École des Ponts
ParisTech

ÉCOLE NATIONALE DES PONTS ET CHAUSSÉES

Rapport du projet de programmation dynamique

OPTIMISATION CONVEXE

Par

de MONICAULT Vianney

ESTEVE Erwann

Table des matières

N.B. : Le code est disponible sur GitHub à l'adresse https://github.com/ErwannE/OPTIM

Exercice 1 : Problème de maintenance

N.B. : Le code associé à cette exercice se trouve dans le document exo1.ipynb.

(a) Le problème peut-être modéliser par une chaine de Markov à horizon fini (car les machines sont changés tous les ans), la maximisation de l'espérance a donc un sens.

(b) Le problème est bien une chaine de Markov controlée, puisque l'état au temps $t + 1$ ne dépend que de l'état et de l'action choisie au temps t.

Les états sont "new", "in good shape", "old" ou "brocken", qui dans notre code sont respectivement noté 1, 2, 3, 4.

Les actions dépendent de l'état :

- Dans l'état "new", la seule action possible est "do nothing" (noté 1)
- Dans l'état "in good shape", les actions sont "do nothing", "maintain", "repair" et "replace" (noté 1, 2, 3, 4)
- Dans l'état "old, les actions sont "do nothing", "maintain", "repair" et "replace" (noté 1, 2, 3, 4)
- Dans l'état "brocken", les actions sont "do nothing", "maintain", "repair" et "replace" (noté 1, 3, 4)

La fonction *simulation* a été implémenté afin de pouvoir simuler un tour (à partir d'un état et d'une action, elle renvoie un nouvel état en respectant les probabilités indiquées par l'énoncé).

(c) La valeure optimale trouvée par programmation dynamique est 110.82, sa politique associée est trouvable sur le notebook.

Exercice 2 : Gestion de stock

On décrit le problème de la manière suivante :
Etat : Stock au soir du jour t, s_t
Contrôle : Commande de nouveux produits (de 0 à 5), a_t
Coût :

- Command : 1
- Stock : 0.1
- Sales : -3

$c_t = 0,1s_t + a_t - 3d_t$
La demande est d_t qui suit une loi binomiale dont les paramêtres sont définis pour les 14 jours.
La relation dynamique de l'état est la suivante :
$s_{t+1} = s_t + a_t - d_t$

L'équation de Bellman que nous proposons pour le problème :
$V_t(s_t) = \underset{at}{max}\, \mathbf{E}[3d_t - a_t - 0,1(s_t + a_t - d_t) + V_{t+1}(s_{t+1})]$

Une politique sera donc le choix du nombre d'unités à acheter pour le lendemain en fonction du stock du soir.
On se donne pour première politique, "naïve" de maintenir le stock à 10. Le gain associé est à peu près de 93.

```
(93.6542999999999, [92.82008058334027, 94.48851941665954])
```

FIGURE 1 – Maintenir le stock à 10 chaque matin

On décide donc de résoudre le problème par programmation dynamique. La stratégie adoptée est la suivante :

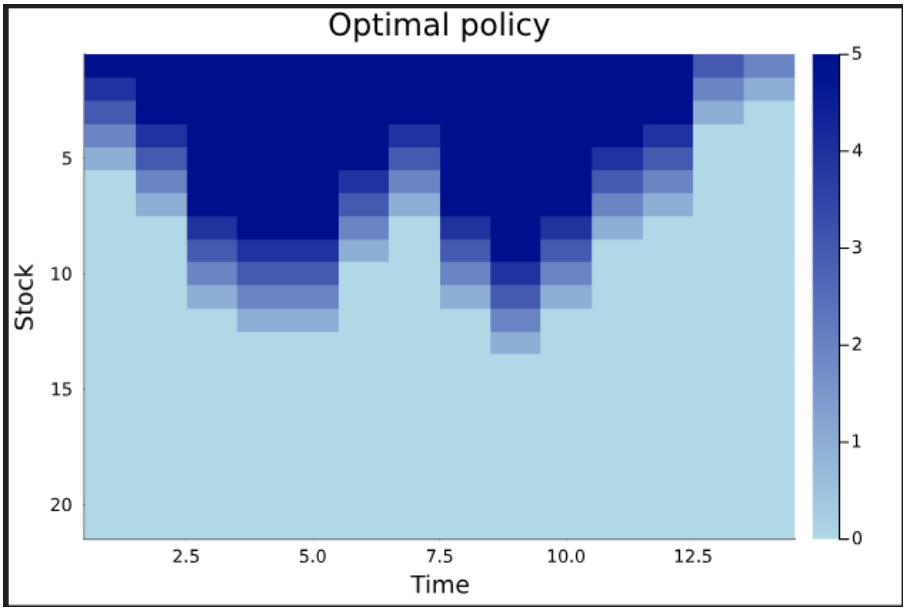

FIGURE 2 – La stratégie optimale

On utilise exactement le même raisonnement pour une demande périodique étalée sur 96 jours.
Le résultat : On observe que de même, la stratégie de commande est périodique, bien qu'elle

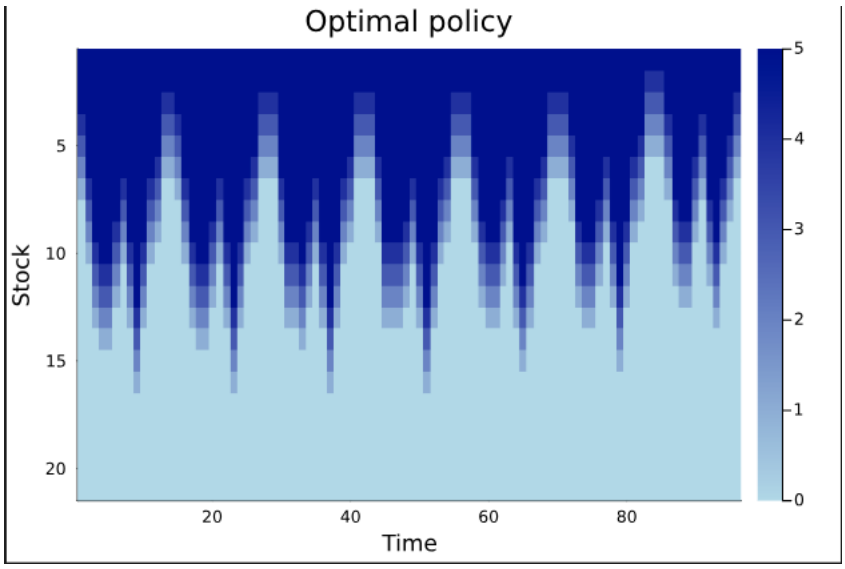

FIGURE 3 – La stratégie optimale sur 96 jours

ne soit pas exactement le même motif répété 7 fois.

Pour la dernière partie, la stratégie est moins facile et intuitive à appréhender. En effet, la commande n'arrive que 2 jours après être passée. La nouvelle équation de Bellman est :

$V_t(s_t) = \max_{a_t, at-1} \mathbf{E}[3d_t - a_t - 0,1(s_t + a_{t-1} - d_t) + V_{t+1}(s_{t+1})]$

avec $s_{t+1} = s_t + a_{t-1} - d_t$

et $V_1(s_1) = \max_{a_1} \mathbf{E}[3d_1 - a_1 - 0,1(s_1 - d_1) + V_2(s_2)]$

Dans les faits pour la programmation dynamique, on ne maximisera que sur une action, vu qu'on part du jour 14 et que pour le jour 14 une seule commande sera à prendre en compte (celle prise le jour 12). Cette commande sera fixée et la récurrence de la programmation dynamique fera qu'on ne s'intéressera qu'à une seule action à la fois (celle à prendre deux jours plus tôt). De la même manière que pour les deux exercices précédents, voici le résultat :

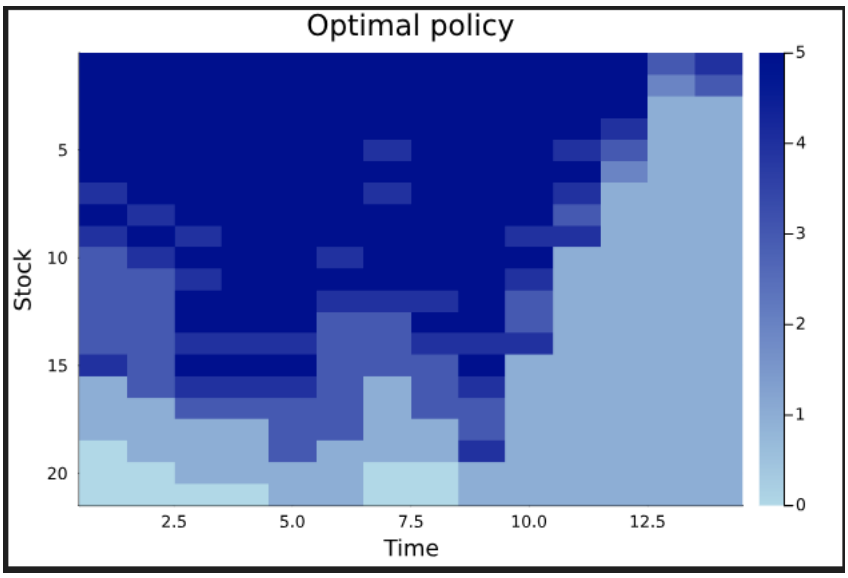

FIGURE 4 – Stratégie optimale

Exercice 3 : Jeu de dés

N.B. : Le code utilisé pour cette partie se trouve dans le document *exo3.ipynb*.

4. Jeu simple

Le jeu consiste en un jeu de lancer de dés sur 10 tours. Le joueur commence sans points et avec un dé. À chaque tour, s'il a plus de 6 points et moins de 3 dés, il peut payer 5 points pour gagner un nouveau dé. Puis la valeur maximale des dés jetés est ajoutée au total de points. Le but est de maximiser son nombre de points au bout des 10 tours.

(a) Ce problème peut être modéliser de la manière suivante.

États :

— Le nombre de points au début du tour t p_t
— Le nombre de dés au début du tour t d_t

Actions : $a_t \in \{0, 1\}$ variable indiquant si un dé a été acheté ou non.

Relations et bornes :

— $0 \leq p_t \leq 6(t-1), \quad \forall t \in [\![1, T]\!]$
— $0 \leq d_t \leq 3, \quad \forall t \in [\![1, T]\!]$
— $p_1 = 0, d_1 = 1$
— $d_{t+1} = d_t + a_{d+1}, \quad \forall t \in [\![1, T-1]\!]$
— $p_{t+1} = p_t - 5 * a_t + \max\limits_{i \in \{1, d_t\}} (u_i), \quad \forall t \in [\![1, T-1]\!]$ (u_i est le score du ième dé)

(b) Nous pouvons représenter une stratégie sur ce système via une matrice 3D binaire de taille $6(T-1)$x3xT. Notons que la représentation d'une stratégie par une matrice n'est pas unique, car certains états sont inaccessibles (par exemple, toutes les valeurs en $t = 1$ avec $p \neq 0$ ou $d \neq 1$).
Une représentation matricielle de la stratégie consistant à ne jamais rien acheter serait donc la matrice nulle. Celle consistant à acheter un dé dès que possible est $(\mathbb{1}_{\{p>6\} \cap \{d<3\}})$.

(c) Le simulateur est encodé dans la fonction *simulator_simple_game*.

```
Simulation de la stratégie "jamais acheter" : (35.067, [34.96158077756402, 35.17241922243598])
Simulation de la stratégie "toujours acheter" : (35.7476, [35.65250252825573, 35.842697471744266])
```

FIGURE 5 – Résultat du calcul de l'espérance et de leur intervalle de confiance à 95% des stratégies sur 10000 simulations

(d) Chaque lancer de dé X_i est supposé indépendant, de loi uniforme sur $[\![1, 6]\!]$. On a alors pour tout n :

$$P(\max_{i \in [\![1,n]\!]} X_i \leq k) = \bigcap_{i \in [\![1,n]\!]} P((X_i \leq k) = \frac{k^n}{6^n} \qquad \forall k \in [\![1, 6]\!]$$

$$P(\max_{i \in [\![1,n]\!]} X_i = k) = P(\max_{i \in [\![1,n]\!]} X_i \leq k) - P(\max_{i \in [\![1,n]\!]} X_i \leq k-1) = \frac{k^n - (k-1)^n}{6^n} \qquad \forall k \in [\![1, 6]\!]$$

Dans le cas $n = 2, 3$:

$$P(max(X_1, X_2) = k) = \frac{2k-1}{36}$$
$$P(max(X_1, X_2, X_3) = k) = \frac{3k^2 - 3k + 1}{216}$$

(e) Le code pour cette question est encodé dans la fonction *dp*. La valeur optimale est 37.6215 (les résultats sont trouvables sur le notebook, on gardera pour la suite 6 chiffres significatifs). La stratégie optimale n'achète au plus qu'un dé. Elle l'achète immédiatement lorsqu'elle a plus de 6 points si elle a suffisamment de points avant le tour 5. Au delà du tour 5, elle n'achète jamais.

```
t=1 : [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
t=2 : [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
t=3 : [0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
t=4 : [0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
t=5 : [0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
t=6 : [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
t=7 : [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
t=8 : [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
t=9 : [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
t=10 : [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

FIGURE 6 – Représentation de la stratégie optimale en fonction du nombre de points (incrémental) avec un seul dé

(f) La valeur est vérifiable via le simulateur :

```
(37.619269, [37.60940229921311, 37.6291357007869])
```

FIGURE 7 – Espérance et intervalle de conviance de la stratégie optimale pour un million de simulations

(g) Comme indiqué en (b), la stratégie "ne jamais acheter" peut être représentée par autre chose qu'une matrice nulle (à cause des valeurs inaccesibles). Ces valeurs inaccesibles sont :

- $\pi[p, 1, t]$ avec $p > 6(t-1)$ ou $p < t-1$
- $\pi[p, 2, t]$ avec $t = 1$ ou $p > 6(t-1) - 5$ ou $p < t-1-5$
- $\pi[p, 3, t]$ avec $t \leq 2$ ou $p > 6(t-1) - 10$ ou $p < t-1-10$

Qu'on peut résumer en : $\pi[p, d, t]$ avec $d+1 > t$ ou $p > 6(t-1)-5*(d-1)$ ou $p < t-1-5(d-1)$

(On peut remarquer que la formule précédente marche aussi pour d différent de 3)

Pour résoudre ce problème, nous avons utilisé la fonction *nullify_inaccessible_values* afin de fixer toutes les valeurs inaccessibles à 0. On trouve alors que la politique optimale est la politique nulle à partir du moment où $T \leq 6$.

(h) La fonction *no_dice_limit_simulator* permet de tester le jeu sans la contrainte de nombre de dés. Le nombre de dés maximum est à minima borné par le nombre de tour, donc une représentation matricielle est toujours possible. La programmation dynamique peut être executé à partir de la fonction *no_dice_limit_dp*.
Notons que la stratégie optimale n'achetait jamais plus qu'un dé pour T=10, donc relaxer la contrainte ne changerait par la stratégie optimale. En augmentant le nombre de tour à T=50, la stratégie optimale achète des dés au dela du 5ème dé (voir notebook).

5. Dépenser des dés

Dans cette partie, le joueur peut décider de dépenser en fin de tour l'un de ses dés (s'il en a plus d'un) pour doubler la valeur de son lancer. Remarquons que ce nouveau jeu est très différent du précédent : en effet, le joueur doit effectué une action avant et après le lancer. Il n'est donc pas possible de simplement "complexifier" le modèle précédent en ajoutant une action, il faut modifier entièrement les états. Pour celà, on découpe chaque tour en deux parties : avant le lancer et après le lancer. On pourrait doubler le pas de temps t, mais afin de garder la cohérence entre le code et les données de l'énoncé, nous allons introduire la variable binaire r et l'entier vr.

On définit alors un état $s = (p, d, vr, t, r)$ avec :

- $p_{t,r}$ le nombre de points du joueur au début du tour t
- $d_{t,r}$ le nombre de dés au début du tour t
- $vr_{t,r}$ est la valeur du lancer au tour t. Notons que $vr_{t,r}$ n'est pas déterminé pour r=0, on pourra ignorer sa valeur car elle n'interviendra pas dans les calculs.
- t le numéro du tour
- r indicatrice de si les dés du tour t ont déjà été lancés

L'espace des actions change selon la valeur de r :

- Si $r = 0$ (dés non lancés), $a_{t,r} = 1$ si un dé est acheté, 0 sinon
- Si $r = 1$ (dés lancés), $a_{t,r}$ =1 si le joueur vend un dé pour doubler son score

Pour une partie à T tours avec une limite de D dés, on a les bornes suivantes :

- $\forall(t, r), \quad 0 \leq p_{t,r} \leq 7t$ (le score ne peut être de plus de 7 au global car au mieux on double le score du dé et on paye pour le nouveau dé à chaque tour)
- $\forall(t, r), \quad 1 \leq d_{t,r} \leq D$ (pour la question (a), D = 5)
- $vr_{t,1} = \max_{i \in \{1, d_{t,0}+a_{t,0}\}}(u_i)$ (u_i est le score du ième dé)
- $p_{t,1} = p_{t,0} - 5 * a_{t+1,0} + vr_{t,1}$ (Notons qu'on attribue les points une première fois après avoir lancer les dés, et on les rajoute une deuxième fois si un dé est vendu)
- $p_{t+1,0} = p_{t,1} + a_{t,1} * (vr_{t,1})$
- $d_{t,1} = d_{t,0} + a_{t,0}$
- $d_{t+1,0} = d_{t,1} - a_{t,1}$

Avec notre modélisation du problème, une stratégie peut être représentée par une matrice 4D de taille $7T$ x D x 6 x T x 2 (l'unicité de la matrice pour une stratégie n'est toujours pas garantie).

(a) Le simulateur est appelable via la fonction *game2_ simulator*, et la progammation dynamique via la fonction *game2_ dp*.

La stratégie optimale est relativement complexe, mais certaines tendances peuvent être remarquées :

- L'achât de dé se fait dès que possible lorsqu'il reste plus de tours à jouer qu'il y a de dés (de façon à avoir le temps de les revendre avant la fin de la partie).
- Les dés sont vendus juste avant la fin de la partie.
- Si un lancer vaut 6, un dé sera en général vendu (logique, puisque le dé vaut 5 à l'achât, sa vente apporte un bénéfice de 1). Notons quand même que ce n'est pas toujours le cas en début de partie lorsque le joueur n'a que 2 dés, car le bénéfice à long terme d'avoir plus de dé est supérieur aux points apportés par la vente.

FIGURE 8 – Stratégie en début de tour quand $d = 2$

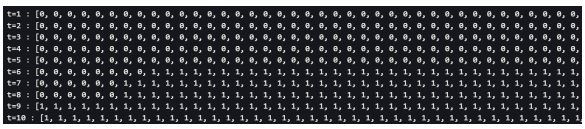

FIGURE 9 – Stratégie en fin de tour quand $d = 2$ et $vr = 6$

(b) De manière générale, lever la restriction du nombre de dés augmente l'espérance (on peut le voir dans le notebook avec la simulation du cas $T = 50$ avec et sans une limite $D = 5$). Cependant pour le cas $T = 10$, on ne constate pas d'augmentation de l'espérance : cela signifie qu'une partie de dépasse que rarement 5 dés (dans notre cas, la stratégie optimale consiste même à rester toujours en dessous de 5 dés afin d'avoir le temps de les vendre avant la fin de la partie).

On peut chercher à tracer l'évolution de l'espérance de la stratégie optimale en fonction de la restriction de dés (on remarque une stabilisation de l'espérance pour $D = 4$ quand $T = 10$ et $D = 12$ quand $T = 50$) :

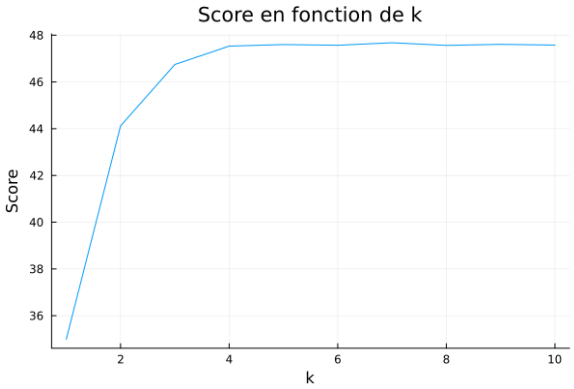

FIGURE 10 – $T = 10$

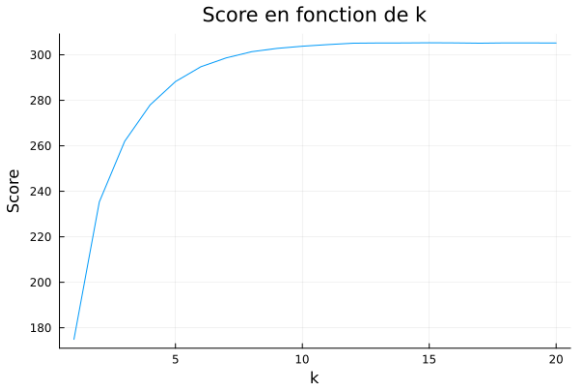

FIGURE 11 – $T = 50$

(c) L'implémentation est faite par la fonction *fcost_ game2_ dp*. La valeur optimale est 47.7016.

(d) La valeur du coup final ne doit plus dépendre que de d mais aussi de p (score). Cette valeur est $k_{p,d} = V(p, d, 10)$, avec $V(p, d, 1)$ l'espérance des gains lorsqu'on commence avec p points et d dés.

L'implémentation a été faite dans la fonction *final_ game2_ dp*. On trouve bien la même valeur optimale en calculant la politique optimale directement pour $T = 20$ qu'avec cette méthode (voir notebook).