

PROG. DYNAMIQUE STOCHASTIQUE

- ◆ Le principe
- ◆ Les maths et l'info
- ◆ Tout ce qu'on peut
mettre dedans
- ◆ Les applis



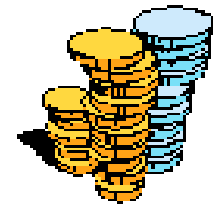
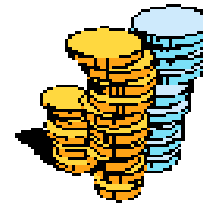
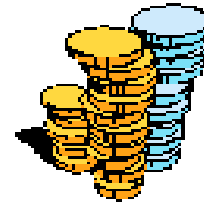
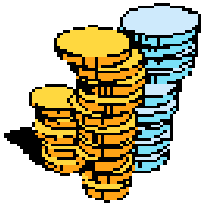
Remerciements :
Sylvain Gelly, Jérémie Mary, Artélyls.

Le principe

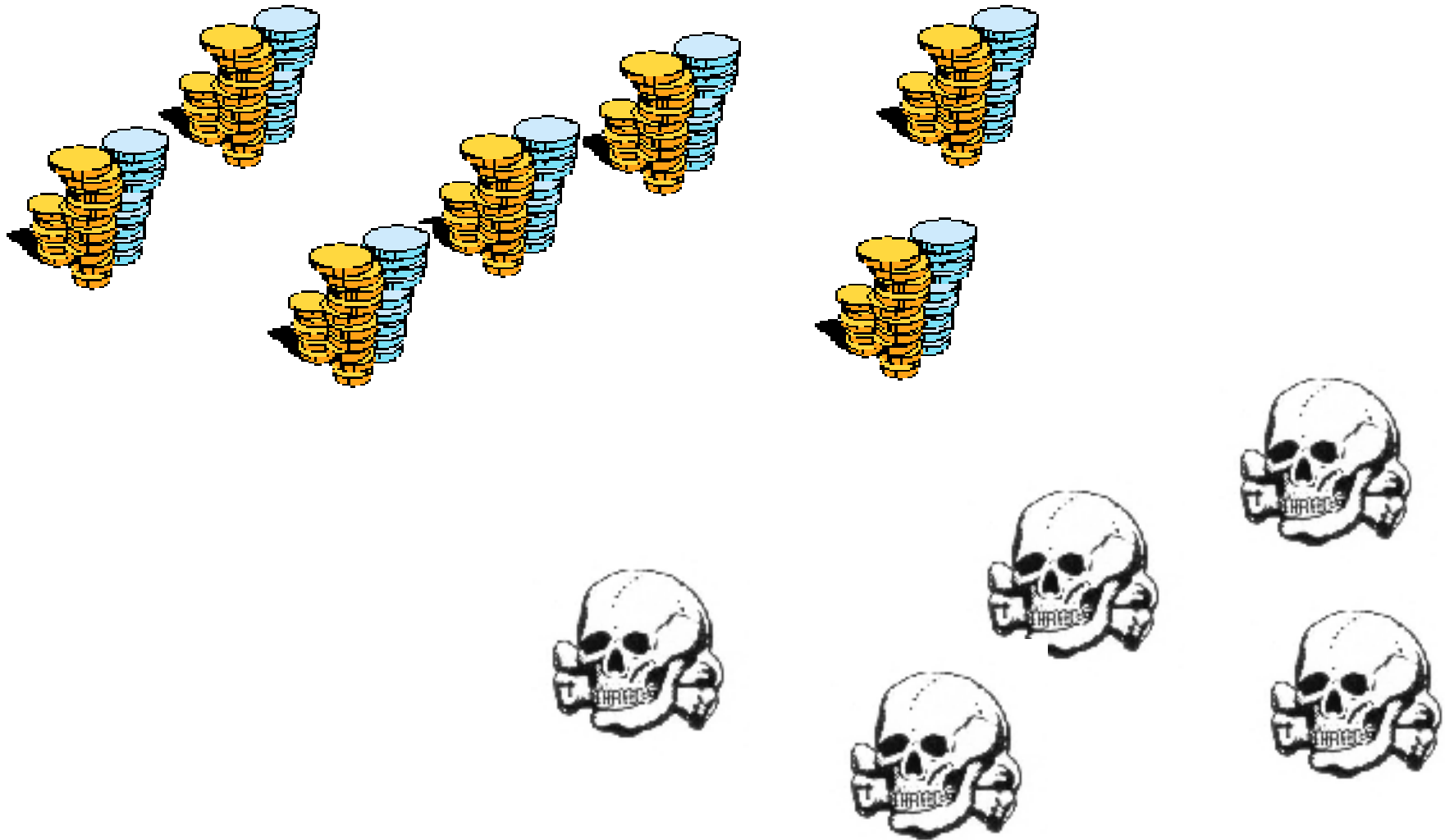


Imaginez que vous vouliez gagner un gros tas de pognon pendant votre mois d'août

Il y a des endroits avec pognon,
et des endroits sans pognons.



et le pognon bouge !



Et les mouvements sont plus ou moins coûteux !



Si on vise toujours la où y'a du
pognon ...

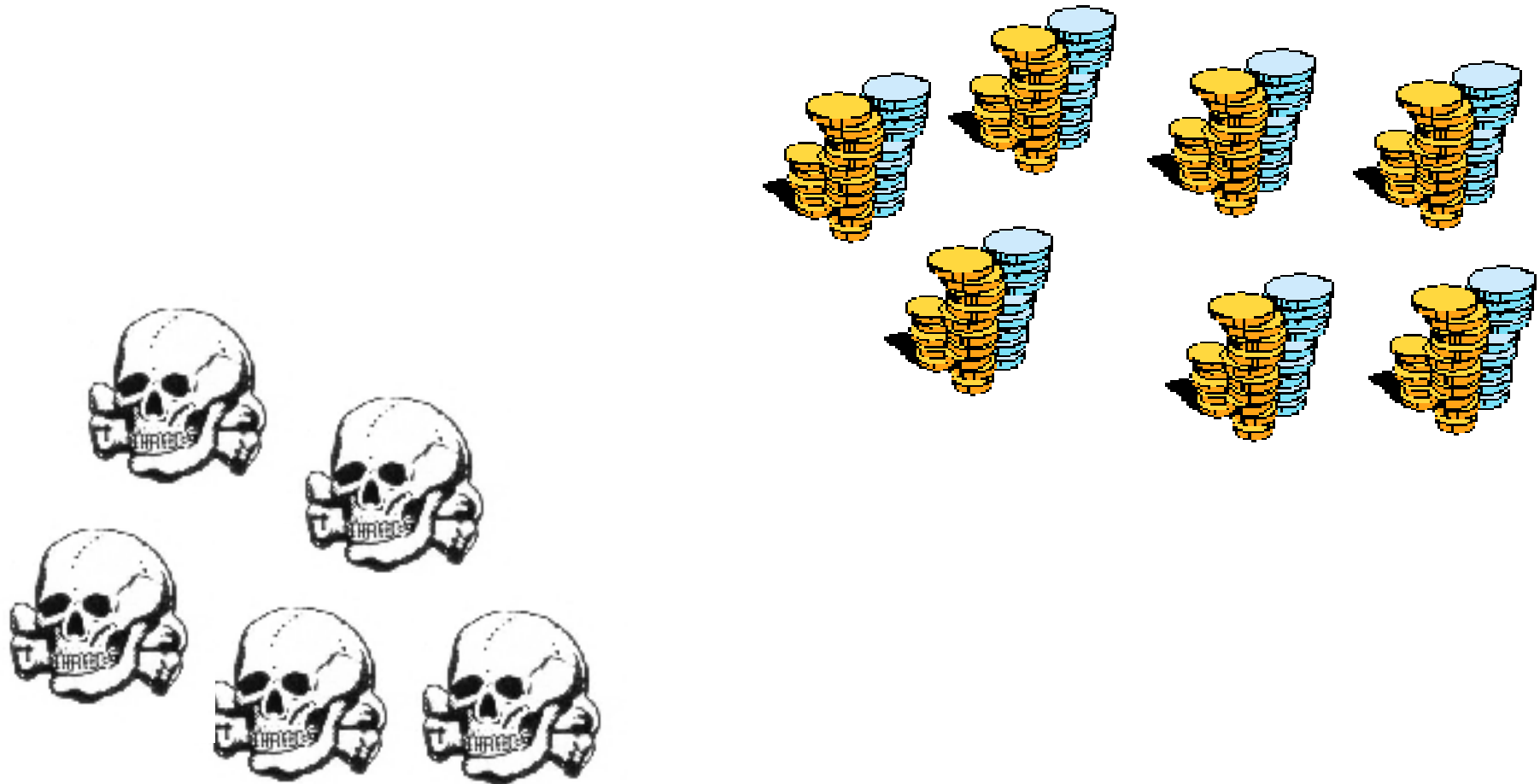


... c'est la cata !

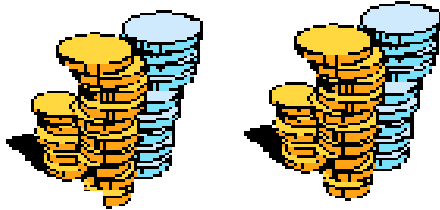


Il faut regarder le pognon tout de suite

...



... et le pognon plus tard.



...et optimiser la somme des deux !



...et optimiser la somme des deux !



Le problème est donc :

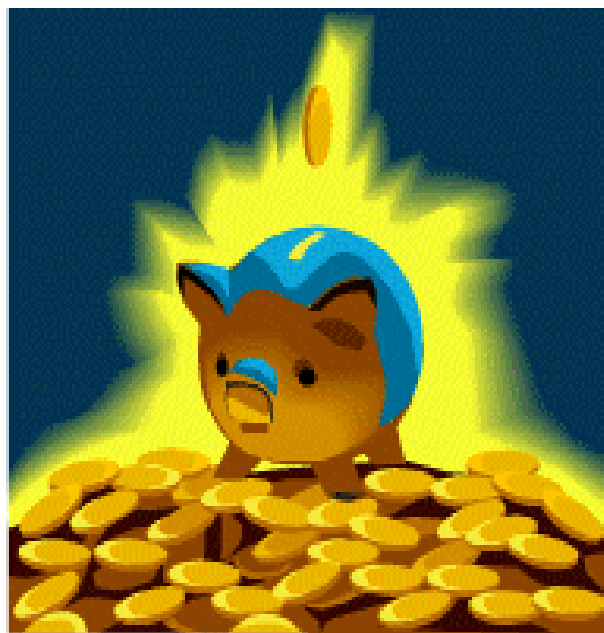
Je sais aujourd'hui quel endroit fournit quel pognon,



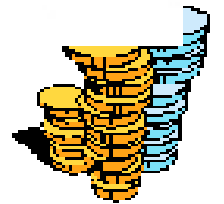
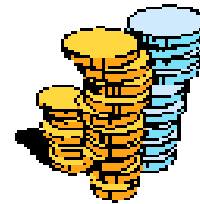
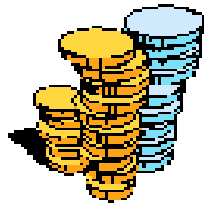
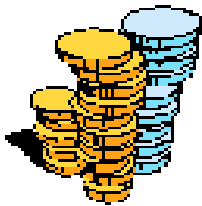
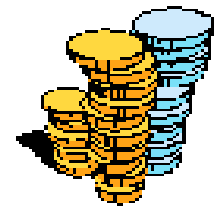
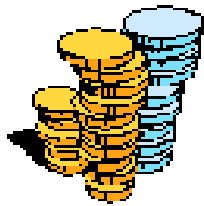
... mais comment savoir quel endroit fournira du pognon
au total du futur ?

\implies si je le sais j'ai plus qu'à faire la somme et optimiser !

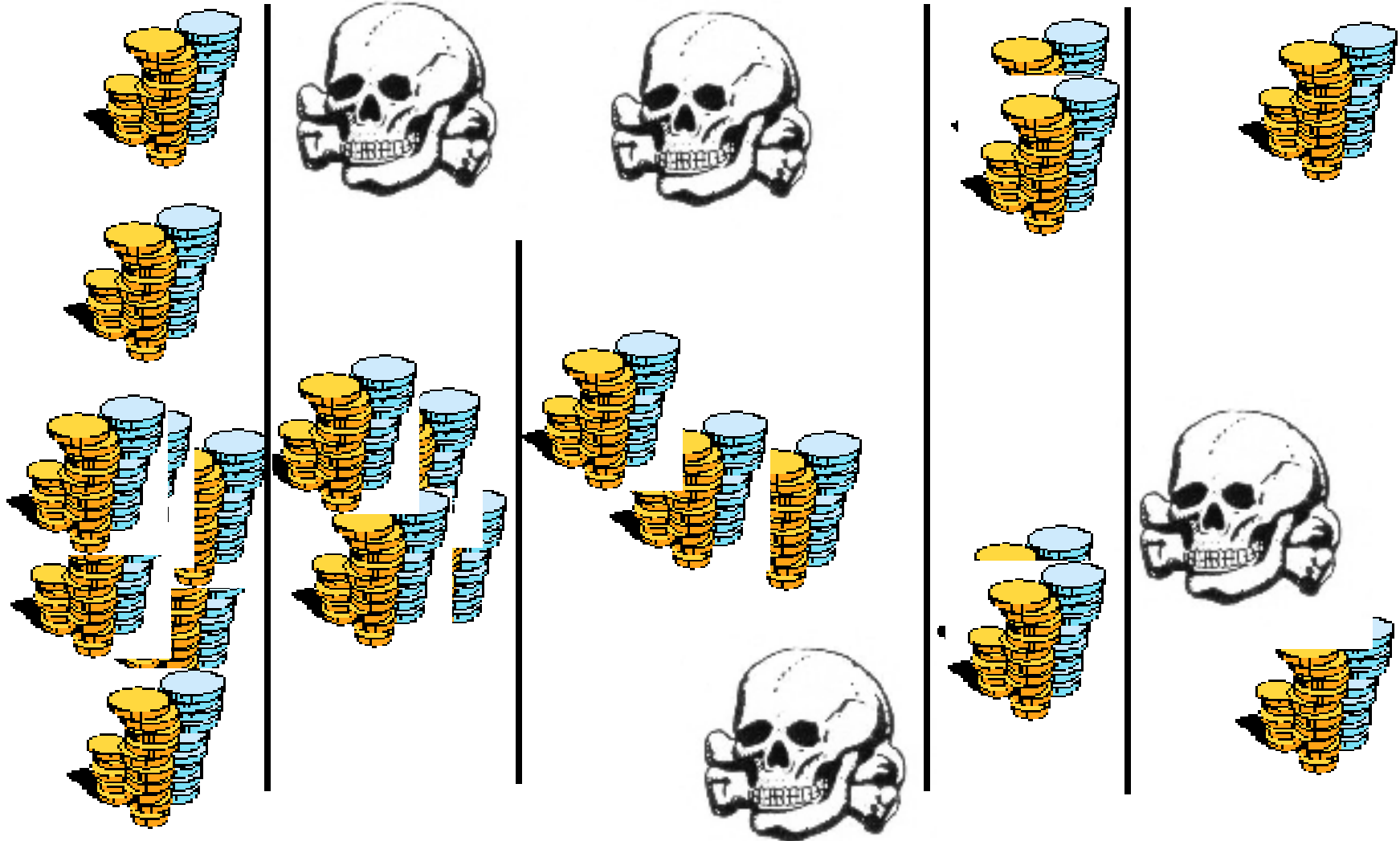
Et en prime le futur est stochastique :
le futur est la moyenne des futurs
possibles !



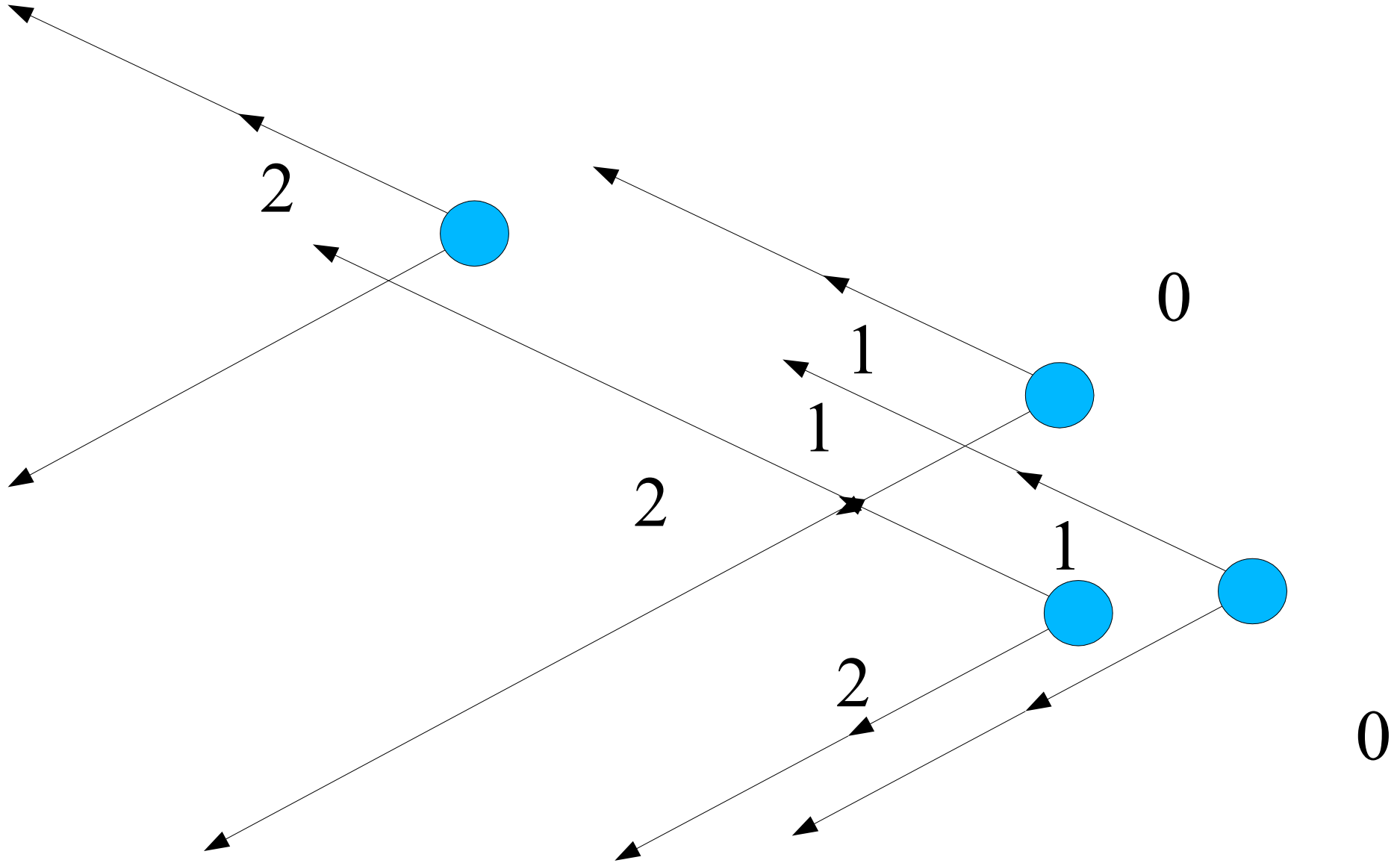
==> la programmation dynamique est une façon d'estimer le bénéfice futur



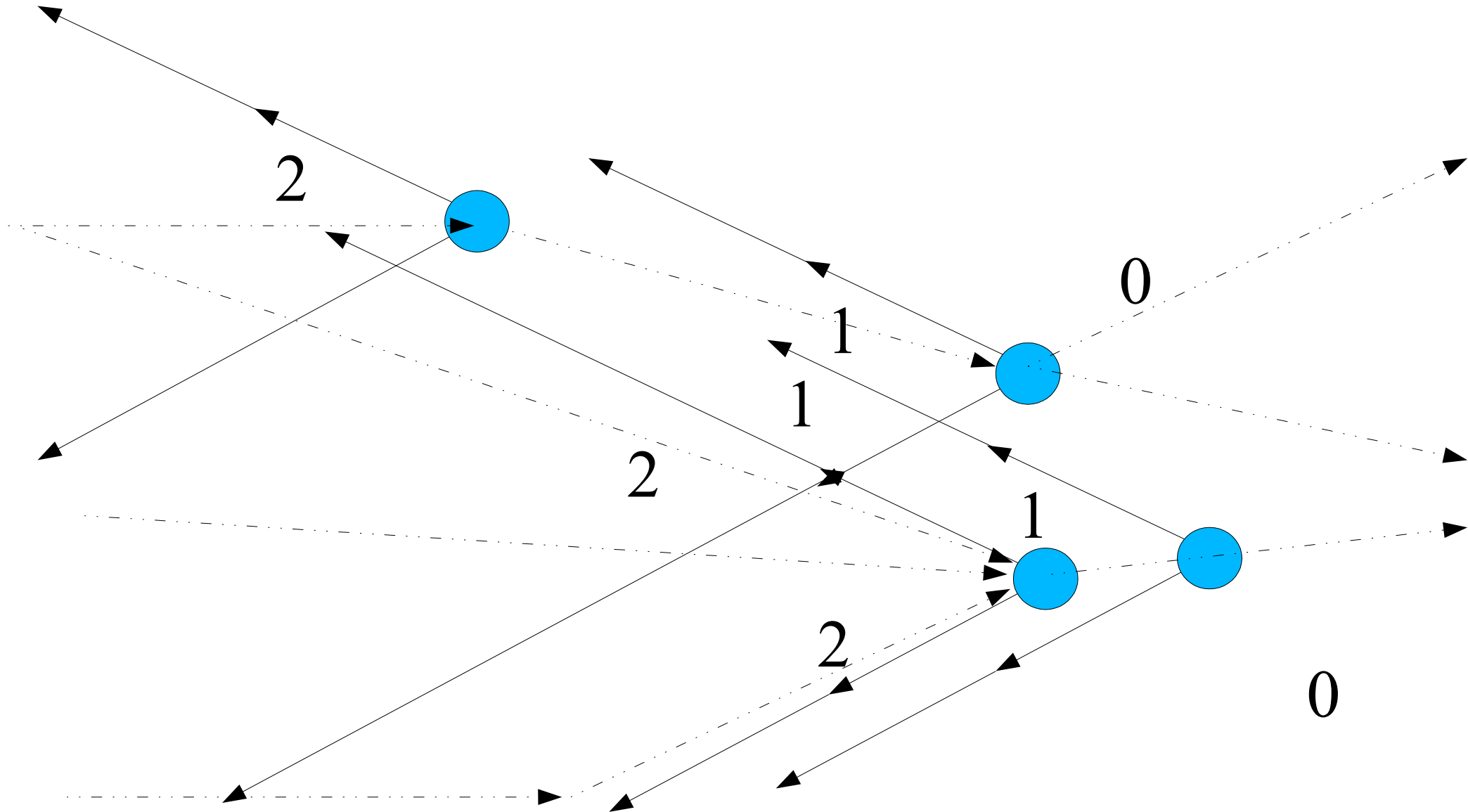
==> la programmation dynamique est une (tps-discrétisation)-sommation !



En gros, on somme ...



... et on suit la pente !



Le concept de valeur de Bellman

$V_{BEL}(t,x)$ = pognon gagné jusqu'à la fin des temps
depuis la position x à la date t

*(si x =niveau de stock,
dérivée par rapport à x : coût marginal)*

On « navigue » sur les valeurs de Bellman

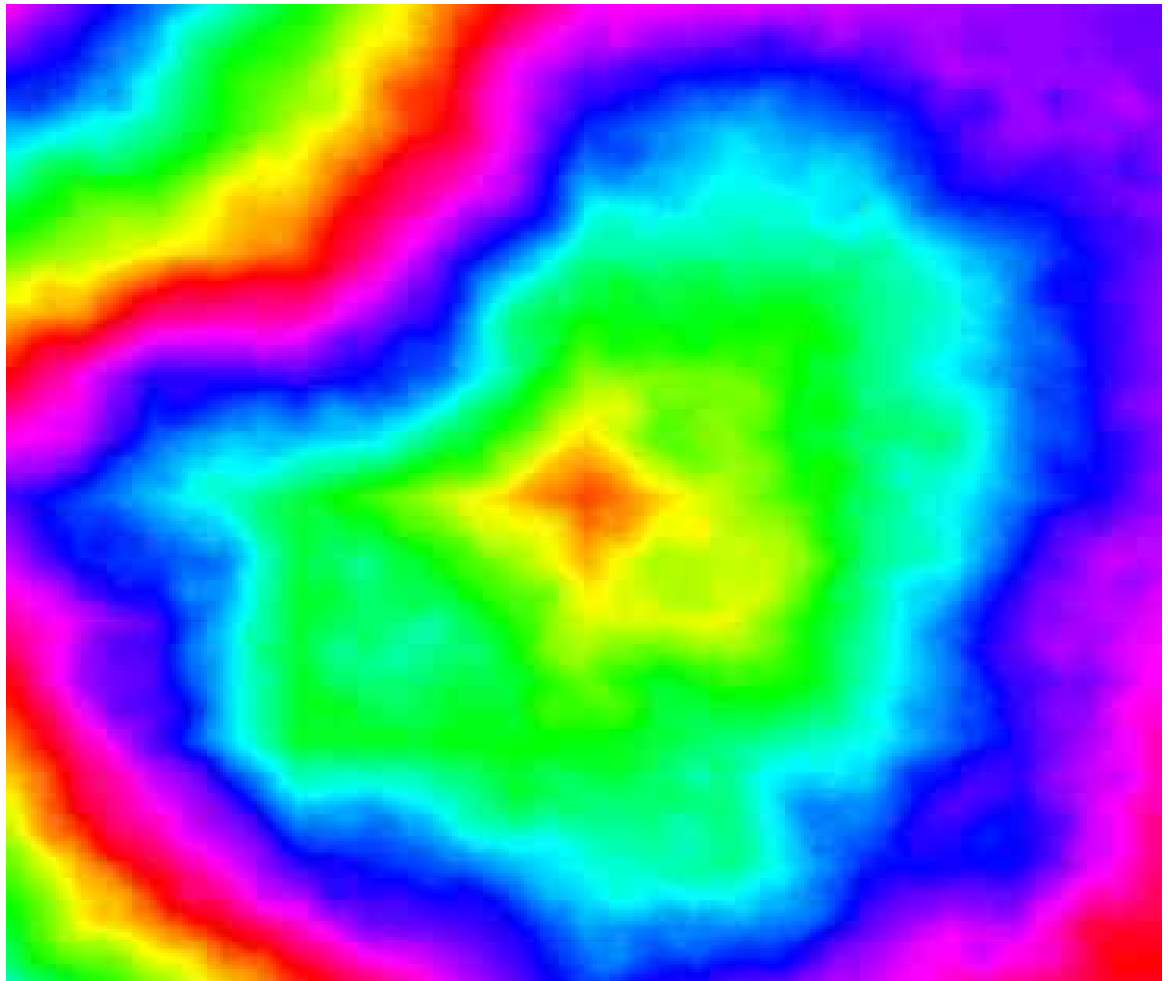


= somme

bénéfice instantané

+

bénéfice futur moyen



Un poil de maths

(Hamilton-Jacobi-Bellman)

Décision $d(\text{situation}) = \operatorname{argmin} C(d, \text{situation})$

$C(d, \text{situation}) =$

$\text{coûtFutur}(d, \text{stratégieFuture}) + \text{coûtInstantané}(d)$

où $\text{coûtFutur} = \text{Bellman}(\text{position}(\text{situation}, d))$

◆ décomposition lagrangienne ;

◆ rétropropagation ;

◆ chaque valeur de Bellman = une optimisation

stochastique (un déplacement instantané).

Ca a l'air classe !

On rétropropage
en optimisant
toutes les décisions,
et on a la stratégie
optimale : toujours optimiser
la somme $V_{bel} + \text{bénéf}$!



Un peu de volumétrie



- ◆ $T \approx 2$ à 7 à 50 ou 5000 pas de temps ;
- ◆ $\approx 3, 7$ ou 15 transitions stochastiques
- ◆ $D =$ dimension de la nappe $\approx 1, 10, 100, 100\ 000$;
- ◆ $M \approx 1$ à 3 ou 15 ou 80 ou 2^{300} valeurs instantanées différents des aléas

$\Rightarrow T \times M = 75.000$ approxs
 $\Rightarrow T$ (ou 1) apps ?


$\Rightarrow T \times M \times \ll 2^D \gg$
 $= 75.000.000$ opts
 $\approx TM2D^2 = 15\ 000\ 000$ opts
 $\Rightarrow T \times SC(D)$ opts ?



Au fait, et l'informatique ?




Au fait, et l'informatique ?

- 
- Y'a toujours des bugs
 - On y voit rien
 - C'est lourdingue à manier
 - Dès qu'on change un truc
faut tout recoder

Et puis, les performances ?



Et puis, les performances ?

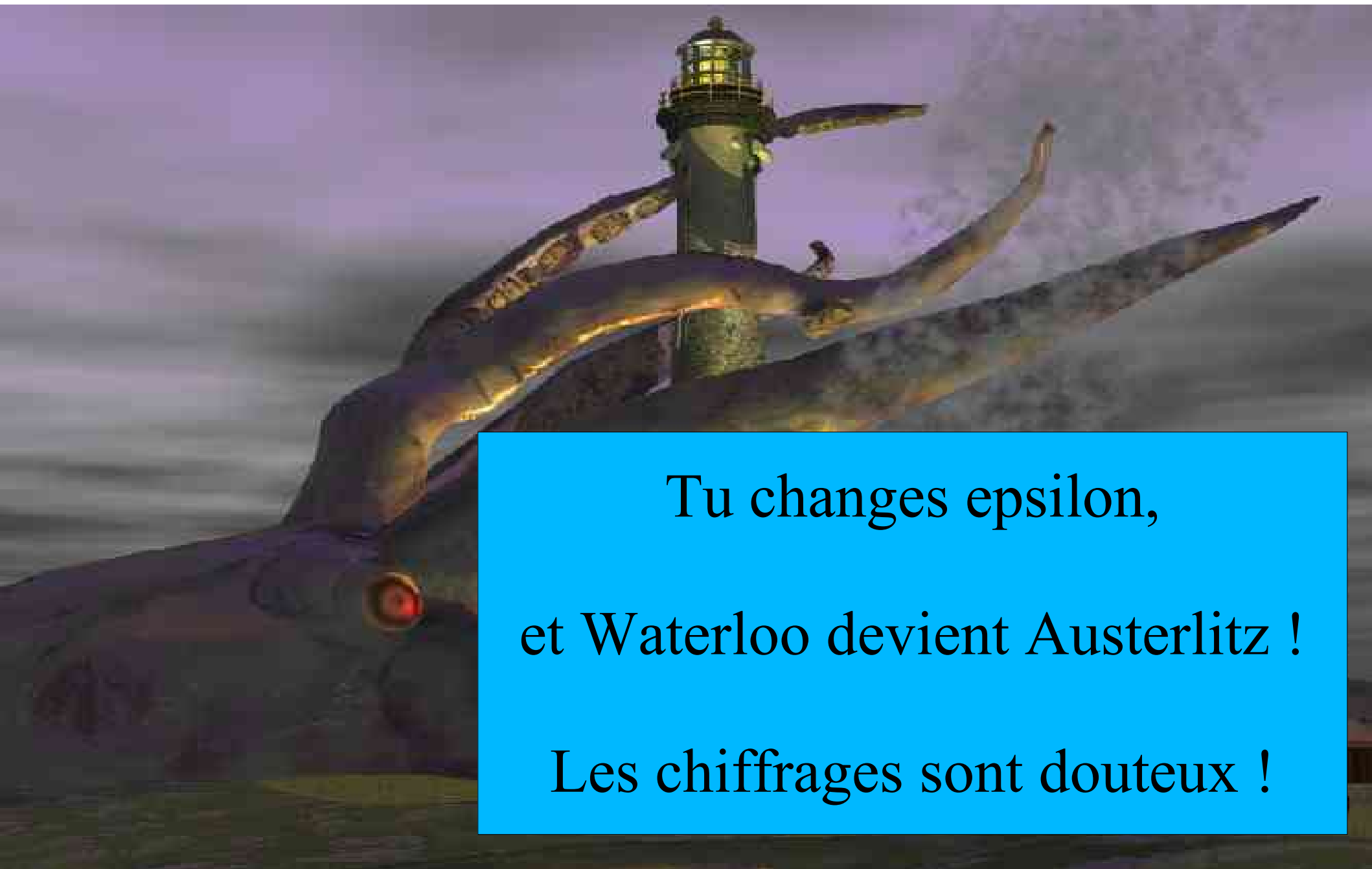


Ok, c'est stable,
mais alors ça rame,
mais alors à un point !

Et puis, la représentation des aléas ?



Et puis, la représentation des aléas ?

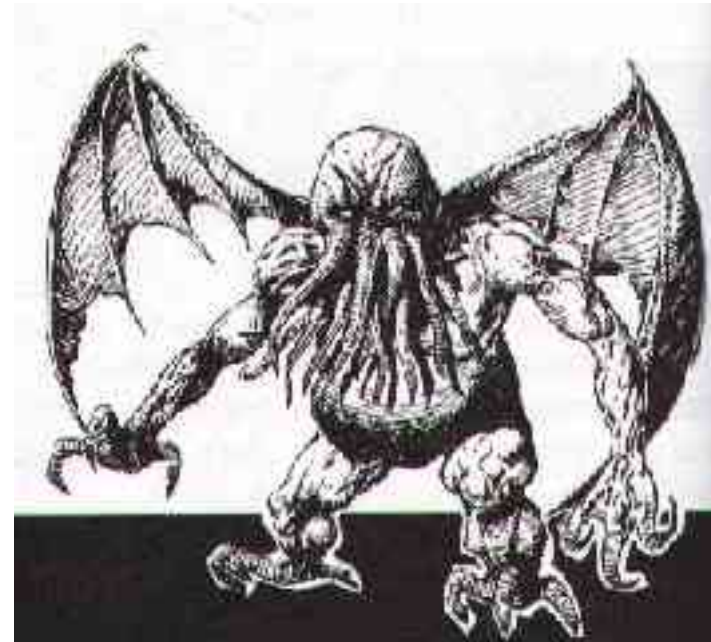


Tu changes epsilon,
et Waterloo devient Austerlitz !
Les chiffrages sont douteux !

Les embêtements

La vie du dynamic-programmeur :

- parce qu'on reprogramme tout à chaque fois :*
- on valide pas toujours très bien ;*
- c'est pas visualisable ;*
- la rep. des aléas est archi-instable*
- y'a des bugs partout ;*
- ça rame grave ;*
- y'a des rustines qui sentent pas bon ;*
- les aléas sont mal modélisés ;*
- mais on a pas mieux*
(RL rame hyper-grave !)





Bilan



Très grande précision

Plus robuste que l'AR (décomposition et non pt fixe)

Grande dimension bricolatoire

Aléas tordus bricolatoires

Manque de codes génériques

==> mettre de l'apprentissage au lieu de l'approx

==> garder la décomposition

==> réfléchir aux aléas et à la validation statistique



Applis



Les classiques :

- ressources en énergie
- réduction de pollution
- irrigation
- ~~finance~~
- chiffreages stratégiques
- influences climatiques
- influences sociétales
- influences d'incertitudes

Les moins classiques :

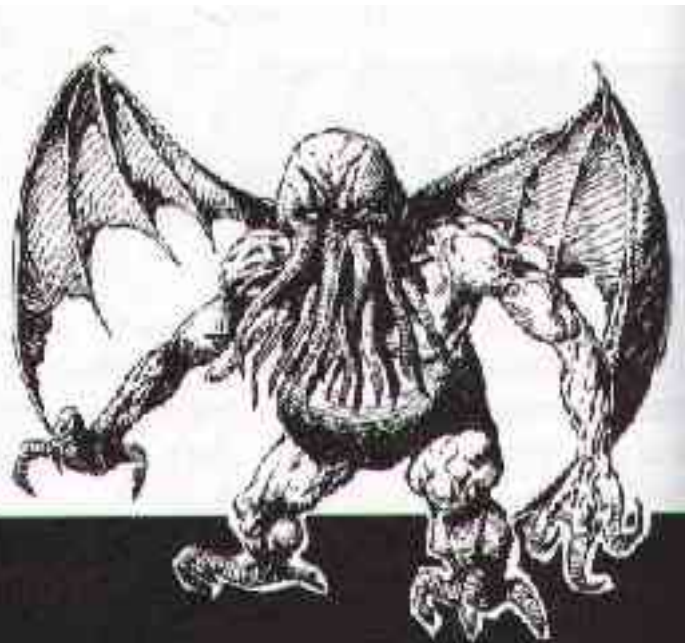
- contrôle-commande
- robotique
- lâchons-nous :
 - médecine
 - agriculture
 - trafic routier

Fournitures

- 1) une plateforme d'expérimentations
(gros code --> plein de manip d'IA)
- 2) plein de solveurs
(on est au bon endroit pour ça)
- 3) des réponses techniques
Markov caché, les rustines grande dim,
combien de courbes types, opt sans
gradient, algorithmique
- 4) des chiffres, des solutions (==> écolos)
(very user-friendly, génie logiciel ok)

Comment blaster les Chtulus

- 1) les performances
- 2) l'instabilité vis-à-vis des aléas
- 3) la propreté anti-bugs



Apprentissage

(pas besoin de déterminer !)

Apprentissage grande dimension

Apprentissage actif (==> thèse Jérémie)

Apprentissage L-infini

Apprentissage « hétérogène »

« en-ligne non-destructif »



Optimisation

~~Différentiation automatique~~

Différentiation sans gradient

Démarrages à chaud (par app. ?)

Comparaisons BFGS + RS + diff. finies / algos G.

Non-acharnement

MRS --> optimisation

Paramétrisation

Contraintes couplantes



Modélisation d'aléas

Markovien non-observable (\implies thèse Sylvain)

Classification (CAH) de courbes (\implies JM Loubes)

(contre les deux Ctulus)

(In-)Certitudes de décideur



Algorithmique & informatique

Tests unitaires & fonctionnels à donf

Programmation « 4 mains »

Profiler à fond

Optimiser partout, tester, comparer

Faire du réutilisable propre et beau





*Y'a plein d'apprentissage,
y'a plein d'optimisation dont sans dérivées,
y'a plein de modélisation d'aléas,
y'a plein d'informatique,*

*y'a un enjeu écologique et économique énorme,
y'a une littérature rikiki,*

*c'est dans **nos compétences**
et dans celles de nos copains matheux,*

*y'a à la fois de la recherche pizza-coca
et de la théorie brute*

MERCI !

