Introduction à l'algorithmique en ligne

Christoph Dürr
Sorbonne Université, CNRS

ROADEF 2021

Exemple : k-serveur

[Manasse, McGeoch, Sleator, 1990]

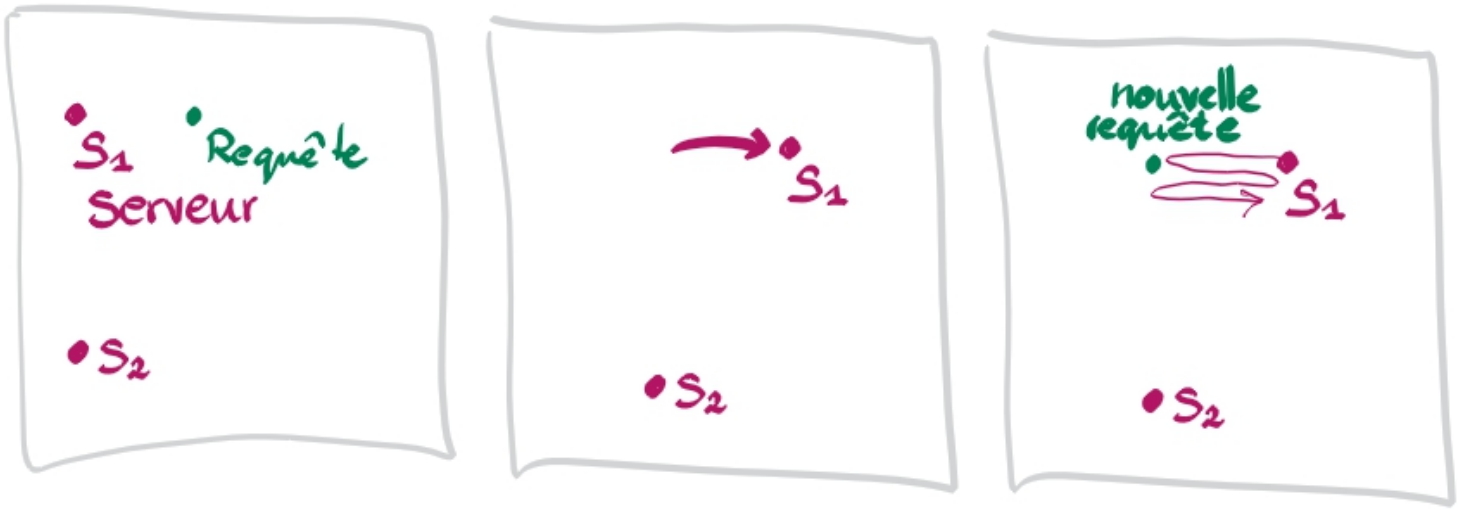
k serveurs (réparateurs mobiles).

Des requêtes d'intervention arrivent au fil de l'eau sous forme de points dans un espace métrique.

Vous devez: choisir un serveur pour servir une requête.

Déplacements et services (réparations) se font de manière instantanée.

But: minimiser les déplacements.



Problématique: comment mesurer la performance ? (Servir avec le serveur le plus proche peut générer un très grand coût. Glouton peut avoir une mauvaise performance.)

Le paradigme des problèmes en ligne

Applications: ordonnancement, couplage, équilibrage de charge, gestion de cache, finance, …

L'instance d'un problème = séquence de requêtes.

À chaque nouvelle requête l'algorithme doit prendre une décision irrévocable…

… sans connaissance de requêtes à venir, ni même de leur nombre.

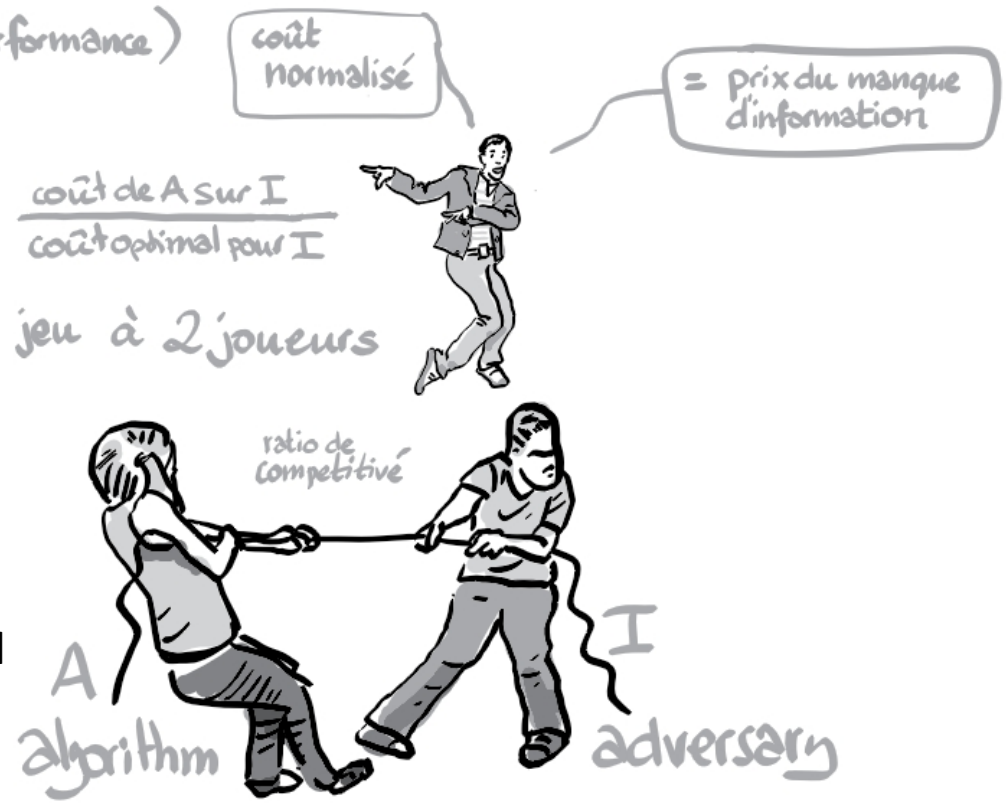
L'optimum aurait pu être calculé si l'instance entière était connue à l'avance

R1 R2 R3 R4 R5 R6 R7 R8 Rn

Ratio de competivité (mesure de performance)
(formulé pour un problème de minimisation)

$$CR := \inf_{\text{algorithme } A} \sup_{\text{instance } I} \frac{\text{coût de A sur I}}{\text{coût optimal pour I}}$$

= jeu à 2 joueurs



Ratio de compétitivité randomisé:
E[coût algorithme]/coût optimal

Adversaire connaît l'algorithme mais pas ses tirages aléatoires (*oblivious adversary*)

Etat de l'art du problème k-serveur

	Borne inférieure	Borne supérieure
Ratio de compétitivité déterministe	k	2k+1 (conjecturé k)
	[Sleator,Tarjan,1985]	[Koutsoupias,Papadimitriou,1994]
	Problème de cache 	WorkFunctionAlgorithm Trouve un équilibre entre envoyer le serveur le plus proche et envoyer le serveur que la solution optimale enverrait
Ratio de compétitivité randomisé	$\Omega\left(\frac{\log k}{\log \log k}\right)$	$0(\log^6 k)$ Passe par des *hierarchical separated tree metrics*
	[Bartal,Linial,Mendel,Naor,2005], [Bartal,Bollobás,Mendel,2006]	[Bubeck,Cohen,Lee, Madry,2018], [Lee,2018]

Techniques de base

équilibrer, illustré par acheter-ou-louer

[Karlin,Manasse,McGeoch, 1994]

Pendant D jours, vous devez utiliser une imprimante 3D.
Chaque matin, décider de louer pour 1 €/jour, ou acheter une fois pour toutes pour B €.

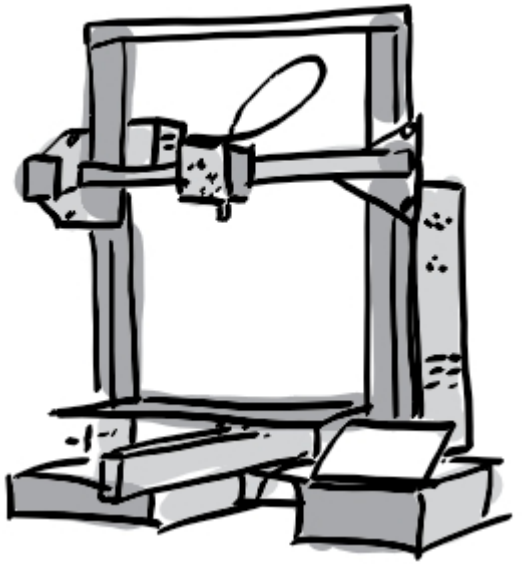
Information cachée : D

But: minimiser vos dépenses

Algorithme = acheter au jour T.

Coût(T,D) = $\begin{cases} D & \text{si } D < T \text{ et} \\ T\text{-}1+B & \text{sinon} \end{cases}$

Ratio de compétitivité = coût(T,D) / min(D,B)



Ratio déterministe = $2 - \frac{1}{B}$ $(T = B)$

Ratio randomisé = (solution à un programme linéaire)

$$1 + \frac{1}{\left(1+\frac{1}{B}\right)^{B} - 1} \approx \frac{e}{e-1} \approx 1{,}58$$

Doubler, illustré par enchère en ligne

[Chrobak,Kenyon-Mathieu,2006]

Ce problème n'entre pas exactement dans le cadre des algorithmes en ligne.

Information cachée : seuil $u \geq 1$

But: faire une enchère qui soit au moins u

Algorithme = séquence x0, x1, …
Ratio de compétitivité =

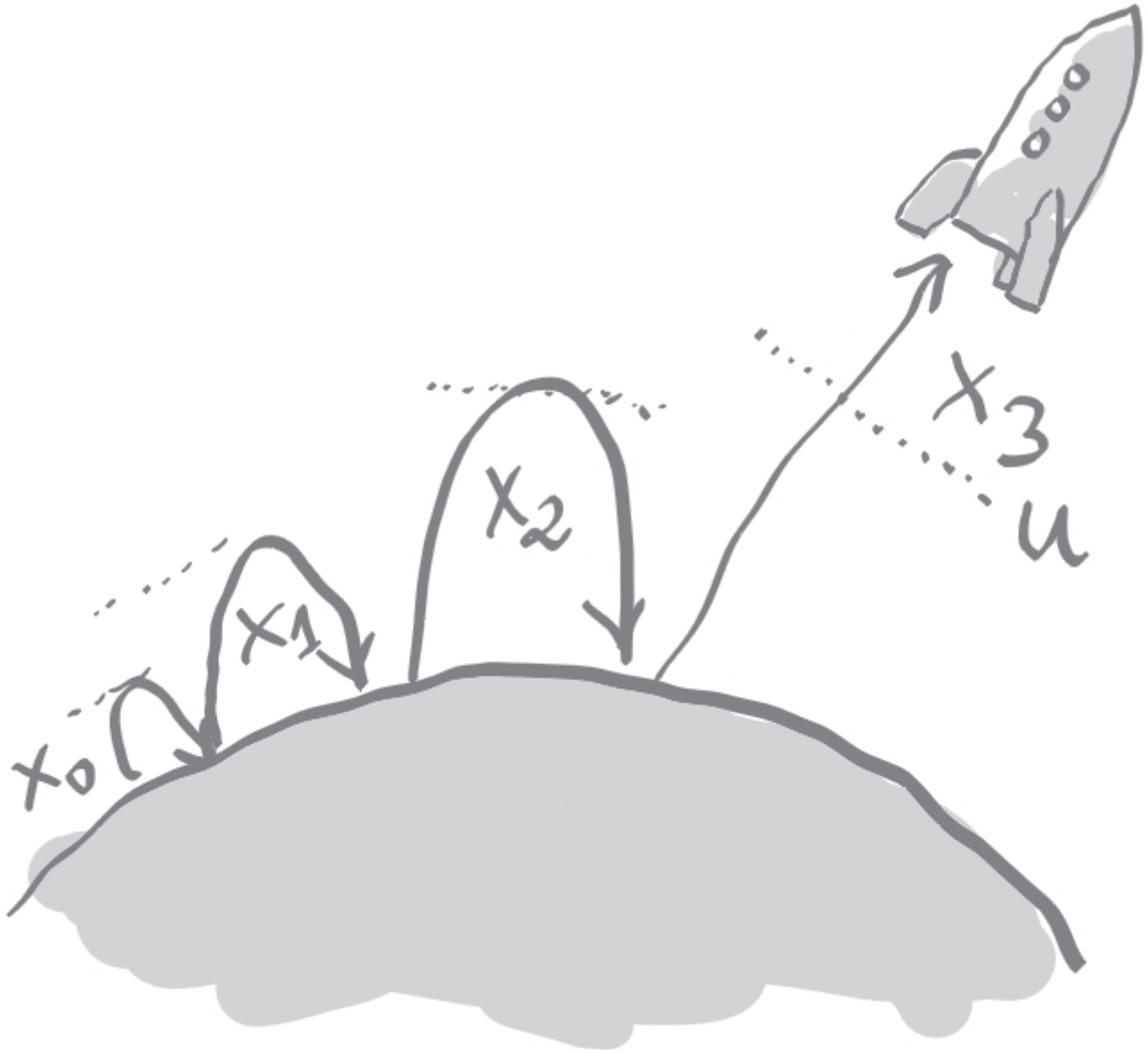
$$\frac{x_0 + x_1 + \ldots + x_k}{u}$$

pour le plus petit k tel que $x_k \geq u$

Ratio déterministe = 4 ($x_i = 2^i$)

Ratio randomisé = e ≈2,72 ($x_i = e^{z+i}$)

pour $0 \leq z < 1$ choisi uniformément au hasard



Compromis, illustré par routage de circuits virtuels

[Aspnes,et al,1993]

Donnée: graphe G=(V,E), capacités $u: E \to N$

Requête: $(s_i, t_i) \in V^2$

Servir: par un chemin de s_i à t_i, qui consomme une unité de capacité le long des arêtes.

(ceci est une des nombreuses approches)

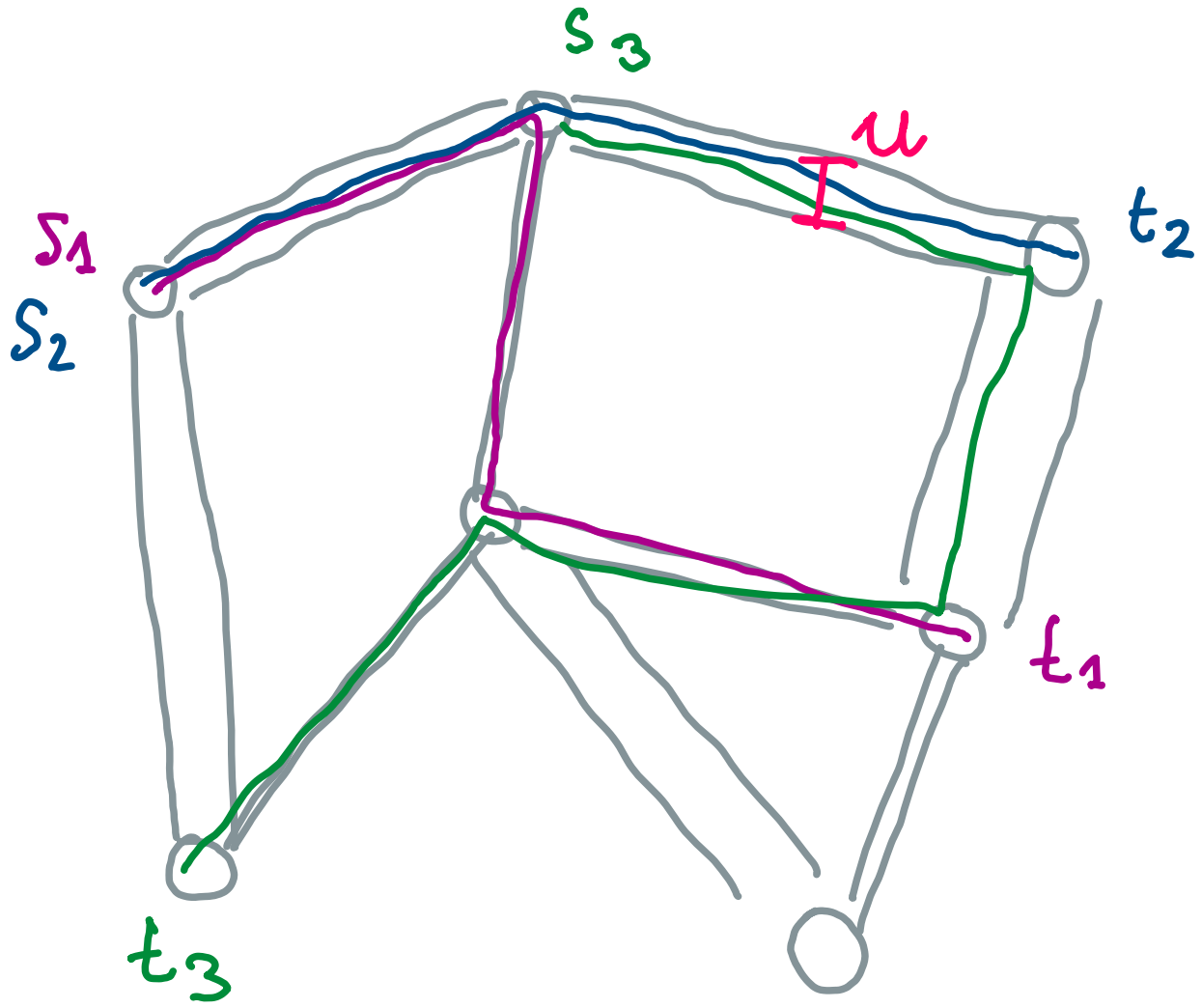
Congestion(e) = # chemins utilisant e / u(e)

But: minimiser congestion maximale

Le ratio du problème est Θ(log |V|)

Idée: router par plus court chemin avec longueur arête = exponentiel de congestion

Analyse: par fonction de potentiel



Approche primal-dual, illustrée par couverture par ensembles

Approche unifiée

Modéliser le problème par un programme linéaire

Contraintes arrivent en ligne.

Pour satisfaire une contrainte les variables peuvent être augmentées mais pas diminuées.

Permet de résoudre des variantes en modifiant le programme linéaire, et utiliser toute la machinerie : ajout de contraintes pour diminuer le gap d'intégralité, etc.

- Min $x_1+x_2+\ldots+x_n$

$x_1+x_2+\ldots+x_n \geq 1$ augmenter $x_1,\ldots,x_n$ à $1/n$

$x_2+\ldots+x_n \geq 1$ augmenter $x_2,\ldots,x_n$ à $1/(n-1)$

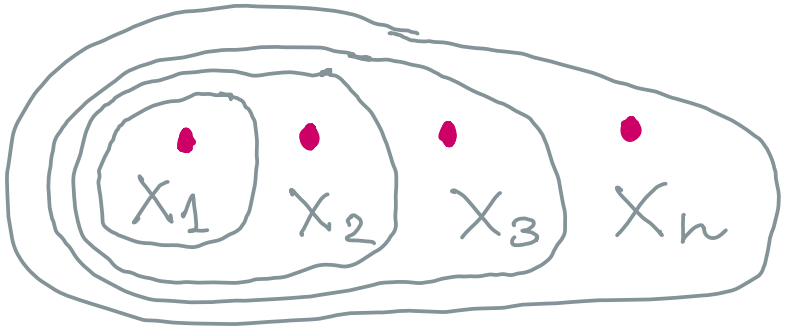
$\ldots$ $\ldots$

$x_n \geq 1$ augmenter x_n à 1

$x \geq 0$

Algorithme paie $1/n + 1/(n-1)+\ldots+1 = H_n \geq \ln(n)$

Optimum paie 1 ($x_n=1$)



Approche primal-dual, illustrée par acheter-ou-louer

[Buchbinder,Naor,2009]

Rappel: louer coûte 1 €/jour, acheter coûte B €,
le nombre total de jours D n'est pas connu d'avance

Algorithme pour une solution fractionnelle :

Initialement $x=z_i=y_i=0$
À l'arrivée de la i-ème contrainte:

Si elle n'est pas satisfaite ($x<1$):

$z_i = 1-x$
$x = x(1+1/B) + 1/(c\ B)$
$y_i=1$

Pour prouver que l'algorithme est (1+1/c)-compétitif

1. Primal est faisable
2. $\Delta P \leq (1 + 1/c)\Delta D$
3. Dual est faisable

C'est la partie technique du tutorial.
Servez vous un café.

- Primal = Min $Bx+\sum z_i$
  tel que $\forall i: x+z_i \geq 1$
  $x, z_i \geq 0$.
  $x=1$ indique l'achat, $z_i=1$ indique la location au jour i
  Contraintes arrivent en ligne.
  Variables peuvent seulement augmenter.
- Dual = Max $\sum y_i$
  tel que $\sum y_i \leq B$ et $\forall i: y_i \leq 1$
  $y_i \geq 0$.
  Variables arrivent en ligne.
  Variable peut augmenter seulement à son arrivée.

Approche primal-dual, illustrée par acheter-ou-louer

[Buchbinder,Naor,2009]

Rappel: louer coûte 1 €/jour, acheter coûte B €,
le nombre total de jours D n'est pas connu d'avance

Algorithme pour une solution fractionnelle :

Initialement $x=z_i=y_i=0$
À l'arrivée de la i-ème contrainte:

Si elle n'est pas satisfaite ($x<1$):

$z_i = 1-x$
$x = x(1+1/B) + 1/(c\ B)$
$y_i=1$

Pour prouver que l'algorithme est (1+1/c)-compétitif

1. Primal est faisable
2. $\Delta P \leq (1 + 1/c)\Delta D$
3. Dual est faisable

C'est la partie technique du tutorial.
Servez vous un café.

- Primal = Min $Bx+\sum z_i$
  tel que $\forall i: x+z_i \geq 1$
  $x, z_i \geq 0$.
  x=1 indique l'achat, z_i=1 indique la location au jour i
  Contraintes arrivent en ligne.
  Variables peuvent seulement augmenter.
- Dual = Max $\sum y_i$
  tel que $\sum y_i \leq B$ et $\forall i: y_i \leq 1$
  $y_i \geq 0$.
  Variables arrivent en ligne.
  Variable peut augmenter seulement à son arrivée.

$$B\Delta x + z_i \leq x + 1/c + 1 - x = 1 + 1/c$$

Approche primal-dual, illustrée par acheter-ou-louer

[Buchbinder,Naor,2009]

Rappel: louer coûte 1 €/jour, acheter coûte B €,
le nombre total de jours D n'est pas connu d'avance

Algorithme pour une solution fractionnelle :

Initialement $x=z_i=y_i=0$
À l'arrivée de la i-ème contrainte:

Si elle n'est pas satisfaite (x<1):

$z_i = 1-x$
$x = x(1+1/B) + 1/(c\ B)$
$y_i=1$

Pour prouver que l'algorithme est (1+1/c)-compétitif

1. Primal est faisable
2. $\Delta P \leq (1+1/c)\Delta D$
3. Dual est faisable

C'est la partie technique du tutorial.
Servez vous un café.

- Primal = Min $Bx+\sum z_i$
  tel que $\forall i: x+z_i \geq 1$
  $x, z_i \geq 0$.
  x=1 indique l'achat, z_i=1 indique la location au jour i
  Contraintes arrivent en ligne.
  Variables peuvent seulement augmenter.
- Dual = Max $\sum y_i$
  tel que $\sum y_i \leq B$ et $\forall i: y_i \leq 1$
  $y_i \geq 0$.
  Variables arrivent en ligne.

x_i = augmentation de x au jour i
(x_i) est une série géométrique de raison (1+1/B) et premier terme 1/(c B)

$$x_1+\ldots+x_{i-1} = \frac{1}{cB} \cdot \frac{\left(1+\frac{1}{B}\right)^{i-1}-1}{\left(1+\frac{1}{B}\right)-1} = \frac{\left(1+\frac{1}{B}\right)^{i}-1}{c}$$

Nous choisissons $c = \left(1+\frac{1}{B}\right)^{B} - 1 \approx e-1$

et obtenons le ratio $1+\frac{1}{c} \approx e/(e-1)$

Le problème avec le ratio de compétitivité

Différents modèles d'adversaire

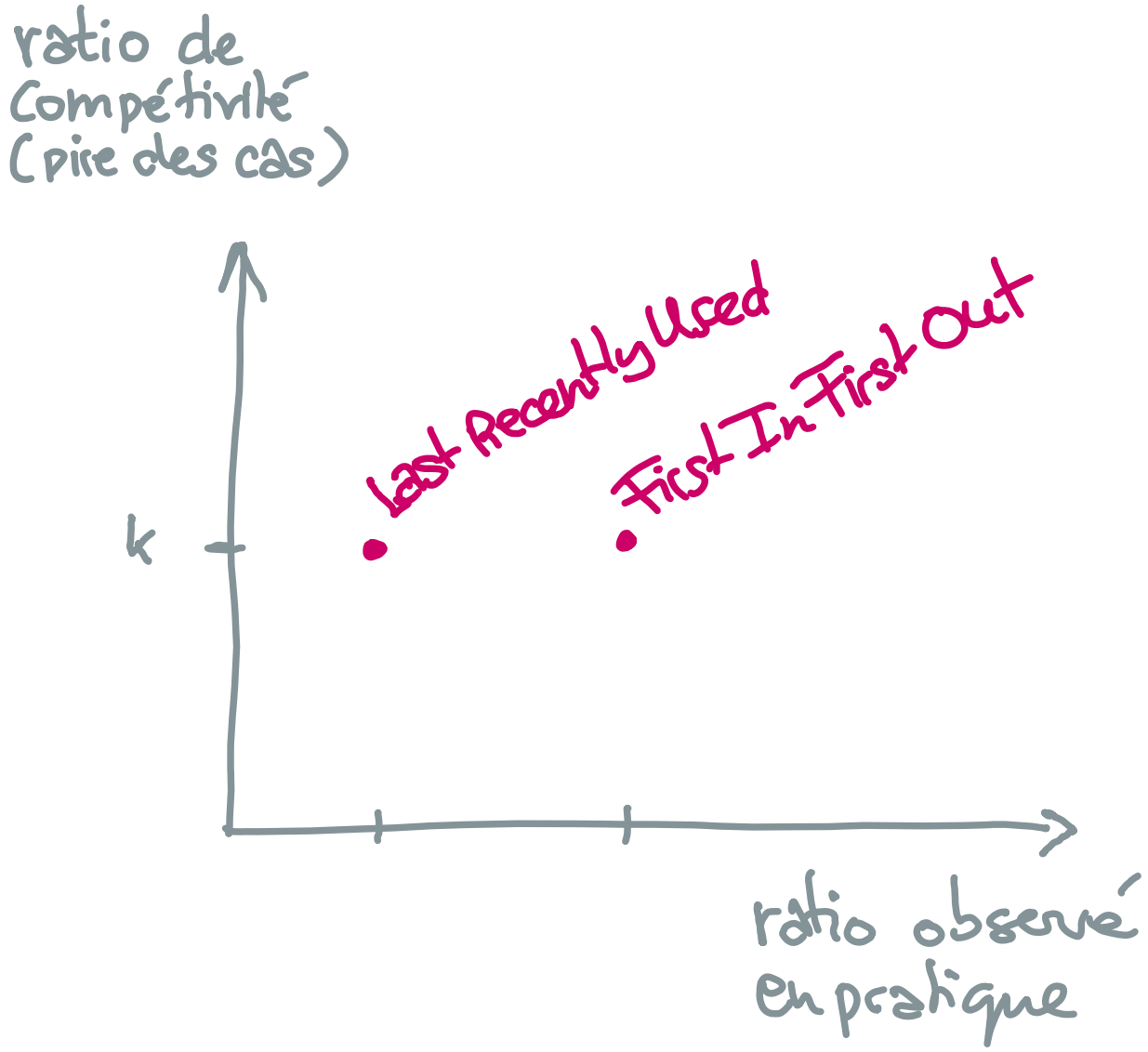
- Stochastique
- Ordre aléatoire
- Chaîne de Markov

Nécessite connaissance précise de l'entrée

Différents modèles d'algorithme

- Avec aperçu sur prochaines requêtes
- Avec conseil

Pas réaliste



Algorithmes en ligne avec prédictions

Différents modèles étudiés

Problème hors ligne : toute l'instance est connue

- Algorithme reçoit un conseil (fonction arbitraire de l'entrée complète)
  - Dépendance entre taille conseil et ratio de compétivité. Effets de seuil.
- Conseil peut être correct, ou complètement faux
  - Algorithmes avec paramètre de confiance
- Conseil avec erreur
  - Performance de l'algorithme devrait se dégrader gentiment avec l'erreur, pour avoir le meilleur de deux mondes

Expert
Apprentissage automatique
Promesse

Problème en ligne : aucune information sur le futur

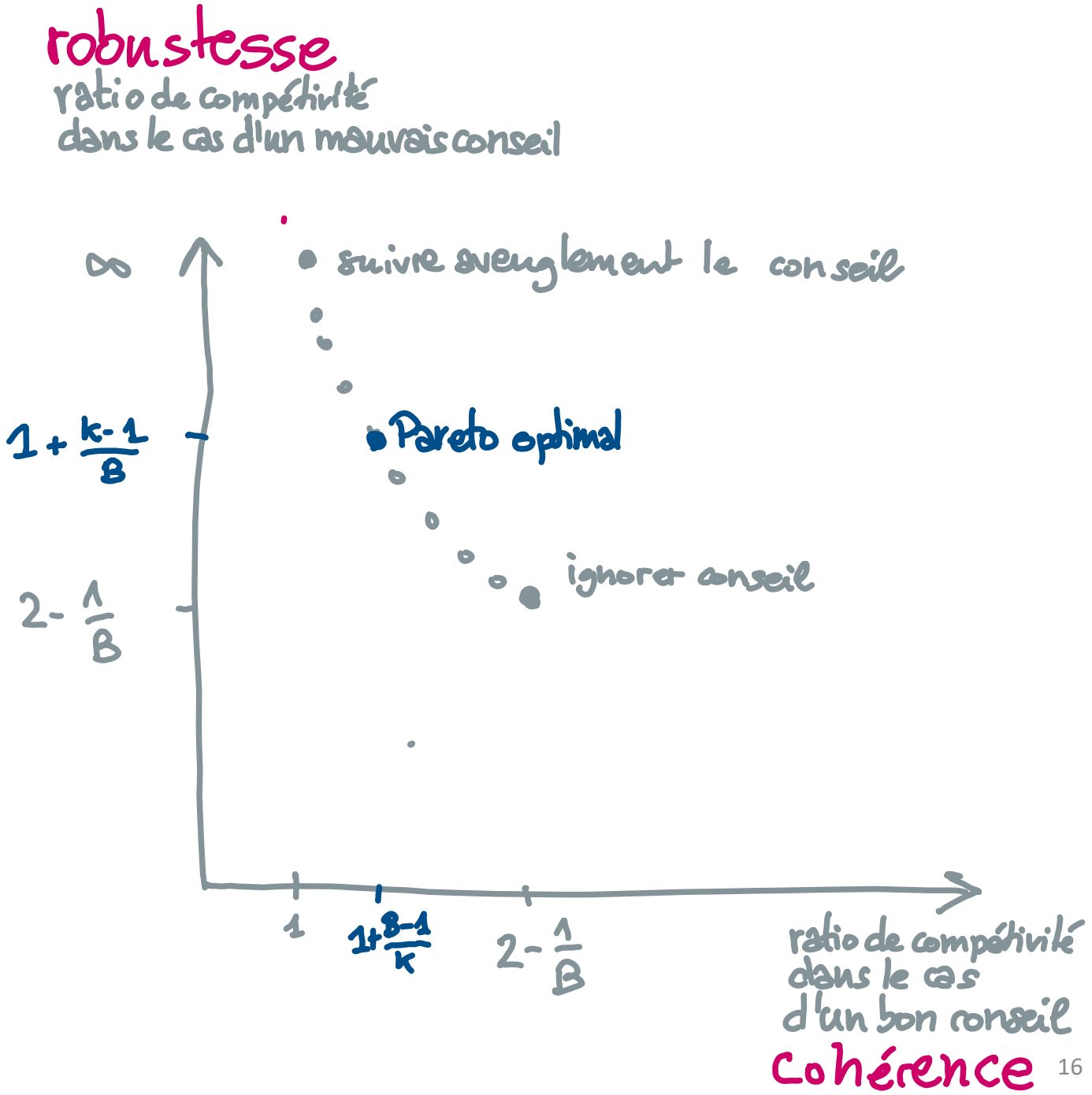
Acheter-ou-louer avec conseil sans confiance

[Purohit,Svitkina,Kumar,2018]
[Angelopoulos,D,Jin,Kamali,Renault,2020]

Rappel:
Louer coûte 1€/jour. Acheter coûte B €.
Inconnu: nombre total de jours D.

Conseil ∈ {acheter, louer}

Notre algorithme:
- conseil=louer? Acheter jour B
- conseil=acheter? Acheter jour k



Gestion de cache avec prédictions

[Lykouris,Vassilvitskii,2019]

K = taille du cache.

- Requête = page.
- Page dans le cache? Coût 0.
- Page pas dans le cache? Coût 1, doit remplacer une page du cache

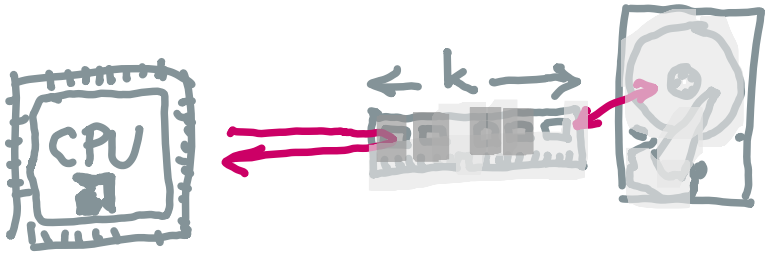
Algo hors ligne optimal [Bélády,1966]: remplacer la page du cache dont la prochaine requête est la plus lointaine

Algo de marquage (sans conseil) est O(log k)-compétitif [Fiat,et al, 1991] et c'est optimal : [Achlioptas,Chrobak,Noga,2000]

- page dans le cache? Marker la page
- page pas dans le cache? Remplacer une page non-marquée aléatoire
- toutes les pages marquées? Commencer une nouvelle phase sans marques

Conseil: pour chaque page, date prochaine requête. ϵ=Erreur L1

En pratique : *Last Recently Used [LRU]* est un bon conseil



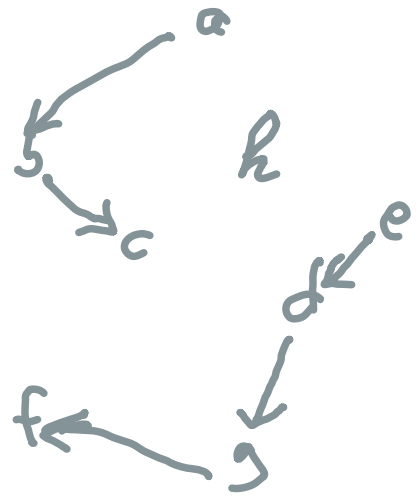
Définition Elément propre: qui arrive dans cette phase, mais pas dans la phase précédente

- page dans le cache? Marker la page
- page x pas dans le cache? Remplacer une page non-marquée y:
  - Si la chaîne de x a une longueur ≤ log(k): x est spécifié par le conseil
    Sinon x est page non-marquée aléatoire
- toutes les pages marquées? Commencer une nouvelle phase sans marques

Ratio de compétitivité = $\min(2 + 2\sqrt{5\epsilon}\,,\, 4\log k\,)$

Graphe du blâme:

x→y: page *x* a été remplacée par *y* (sous cond.)



Conditionnement en boîtes (bin packing) avec prédictions

[Angelopoulos,Kamali,Shadkami,2021]

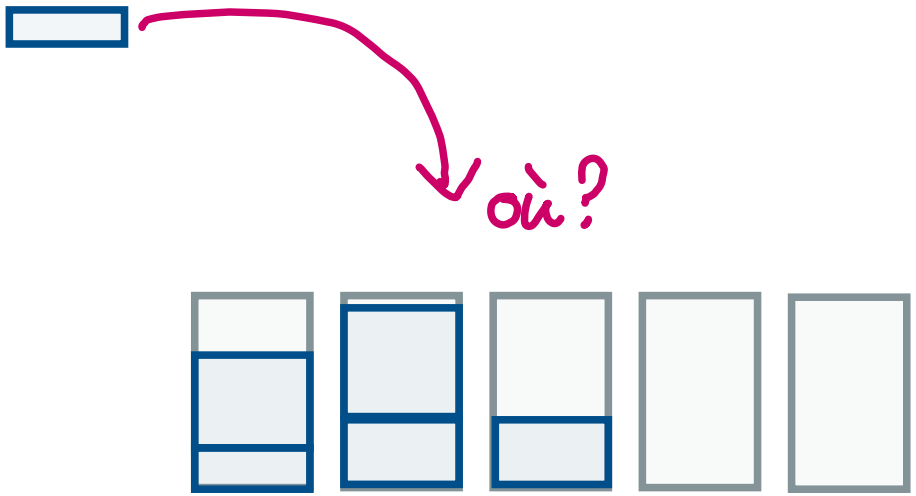
Entrée: des elements de taille $\in \{1, .., k\}$

Servir: les placer dans un nombre minimum de boîtes de capacité k

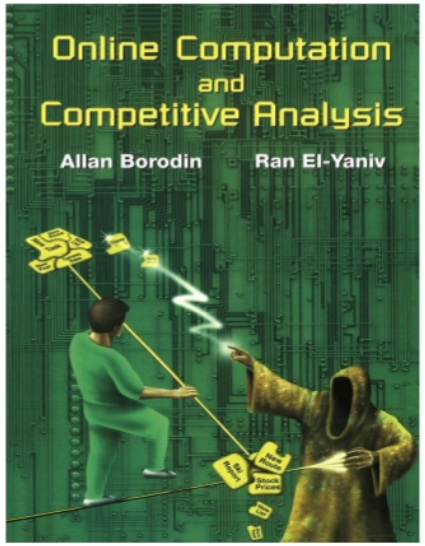
Conseil: fréquence pour chaque taille, avec erreur L_1 η

Paramètre: confiance λ

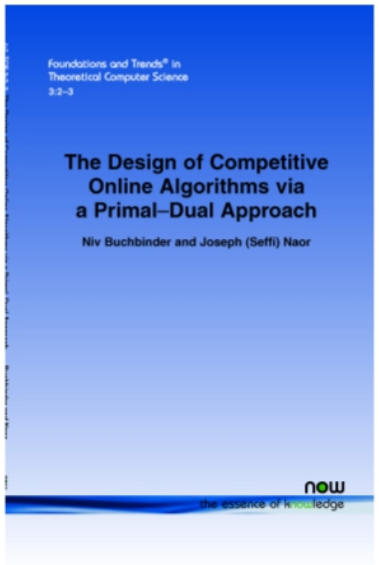
Algorithme: servir une proportion λ en utilisant un conditionnement optimal ($1 + \epsilon$ approximé) en function du conseil
et une proportion $1 - \lambda$ en utilisant un algorithme A ignorant le conseil (de ratio C_A)
Ratio de compétitivité:
$(1 + \varepsilon)\big((1 + (2 + 5\varepsilon)\eta k + \varepsilon)\lambda + C_A(1 - \lambda)\big)$



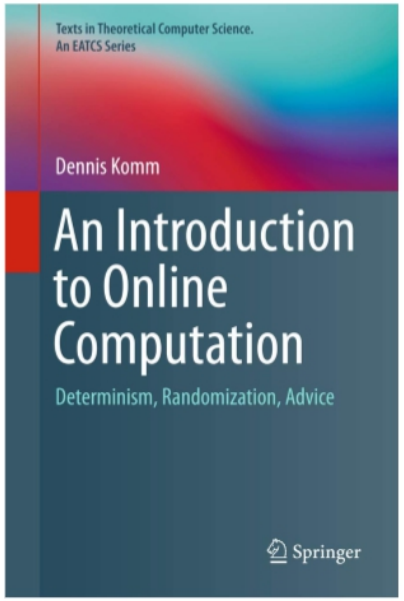
Pour en savoir plus



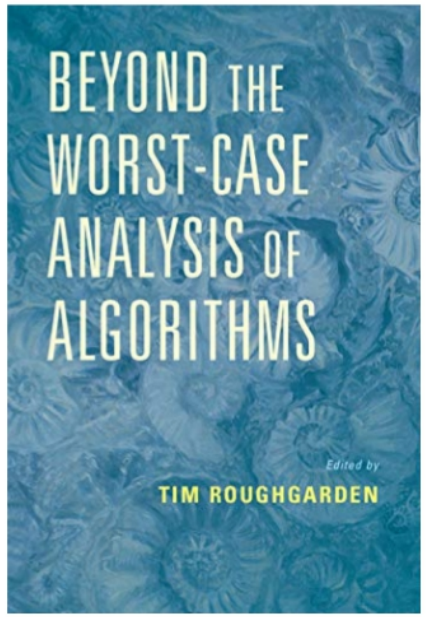
1998



2009



2016



2020