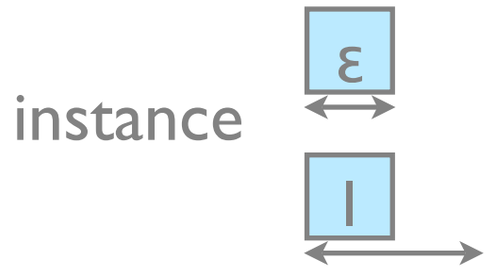
- Il y a encore un problème, l'algorithme pourrait toujours envoyer le paquet plus urgent et les instances doivent être finies.
- Alors au lieu de la valeur Φ^i, l'adversaire choisit la valeur $w_i := (1-\varepsilon)\Phi^i + \varepsilon(\Phi+1)^i$ pour un petit $\varepsilon>0$
- et arrête la séquence à un temps fini n
- finalement l'analyse considère trois cas

1. l'algorithme choisit le paquet urgent au temps 0
2. l'algorithme choisit la première fois le paquet urgent au temps t
3. l'algorithme choisit toujours le plus grand paquet

ratio $\square$ Φ pour n $\square$ ∞

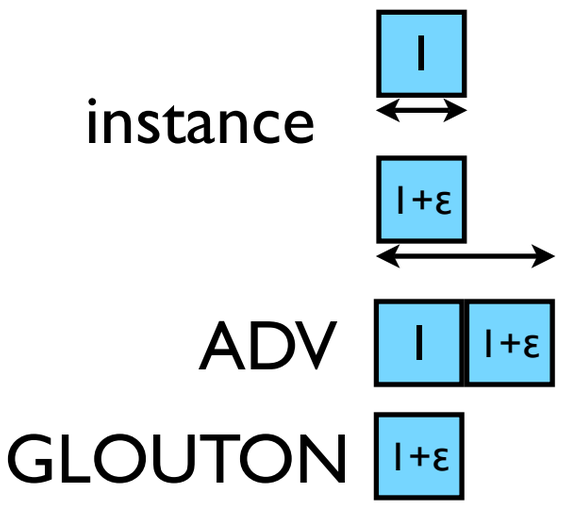
2, une borne supérieure

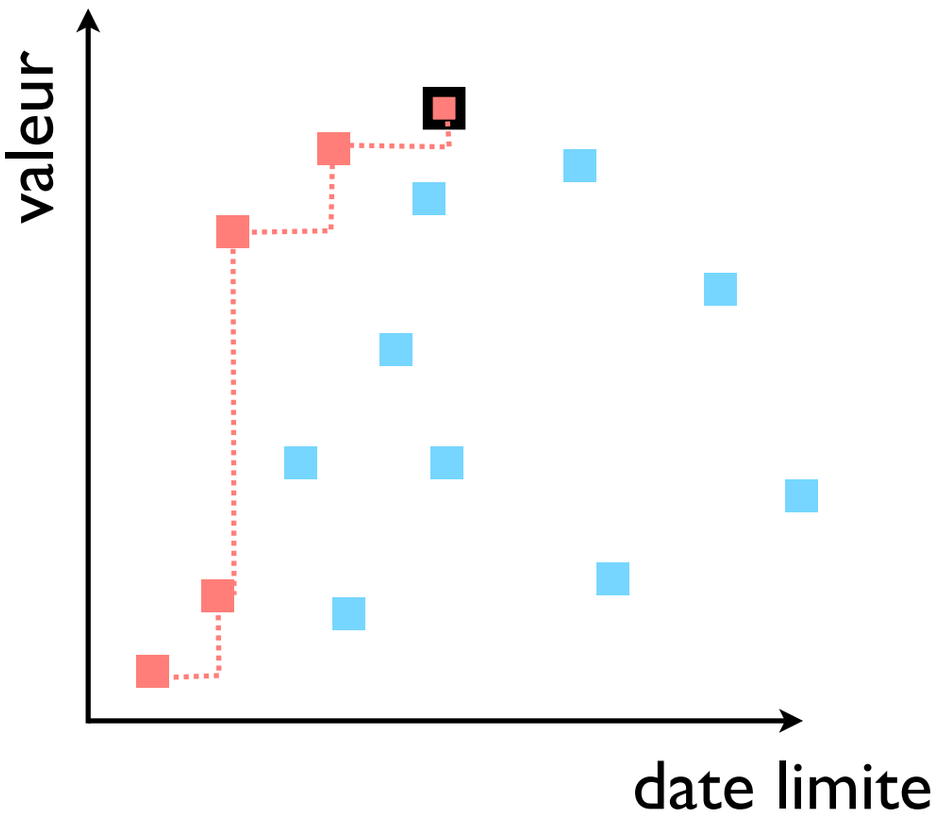
- Algorithme **pressé naïf**:
  toujours envoyer le paquet le plus urgent.
  Son ratio est non-borné.



- Algorithme **glouton**: toujours envoyer le paquet de plus grande valeur

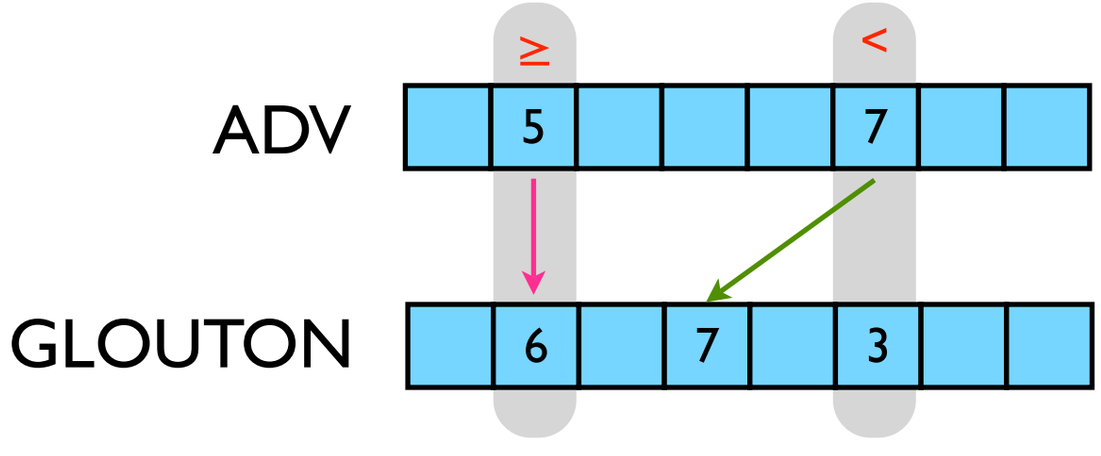
- Le ratio de competivité de glouton est au plus 2





2, une borne supérieure

- Le ratio de competivité de glouton est au moins 2
- Pour l'analyse, fixer une instance et comparer les solutions
- Pour chaque unité de temps
  - soit l'algorithme envoie un paquet d'une valeur au moins aussi grande que l'adversaire
  - soit l'algorithme envoie un paquet d'une valeur moindre, mais alors il a envoyé ce paquet avant.



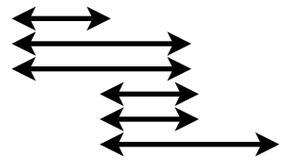
Chaque paquet j envoyé par l'adversaire est facturé à un paquet envoyé par l'algorithme de valeur au plus w_j.

Chaque paquet i envoyé par l'algorithme reçoit au plus deux factures dont la valeur totale ne dépasse pas $2w_i$.

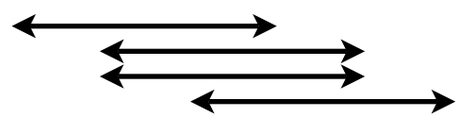
le ratio est ≤ 2

Des cas particuliers

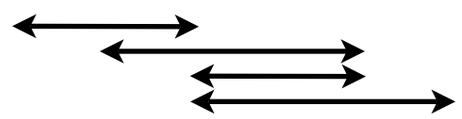
- une instance est s-bornée si l'écart est $d_j - r_j \leq s \ \forall s$



- une instance est s-uniforme si l'écart est $d_j - r_j = s \ \forall s$



- une instances est agréable (*agreable*) si les dates limites sont ordonnées comme les dates de relâchement



- La construction de la borne inférieure est 2-bornée, est-ce qu'on peut trouver un algorithme qui exploite cette structure?

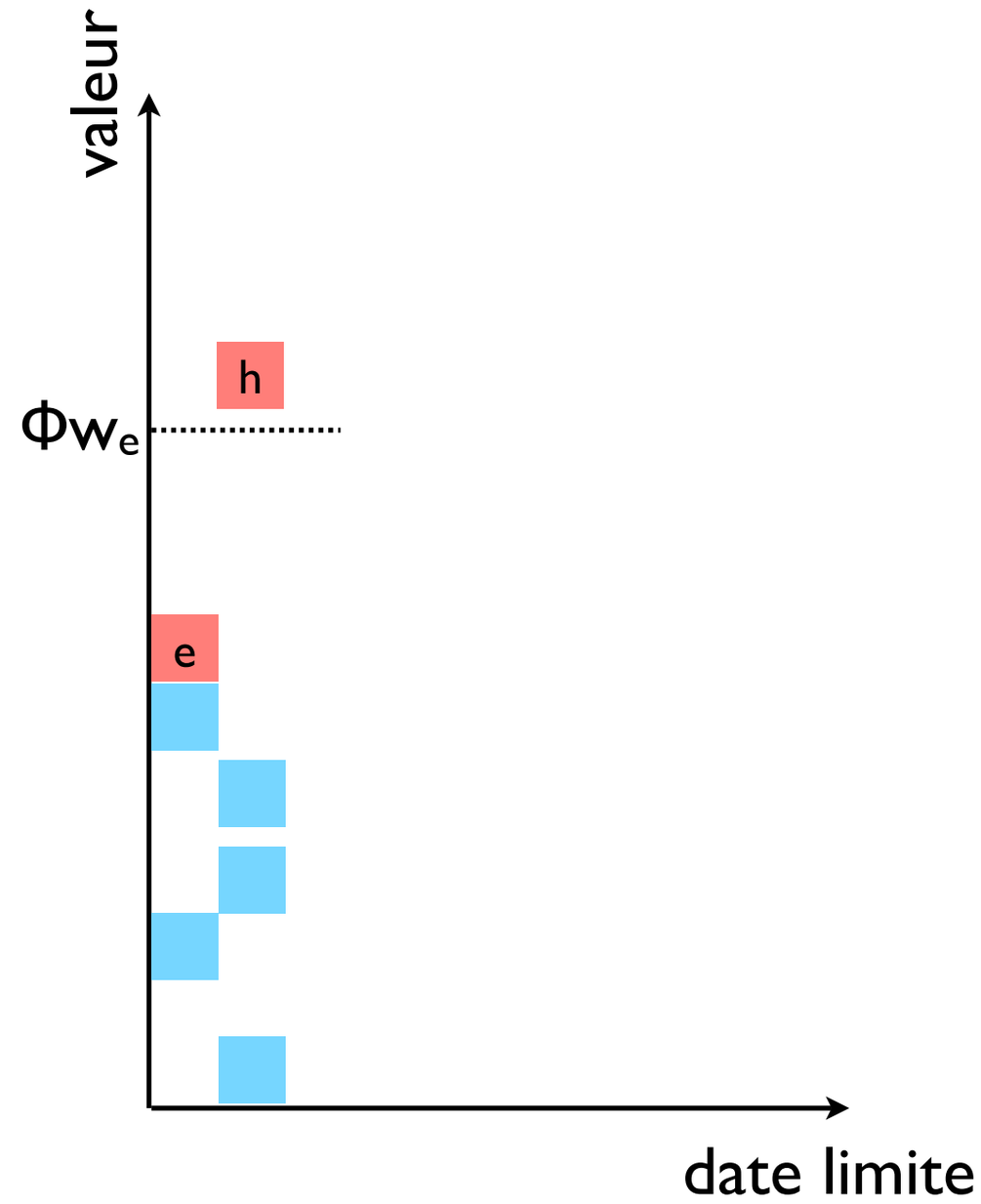
Algorithme Balance

- **Simple règle de priorité**

- à chaque temps t, il y a des paquets qui expirent au temps t+1 et des paquets qui expirent au temps t+2.

- Soit e la plus grande valeur d'un paquet urgent et h la plus grande valeur d'un paquet moins urgent

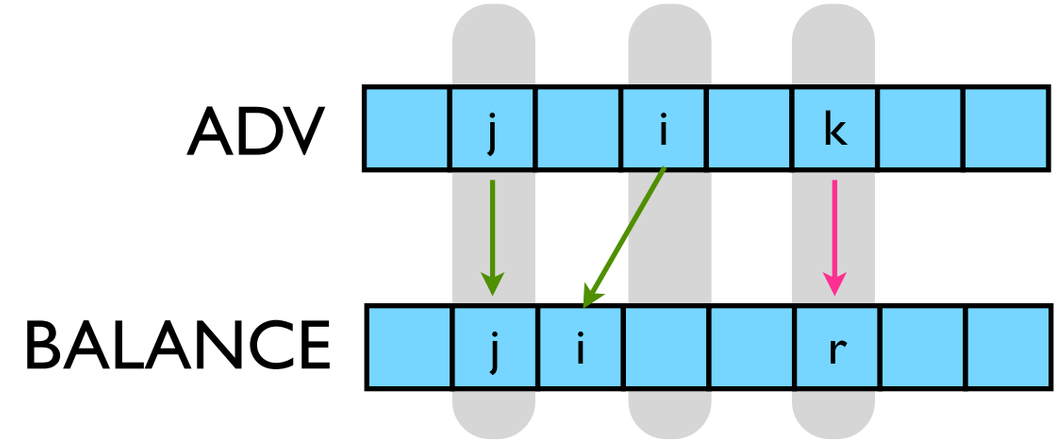
- si $w_h \geq \Phi w_e$, alors envoyer h, sinon envoyer e



Algorithme Balance

- Analyse par facturation
- Les paquets envoyés par l'adversaire sont facturés aux paquets envoyés par l'algorithme
- chaque paquet i envoyé par l'algorithme reçoit au plus deux factures, dont la somme ne dépasse pas Φw_i. **Preuve**:
- ok, s'il reçoit qu'une facture verte ou rouge.
- cas il reçoit une facture verte et une rouge de k : Comme il a envoyé i à la place de k, on a $w_i > \Phi w_k$, et la facture totale est

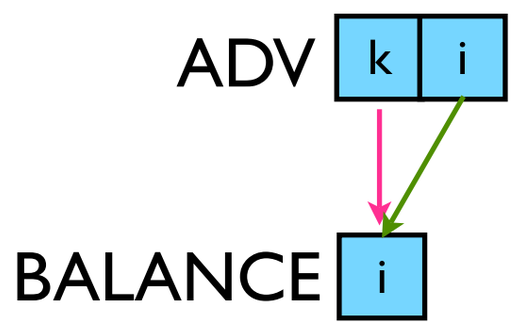
$$w_i + w_k = (1 + 1/\Phi) w_i = \Phi w_i$$



Pour chaque paquet i envoyé par l'adversaire au temps t

si i a été envoyé par l'algorithme au temps t-1 ou t, facturer vers lui (type vert)

sinon facturer vers le paquet que l'algorithme envoie (type rouge)



Algorithmes en ligne randomisés

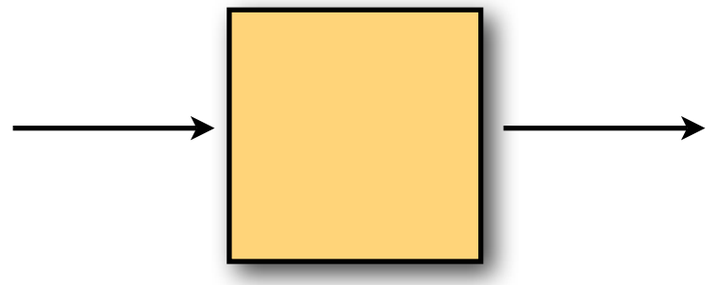
- font des décisions aléatoires pour se protéger du pire des cas

- ratio de compétitivité d'un algorithme A
  := max OPT[I] / E[A(I)]

- deux modèles d'adversaire, qui *lui* sans perte de généralité est déterministe

  - **adversaire aveugle** (*oblivious*) construit l'instance I en avance en fonction de l'algorithme, mais sans connaître sa source d'aléa

  - **adversaire adaptatif** construit l'instance I au fur et à mesure en fonction des décisions de l'algorithme

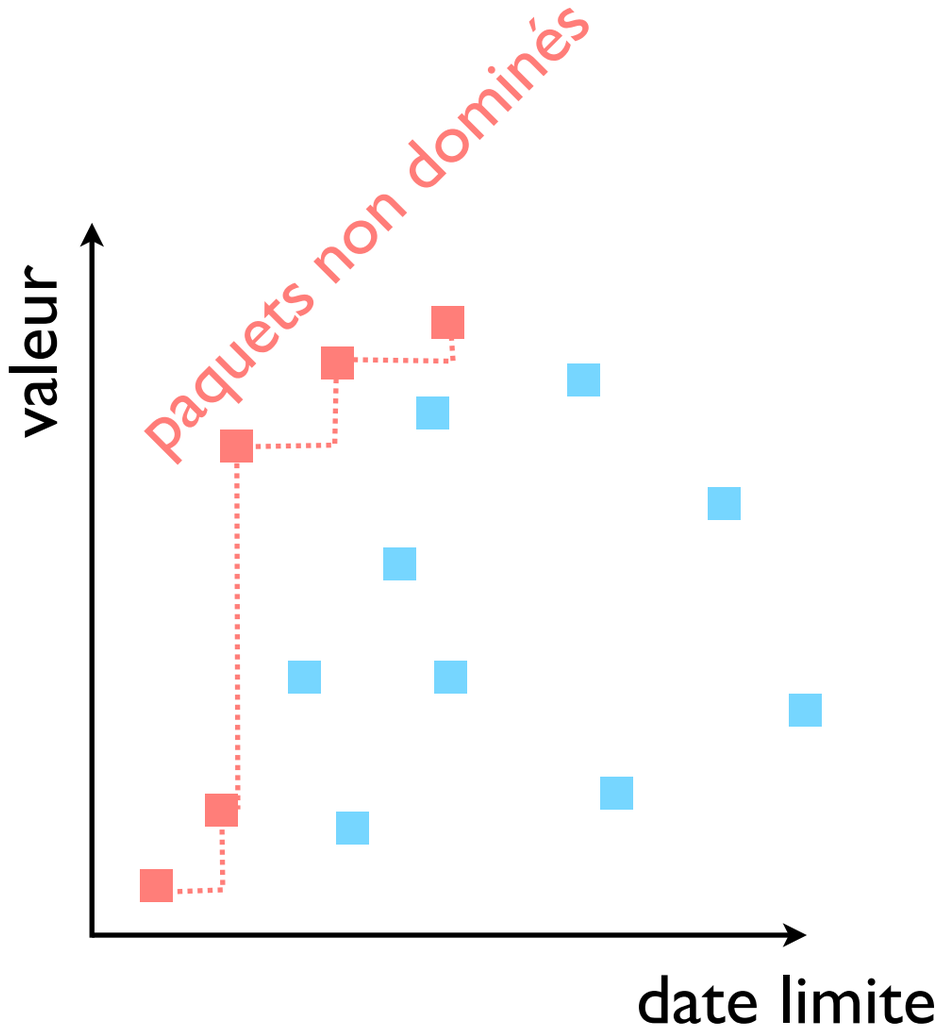


le comportement d'un routeur influe sur le trafic qu'il reçoit, car des paquets perdus sont renvoyés, ou que les tables de routage sont adaptées.

Le modèle d'adversaire adaptatif semble adapté, donc on l'adopte.

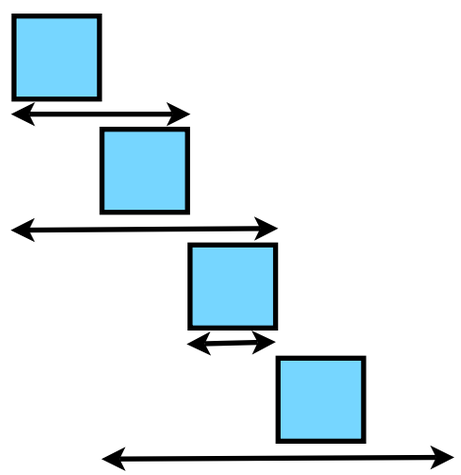
Hypothèses sur l'adversaire

- sans perte de généralité un adversaire adaptatif n'envoie **jamais un paquet dominé** (argument d'échange)



- sans perte de généralité un adversaire aveugle qui envoie un ensemble de paquets S, les envoie **dans l'ordre de leur date limite** (*earliest deadline first*) (argument d'échange)

- on ne peut pas dire autant d'un adversaire adaptatif, car l'ensemble S n'est pas déterminé à l'avance



L'algorithme RMix

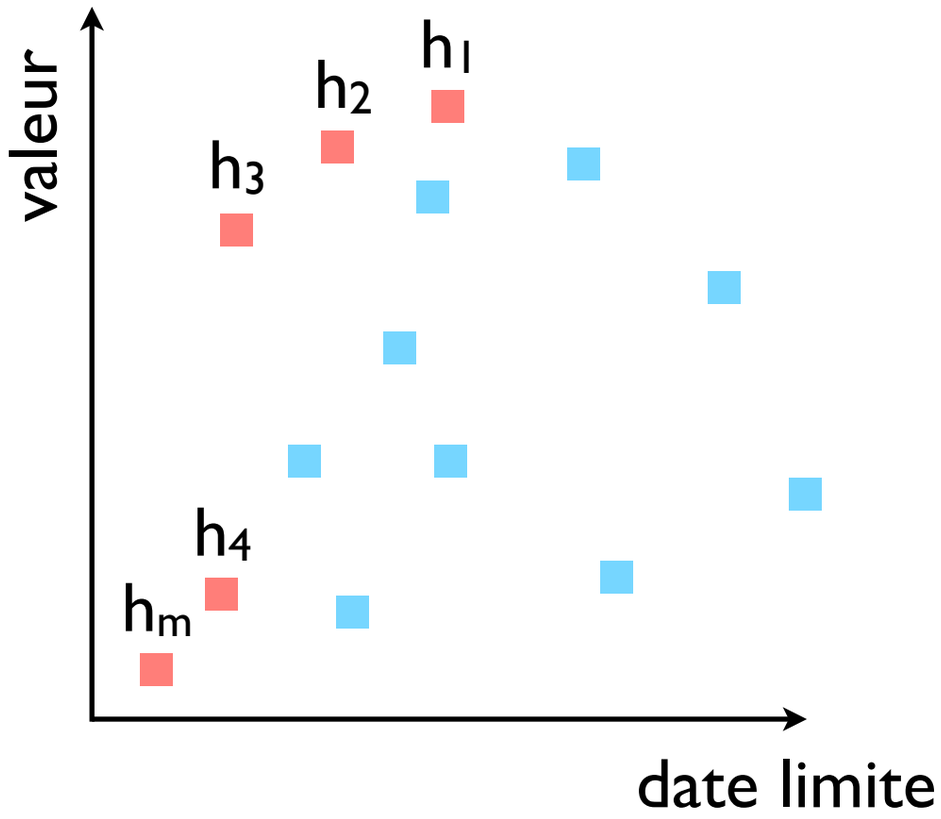
- s'il y a des paquets dans le tampon, considérer les paquets non-dominés en ordre décroissant $h_1,\ldots,h_m$

  - Pour chaque $n=1,2,\ldots$ poser $p_n := 1-w_{n+1}/w_n$, jusqu'à ce

    - qu'on ait $p_1+\ldots+p_i \geq 1$ ou $n=m$

    - finalement poser $p_n := 1-p_1-\ldots-p_{n-1}$, tel que $p_1+\ldots+p_n=1$

- choisir paquet h de $h_1,\ldots,h_n$ avec probabilité $p_1,\ldots,p_n$ et l'envoyer



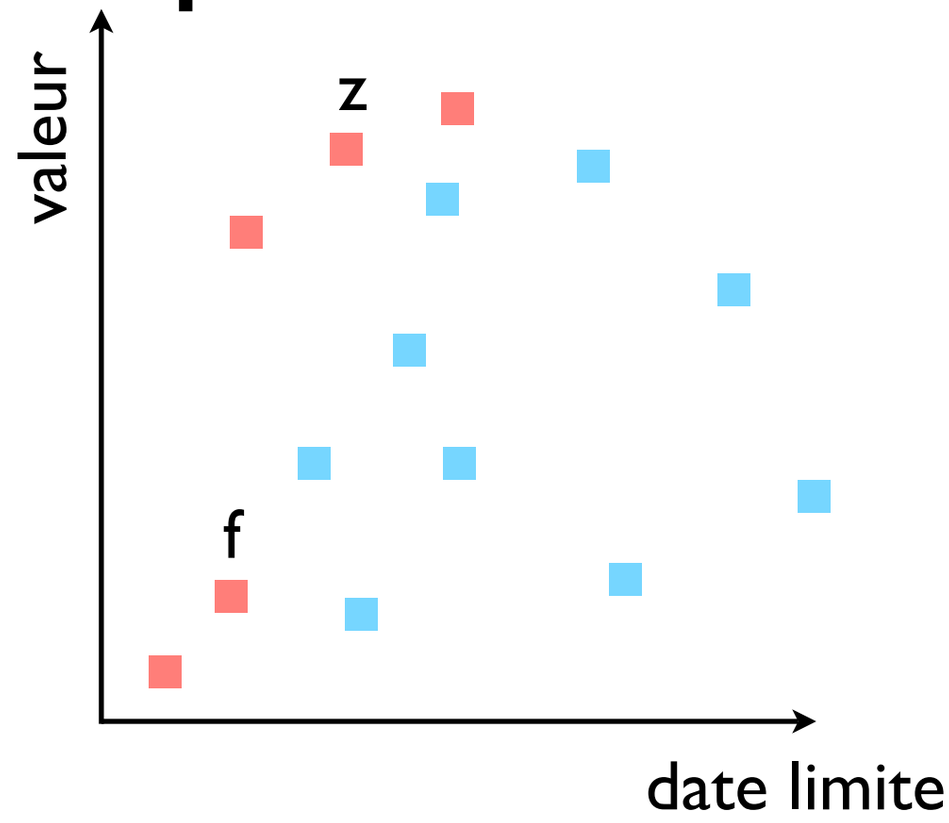
Tampons identiques

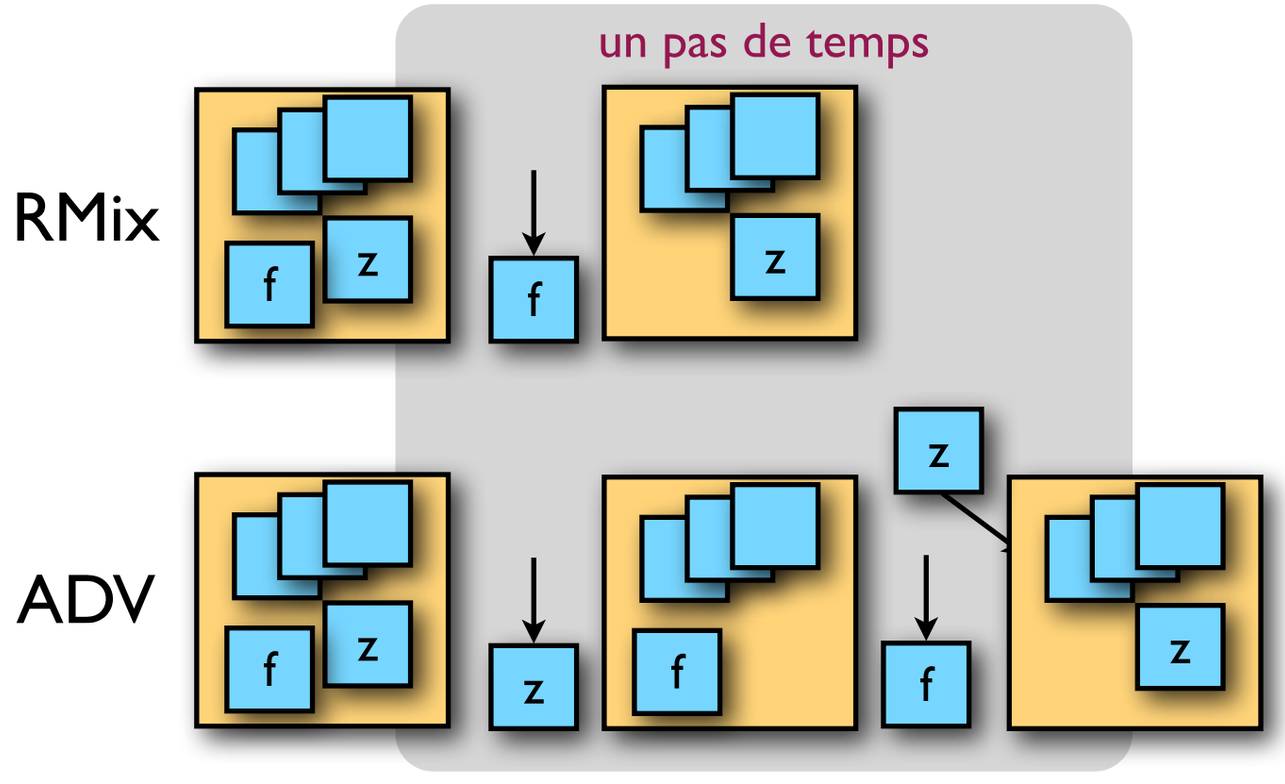
- on effectue des opérations avantageux pour l'adversaire afin que les tampons de l'adversaire et de l'algorithme soient identique à chaque pas de temps

- algorithme envoie f, adversaire envoie z

- si $d_f \leq d_z$, remplacer f par z dans le tampon adverse (*upgrade*)

- si $d_f > d_z$, laisser l'adverse envoyer f aussi, et rajouter z dans son tampon





Les étapes de l'analyse

- On doit montrer que même sous ces opérations avantageuses

  1. **E**[gain ADV]$\leq w_1$
  2. **E**[gain RMix]$\geq w_1(1-(1-1/n)^n)$

l'aléa vient du paquet supplémentaire

- Or $1-(1-1/n)^n$ tend vers $e/(e-1)=1.582$ d'en bas
- Donc le ratio de compétitivité randomisé contre un adversaire adaptatif est 1.582
- L'algorithme se comporte mieux quand il y a moins de choix (n petit)

Gain adverse

- E[gain ADV | envoie z] = $w_z + \sum_{i<z} p_i w_i$
- E[gain ADV] $\leq \max_z$ E[gain ADV | envoie z]
- mais **E**[gain ADV | z] - **E**[gain ADV | z+1] = $w_z - w_{z+1} - p_z w_z \geq 0$
- donc **E**[gain ADV | z] $\leq$ **E**[gain ADV | envoie 1] = w_1

utilise définition des probabilités

$p_z \leq 1 - w_{z+1}/w_z$

Gain de l'algorithme RMix

- E[gain RMix] = $\Sigma\, p_i w_i$
- or $p_i w_i = w_i - w_{i+1}$ pour tout $i<n$ et $p_n = 1 - p_1 - \ldots - p_{n-1}$, donc
- E[gain RMix] = $(\Sigma_1^{n-1} w_i - w_{i+1}) + (1 - \Sigma_1^{n-1} p_i)\, w_n = w_1 - w_n \Sigma_1^{n-1} p_i$
- or $w_i = w_{i-1}(1 - p_{i-1})$ pour tout $i \leq n$ et donc
- $w_n = w_1 \prod_1^{n-1}(1 - p_i)$ et
  E[gain RMix] = $w_1(1 - \prod_1^{n-1}(1-p_i)\Sigma_1^{n-1} p_i)$
- mais $\Sigma(1-p_i) + \Sigma p_i = n-1$ et $\prod(1-p_i)\Sigma p_i \leq (1-1/n)^n$
- pour finalement
  E[gain RMix] = $w_1(1-(1-1/n)^n)$

utilise inégalité entre moyenne arithmétique et géométrique

$\sqrt[n]{x_1 x_2 \ldots x_n} \leq ((x_1 + \ldots x_n)/n))$

Résultats randomisés

	borne inf	borne sup	
déterministe	1.618	1.828	
rand. adaptatif	1.333	1.582	[jez,2009] [bienkowski,chorbak,jez,2008]
rand. aveugle	1.25	1.582	[chin,chrobak,fung,jawor,sgall,tichy,2006] [chin,fung,2006]

Résultats déterministes

	borne inf	borne sup	
général	1.618	1.828	[englert,westerman,2007] par le cas 2-borné
2-borné	1.618	1.618	[kesselman,lotker,mansour,patt-shamir, schieber, sviridenko,2007] [Chin,Fung,2003]
s-borné	1.618	2-2/s+o(1/s)	[chin,chrobak,fung,jawor,sgall,tichy,2006]
agréable	1.618	1.618	[li,sethuraman,stein,2007] par le cas 2-borné
2-uniforme	1.377	1.377	[chrobak,jawor,sgall,tichy,2007]
s-uniforme	1.377	1.618	par le cas agréable par le cas 2-borné

Directions

- fermer le fossé entre les bornes
  (déterminer le meilleur algorithme pour le cas général)

- appliquer la technique d'analyse primale-duale à ces problèmes

- appliquer l'analyse bijective pour comparer des algorithmes