IFT-7201: Apprentissage par renforcement
Devoir 1

Réalisé par: Wissam Akretche
Majid Heidary
Ayoub Echchahed

Présenté à: Audrey Durand

Octobre 2022

Note

Le code relatif à ce projet est contenu dans le jupyter notebook joint. Les algorithmes y sont implémentés dans l'ordre et les réponses sont numérotés suivant le numéro de la question associée.

1 La stratégie Explore-Then-Commit

On considère la stratégie Explore-Then-Commit (ETC) avec une phase exploratoire de durée $m = \frac{4\sigma^2}{\Delta^2} \ln \frac{T\Delta^2}{4\sigma^2}$ pour un gap de sous-optimalité minimal $\Delta = \min_{\substack{k \in \{1,2,\ldots,K\} \\ k \neq k_*}}$ permettant d'obtenir une borne supérieure sur le pseudo-regret cumulatif sur un horizon T:

$$R(T) \leq \min \left\{ \frac{4\sigma^2}{\Delta^2} \left[\ln \frac{T\Delta^2}{4\sigma^2} + 1 \right], \Delta T \right\} \tag{1}$$

dans les environnements de bandits stochastiques à K actions dont les récompenses sont échantillonnées de distributions σ-sous-Gaussiennes.

On considère la classe d'environnements de bandits stochastiques à deux actions tels que les récompenses associées à l'action k sont échantillonnées d'une distribution Bernoulli $\mathcal{B}(\mu_k)$, avec les espérances de récompenses μ_k contenues dans l'intervalle $[0.8, 1]$ pour toutes les actions k et avec $\Delta \geq 0.01$.

1.1 La durée de phase exploratoire (m) recommandée par la théorie

Les récompenses associées aux actions de la classe de bandits considérée sont tirés suivant des distributions de Bernoulli. Autrement dit, elles prennent des valeurs dans l'ensemble $\{0, 1\}$. Ces récompenses étant donc bornées entre 0 et 1, elles sont de distribution σ-sous-Gaussiennes avec $\sigma = 0.5$, selon le lemme de Hoeffding. Ainsi, en considérant la durée de la phase exploratoire énoncée ci-dessous, soit $m = \frac{4\sigma^2}{\Delta^2} \ln \frac{T\Delta^2}{4\sigma^2}$, pour $\sigma = 0.5$, on aura donc:

$$m = \frac{1}{\Delta^2} \ln T\Delta^2$$

Or, sachant que $0.01 \leq \Delta \leq 0.2$ ($\Delta \leq 0.2$ car les espérances des récompenses μ_k sont contenues dans l'intervalle $[0.8, 1]$ pour toutes les actions k), on aura ainsi deux cas extrêmes:

$$\Delta = 0.01 \rightarrow m = 10000 \ln(\frac{T}{10000})$$

$$\Delta = 0.2 \rightarrow m = 25 \ln(\frac{T}{25})$$

Mais ici, afin d'avoir une garantie sur l'ensemble des instances, on s'intéresse au pire cas, soit le cas où le gap de sous-optimalité serait le plus petit, donc $\Delta = 0.01$. Cela représenterait donc le scénario où il serait le plus difficile de distinguer les deux actions, et donc notre phase d'exploration (m) sera la plus grande pour pouvoir faire cette distinction. L'équation qui nous importe serait donc la première, soit:

$$m = 10000 \ln(\frac{T}{10000})$$

1.2 La longueur minimale d'horizon T

À partir de l'équation 6.5 du livre suggéré en cours Bandit Algorithms [3] : $m = max\left\{1, \left\lceil \frac{4\sigma^2}{\Delta^2} \log(\frac{T\Delta^2}{4\sigma^2}) \right\rceil\right\}$ il est possible d'affirmer que la durée d'exploration minimum est m=1, c'est à dire 1 passe sur les k actions disponibles. De ce fait, en remplaçant m=1 dans l'équation obtenue en a, $1 = 10000 \ln(\frac{T}{10000})$, puis en isolant le T, il est nous est possible de déduire que la valeur minimum possible pour T tout en respectant la théorie serait de T=10 001.

2 La stratégie KL-UCB

2.1 L'algorithme KL-UCB

Tout d'abord, afin de quantifier notre degré de confiance face aux paramètres inconnus d'une distribution, nous maintenons des intervalles de confiances *optimistes* sur les moyennes empiriques de chaque action. De ce fait, nous allons utiliser la divergance de Kullback–Leibler comme mesure de similarité entre différentes distributions. Plus précisemment, l'entropie relative entre deux distributions Bernouli avec paramètres $p, q \in [0, 1]$ est:

$$\mathrm{kl}(p, q) = p \ln(\frac{p}{q}) + (1 - p) \ln(\frac{1 - p}{1 - q}),$$

Tandis que la borne de confiance supérieure (optimiste) est donnée par:

$$UCB_k(t) = \max\left\{q : \ln(t) + c\ln(\ln(t)) - N_k(t-1) * \mathrm{kl}(\hat{u}_k(t-1), q) \geq 0\right\},$$

KL-UCB (Bernoulli) commence par jouer chaque action une fois via une phase d'exploration, puis l'agent devra sélectionner l'action qui possède la borne supérieure UCB maximale. Cette borne supérieure est calculé par la fonction $UCB_k(t)$ qui nous retourne la valeur maximale de q telle que l'action k soit considérée comme sous-échantillonée. Cette fonction prend en paramètre la moyenne empirique de l'action, le nombre de sélection de l'action (Nk), le pas de temps t ainsi qu'une constante c, puis elle nous retourne la valeur maximum de q. Voici le pseudo-code de l'algorithme:

Algorithme 1: KL-UCB (Bernoulli)

Entrée: Hyperparamètre c, taille de l'horizon T, nombre d'actions K
Initialisation des $N_k(t) = 0, \forall k \in \{1, \ldots, K\}$
Initialisation des récompenses cumulées $R_k(t) = 0, \forall k \in \{1, \ldots, K\}$
Pour $t = 1, \ldots, T$ **faire**
  Si $t \leq K$ **alors**
    $k_t = t$
  Sinon
    Pour *k=1, ..., K* **faire**
      $\hat{u}_k(t-1) = \frac{R_k(t-1)}{N_k(t-1)}$
      $UCB_k(t) = \max\left\{q : \ln(t) + c\ln(\ln(t)) - N_k(t-1) * \mathrm{kl}(\hat{u}_k(t-1), q) \geq 0\right\}$
    Sélectionner $k_t = \underset{k}{\mathrm{argmax}}\Big(UCB_k(t)\Big)$
  Jouer l'action k_t et observer la récompense r_t
  Mettre à jour $N_k(t) = N_k(t-1) + 1$
  Mettre à jour $R_k(t) = R_k(t-1) + 1$

Étant donné qu'il n'y a pas de forme close permettant de calculer cette borne en pratique, le calcul de la *Upper Confidence Bound*, pour une action donnée k à un instant t, soit $UCB_k(t)$, a été fait à

l'aide de la fonction **minimize** de la librairie **scipy.minimize** en minimisant la fonction objectif $-q$ (qui reviendrait à maximiser q) en respectant la contrainte

$$\ln(t) + c\ln(\ln(t)) - N_k(t-1) * \mathrm{kl}(\hat{u}_k(t-1), q) \geq 0$$

Ainsi, cette méthode présente des limites notamment du fait que l'on doit calculer cette borne à chaque pas de temps et pour toute action k à l'aide de la fonction d'optimisation, ce qui nécessite beaucoup de temps en pratique relativement aux méthodes où la UCB est calculée directement à l'aide d'une forme close telles que a stratégie KL-UCB (Normale).

2.2 KL-UCB Bernoulli VS KL-UCB Gaussien

Sur des bandits stochastiques avec récompenses à distribution Bernoulli, nous comparons la performance de notre stratégie, décrite précédemment, avec celle de KL-UCB utilisant la KL divergence de la distribution normale. Pour ce faire, nous considérons un horizon $T = 5000$ sur les instances suivantes: $\mu^{(1)} = \{0.1, 0.9\}$, $\mu^{(2)} = \{0.4, 0.7\}$, $\mu^{(2)} = \{0.45, 0.55\}$ et effectuons $N = 10$ répétitions par configuration. Les résultats obtenus sont présentés dans les figures ci-bas:

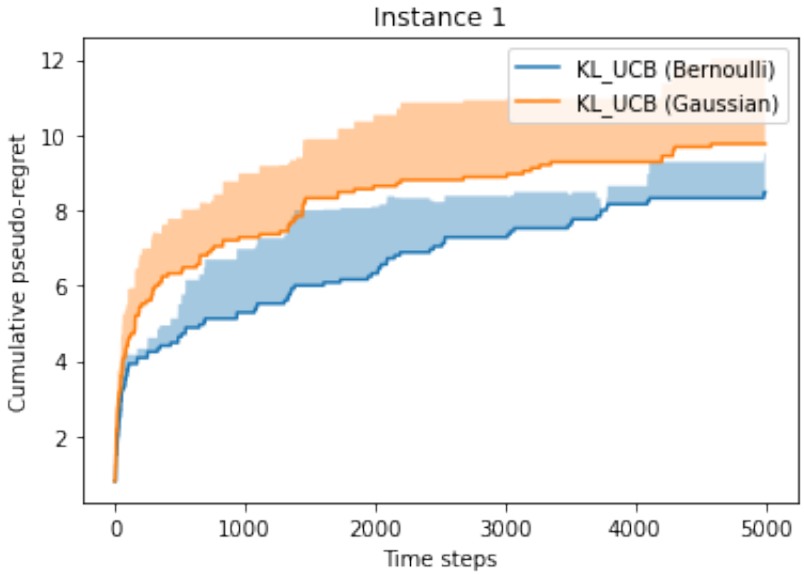

Figure 1: Évolution du pseudo-regret cumulatif dans le temps pour l'instance $\mu^{(1)} = \{0.1, 0.9\}$

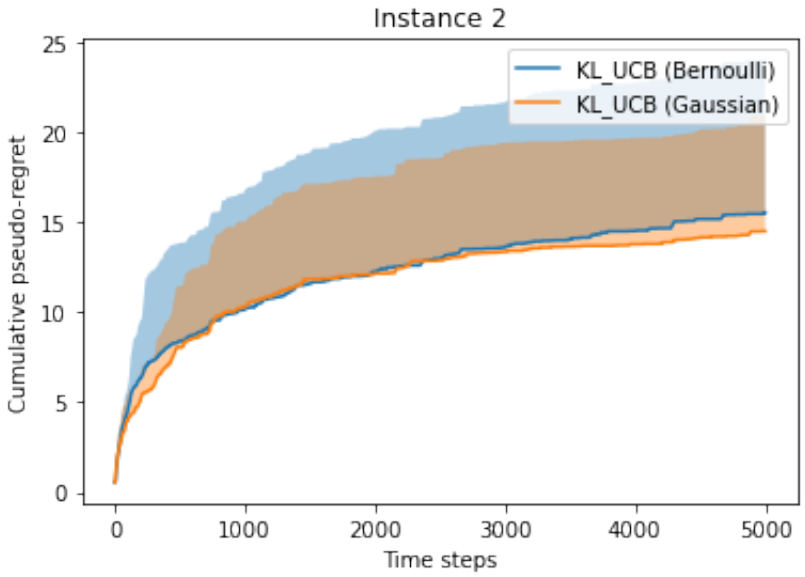

Figure 2: Évolution du pseudo-regret cumulatif dans le temps pour l'instance $\mu^{(2)} = \{0.4, 0.7\}$

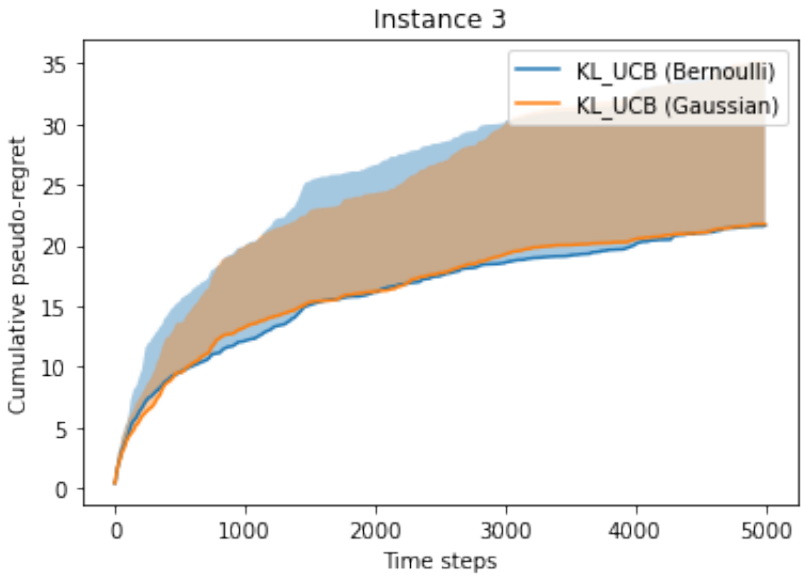

Figure 3: Évolution du pseudo-regret cumulatif dans le temps pour l'instance $\mu^{(3)} = \{0.45, 0.55\}$

Le but de cette comparaison est d'analyser la performance des stratégies KL-UCB (Normale) et KL-UCB (Bernoulli) à mesure que le gap entre les actions se resserre. Lorsque le gap entre les deux actions est relativement grand, comme c'est le cas pour $\mu^{(1)} = \{0.1, 0.9\}$, la différence de performance entre les deux stratégies est considèrable. Mais à mesure que le gap (et donc l'entropie relative) se resserre, et que les observations générées peuvent sembler venir d'une distribution normale plutôt que Bernoulli (la KL divergence normale et bernoulli se rapprochent), la différence de performance devient quant à elle moins certaine, comme il est possible de le voir via l'instance $\mu^{(2)} = \{0.4, 0.7\}$ et l'instance $\mu^{(3)} = \{0.45, 0.55\}$. Dans le cas $\mu^{(3)} = \{0.45, 0.55\}$, la KL divergence de bernoulli se rapproche de la KL divergence de la distribution normale. Ce qui se traduit par des performances similaires avec les deux approches.

À noter qu'il est possible d'utiliser la stratégie KL-UCB (Normale) appliquée à des observations

de récompense échantillonnées de distributions Bernoulli même si cela ne sera pas efficace dans une grande majorité des cas. Donc utiliser KL-UCB (Bernoulli) dans ce cas-ci serait souvent plus optimale car les intervalles de confiance seront plus adaptés à la distribution, donc plus serrés. En effet, les UCB obtenus en utilisant la KL Bernoulli se resserreront plus rapidement, de telle sorte qu'une meilleure convergence vers la vraie espérance de la distribution ayant générée les observations sera obtenue.

Par contre, la forme close pour le calcul de la borne supérieure UCB rend parfois la stratégie KL-UCB (Normale) plus intéressante côté simplification lors de l'implémentation de l'algorithme mais surtout le temps de calcul. Pour toutes les expériences précédentes, nous avons fixé $\sigma = 0.5$ pour KL-UCB (Normale) et ce car la variance maximale d'une distribution de Bernoulli est de $\sigma^2 = 0.25$.

3 La stratégie Thompson Sampling

On considère la classe d'environnements de bandits stochastiques à K actions tels que les récompenses associées à l'action k sont échantillonnées d'une distribution normale $\mathcal{N}(\mu_k, \sigma^2)$ variance σ^2 de valeur quelconque connue.

3.1 La stratégie Thompson Sampling avec variance σ^2 connue et prior $\mathcal{N}(\mu_0, \sigma_0^2)$

Étant donné un prior $\mathcal{N}(\mu_0, \sigma_0^2)$, une variance connue σ^2, la distribution posterior est donnée sous la forme suivante, suivant les n observations $x_1, \ldots, x_n$ rencontrées au cours des expériences [1]:

$$\frac{1}{\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}} \left(\frac{\mu_0}{\sigma_0^2} + \frac{\sum_{i=1}^{n} x_i}{\sigma^2} \right), \left(\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2} \right)^{-1}$$

Ainsi, le pseudo-code de l'algorithme sera donné comme suit:

Algorithme 2: Thompson Sampling

Entrée: Taille de l'horizon T, nombre d'actions K, priors μ_0 et σ_0, variance σ^2
Initialisation des $N_k(t) = 0, \forall k \in \{1, \ldots, K\}$
Initialisation des récompenses cumulées $R_k(t) = 0, \forall k \in \{1, \ldots, K\}$
Pour $t = 1, \ldots, T$ **faire**
  Pour *k=1, ..., K* **faire**
    $\hat{u}_k(t) = \frac{1}{\frac{1}{\sigma_0^2} + \frac{N_k(t-1)}{\sigma^2}} \left(\frac{\mu_0}{\sigma_0^2} + \frac{R_k(t-1)}{\sigma^2} \right)$
    $\hat{\sigma}_k(t) = \left(\frac{1}{\sigma_0^2} + \frac{N_k(t-1)}{\sigma^2} \right)^{-1}$
    Sample $\theta_{k,t} \sim \mathcal{N}(\hat{u}_k(t), \hat{\sigma}_k(t))$
  Sélectionner $k_t = \underset{k}{\text{argmax}} \theta_{k,t}$
  Jouer l'action k_t et observer la récompense r_t
  Mettre à jour $N_k(t) = N_k(t-1) + 1$
  Mettre à jour $R_k(t) = R_k(t-1) + 1$

3.2 Valeurs des paramètre (σ^2 et priors) pour lesquelles notre stratégie équivaut à TS-Normal d'Agrawal and Goyal[1]

La stratégie TS-Normal d'Agrawal and Goyal est en fait un cas spécial de la stratégie présentée ci-dessus. En effet, pour des valeurs $\mu_0 = 0$, $\sigma = \sigma_0 = 1$, nos mises à jours sont identiques aux leurs.

À chaque pas de temps t, et pour toute action $k \in \{1, \ldots, K\}$, dans la stratégie d'Agrawal and Goyal, on échantillonne, de manière indépendante:

$$\theta_{k,t} \sim \mathcal{N}(\hat{u}_k(t), \hat{\sigma}_k(t))$$

[1] `https://en.wikipedia.org/wiki/Conjugate_prior`

On sélectionne k_t et on observe la récompense r_t. Puis, on met à jour les paramètres $\hat{u}_k(t)$ et $\hat{\sigma}_k(t)$ comme suit:

$$\hat{u}_k(t+1) = \frac{\hat{u}_k(t)k_i(t) + r_t}{k_i(t)+1} = \frac{R_k(t)}{N_k(t)}$$

$$\hat{\sigma}_k(t+1) = \frac{1}{k_i(t)+1} = \frac{1}{N_k(t)+1}$$

avec $\hat{u}_k(1) = 0$ et $\hat{\sigma}_k(1) = 0, \forall k \in \{1, \ldots, K\}$

Notons que dans notre méthode, nos paramètres sont calculés avant l'échantillonnage (au début de chaque itération) et que dans la version d'Agrawal and Goyal c'est fait à la fin (donc pour l'itération suivante).

Dans notre méthode, pour $\mu_0 = 0$ et $\sigma = \sigma_0 = 1$, nos paramètres sont définis comme suit:

$$\hat{u}_k(t) = \frac{R_k(t-1)}{N_k(t-1)}$$

$$\hat{\sigma}_k(t) = \frac{1}{N_k(t-1)+1}$$

Les deux méthodes sont donc complètement identiques pour $\mu_0 = 0$ et $\sigma = \sigma_0 = 1$.

3.3 L'impact de la valeur du prior μ_0

Nous évaluons l'impact de la valeur du prior μ_0 en réalisant des expériences sur des bandits stochastiques dans la classe considérée avec $\sigma^2 = 0.25$. pour ce faire, nous appliquons notre stratégie sur 100 instances à $K = 2$ actions dont les espérances μ_k sont échantillonnées uniformément sur l'intervalle $[0, 10]$). Nous considérons un horizon $T = 50$, un prior $\sigma_0 = 5$ et comparons nos performances pour les valeurs de prior $\mu_0 \in \{0, 5, 10\}$. la figure ci-dessous montre les performances enregistrées:

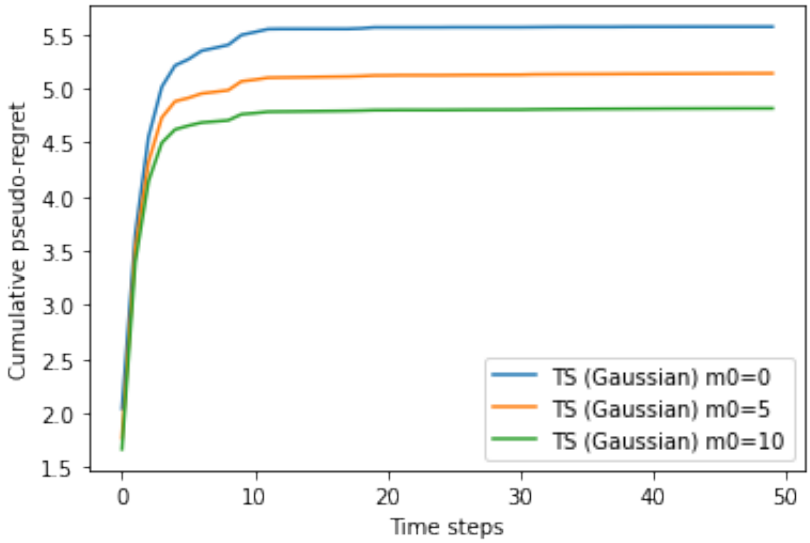

Figure 4: Évolution de pseudo-regret cumulatif dans le temps pour différentes valeurs de μ_0

La figure montre que la stratégie TS est assez robuste face aux différentes valeurs du prior μ_0. Le prior parfait étant $\mu_0 = 5$, sa sur-estimation ne semble pas détériorer la perfomance de la stratégie, bien au contraire ça semble l'améiorer. En revanche, la sous-estimation de μ_0 semble détériorer la performance de la stratégie mais elle reste relativement robuste. La stochasticité présente dans TS permet d'atténuer les effets de la méconnaissances des priors et les mises à jours du posterior μ au fil du temps permettent se rapprocher de la vraie estimation en se basant sur les observations rencontrées au fur et à mesure. Finalement, il semble que la valeur du prior $\mu_0 = 10$ est celle qui minimise le pseudo regret cumulatif.

3.4 La robustesse de la stratégie à une mauvaise connaissance de la variance σ^2

Ici, nous évaluons l'impact de d'une mauvaise connaissance de la variance σ^2 en réalisant des expériences sur des bandits stochastiques dans la classe considérée avec $\sigma^2 = 0.25$. pour ce faire, nous appliquons notre stratégie sur 100 instances à $K = 2$ actions dont les espérances μ_k sont échantillonnées d'une distribution normale $\mu_k \sim \mathcal{N}(0, 1)$. Nous considérons un horizon $T = 1000$, un prior $\sigma_0 = 1$ et $\mu_0 = 0$ (les priors parfaits) et comparons nos performances pour les valeurs $\sigma \in \{0.0001, 0.5, 1, 2, 5\}$. la figure ci-dessous montre les performances enregistrées:

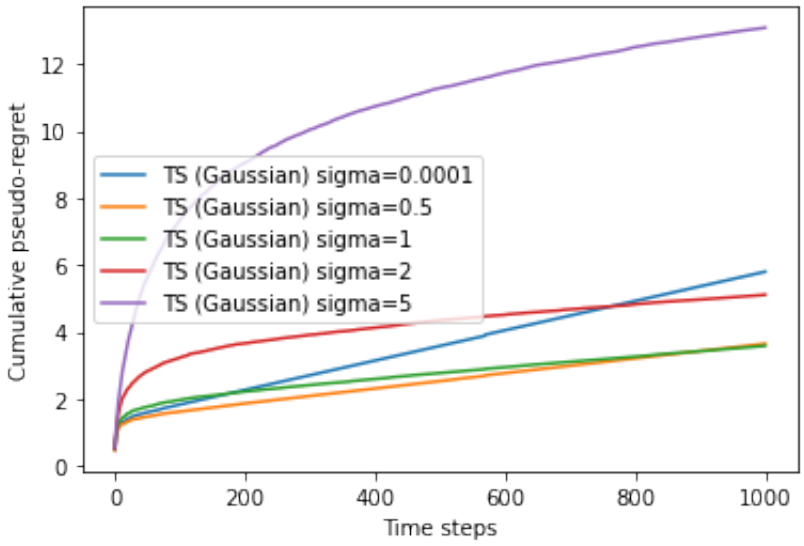

Figure 5: Évolution de pseudo-regret cumulatif dans le temps pour différentes valeurs de σ

Sans surprise, on constate que plus la *connaissance* de σ s'éloigne de la réalité, qui est $\sigma = 0.5$, plus les les performances se dégradent. En effet, on constate qu'un σ trop sur-estimé ou sous-estimé mène à des performances bien inférieures que celle obtenue avec une vraie connaissance de σ. On voit ceci notamment avec la pire performance pour $\sigma = 5$, suivi par $\sigma = 0.0001$ et $\sigma = 2$. On enregistre les meilleures performances pour la vraie valeur de σ mais il reste assez robuste pour une valeur $\sigma = 1$ qui n'est relativement pas loin de la vraie valeur. Les valeur trop grandes mènent à une sur-exploration tandis que les plus petites mènent à une sur-exploitation ce qui conduit dans les deux cas à une détérioration des performances. Le pseudo-regret cumulatif reste toutefois sous-linéaire.

4 Les approches Bayésiennes

On considère la classe d'environnements de bandits stochastiques à K actions tels que les récompenses associées à l'action k sont échantillonnées d'une distribution Bernoulli $\mathcal{B}(\mu_k)$

4.1 Implémentation des stratégies Bayes-UCB et Thompson Sampling

Similairement à la stratégie Thompson Sampling (TS), Bayes-UCB est une stratégie qui va se baser sur une distribution posterior sur l'espérance de récompense pour chaque action. Contrairement à TS, Bayes-UCB commence par jouer chaque action une fois. Cette phase d'exploration n'est pas nécessaire dans (TS) puisque cette stratégie est stochastique et que l'exploration se fait naturellement lors de l'échantillonnage des $\theta_{k,t}$ avec le chevauchement entre les distributions relatives aux différentes actions. Ainsi, à chaque étape t, TS va échantillonner des paramètres $\theta_{k,t} \sim \text{Beta}(\alpha + R_k(t-1), \beta + N_k(t-1) - R_k(t-1))$ tandis que Bayes-UCB calcule l'UCB de chaque action qui n'est autre que le quantile d'ordre $1 - \frac{1}{t(\ln T)^c}$ de la distribution $\text{Beta}(\alpha + R_k(t-1), \beta + N_k(t-1) - R_k(t-1))\Big)$ qui est la même que pour TS. Les deux stratégies sélectionnent ensuite l'action maximisant $\theta_{k,t}$ pour TS et $UCB_k(t)$

pour Bayes-UCB avant d'observer la récompense résultantes et mettre à jour $N_k(t)$ et $R_k(t)$. Les deux méthodes nécessitent des paramètres prior α et β. Le pseudo-code de chacune des méthodes est donné ci-dessous:

Algorithme 3: Thompson Sampling (Bernoulli)

Entrée: Taille de l'horizon T, nombre d'actions K, priors α et β
Initialisation des $N_k(t) = 0, \forall k \in \{1, \dots, K\}$
Initialisation des récompenses cumulées $R_k(t) = 0, \forall k \in \{1, \dots, K\}$
Pour $t = 1, \dots, T$ **faire**
  Pour *k=1, ..., K* **faire**
    Sample $\theta_{k,t} \sim \text{Beta}(\alpha + R_k(t-1), \beta + N_k(t-1) - R_k(t-1))$
  Sélectionner $k_t = \underset{k}{\text{argmax}} \theta_{k,t}$
  Jouer l'action k_t et observer la récompense r_t
  Mettre à jour $N_k(t) = N_k(t-1) + 1$
  Mettre à jour $R_k(t) = R_k(t-1) + 1$

Algorithme 4: Bayes-UCB (Bernoulli)

Entrée: Taille de l'horizon T, nombre d'actions K, priors α et β
Initialisation des $N_k(t) = 0, \forall k \in \{1, \dots, K\}$
Initialisation des récompenses cumulées $R_k(t) = 0, \forall k \in \{1, \dots, K\}$
Pour $t = 1, \dots, T$ **faire**
  Si $t \leq K$ **alors**
    $k_t = t$
  Sinon
    Pour *k=1, ..., K* **faire**
      Calculer le Quantile d'ordre $1 - \frac{1}{t(\ln T)^c)}$ de la distribution posterior:
      $UCB_k(t) = \mathcal{Q}\Big(1 - \frac{1}{t(\ln T)^c}, \text{Beta}(\alpha + R_k(t-1), \beta + N_k(t-1) - R_k(t-1))\Big)$
    Sélectionner $k_t = \underset{k}{\text{argmax}} UCB_k(t)$
  Jouer l'action k_t et observer la récompense r_t
  Mettre à jour $N_k(t) = N_k(t-1) + 1$
  Mettre à jour $R_k(t) = R_k(t-1) + 1$

4.2 Bayes-UCB VS Thompson Sampling

On compare les stratégies précédentes sur des instances de la classe d'environnements donnée. Plus spécifiquement, nous considérons 100 instances à $K = 2$ actions générées en échantillonnant les espérances μ_k d'une distribution Beta(2,5). Nous exécutons chaque stratégie durant $T = 1000$ pas de temps sur chaque instance et utilisons les priors parfaits et fixons la valeur de l'hyperparamètre $c = 5$. Les résultats obtenus sont présentés dans les figures ci-dessous:

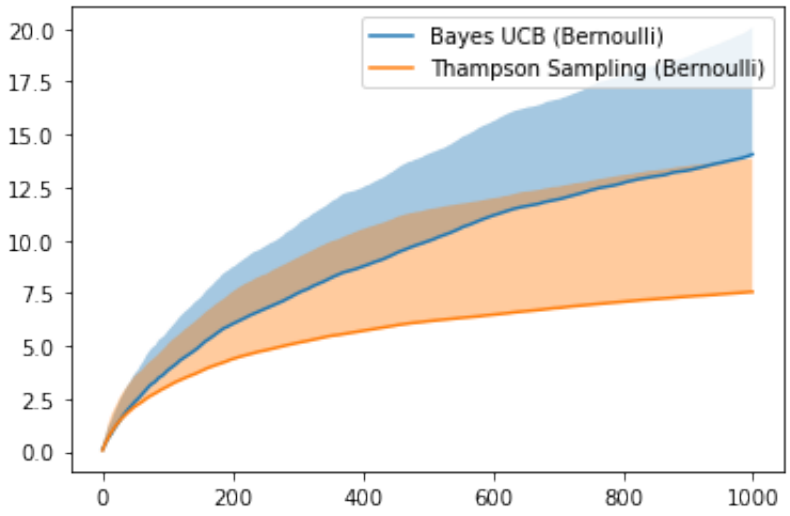

Figure 6: Évolution du pseudo-regret cumulatif dans le temps pour Bayes UCB et TS

Il est possible de constater que la stratégie Thompson Sampling (Bernoulli) minimisent beaucoup plus efficacement le pseudo-regret cumulatif à travers le temps si on la compare à la stratégie Bayes-UCB. Maintenant si l'on compare la nature des deux stratégies, les approches de types UCB sont considérées comme des stratégies optimistes car elles comparent les actions en se basant sur leurs meilleures valeurs possibles, tandis que TS est plutôt une approche réaliste comparant les actions en se basant sur des échantillons de leur distribution posterior pouvant prendre n'importe quelle valeur avec une probabilité dépendant de la connaissance a priori et de l'historique des observations obtenues jusqu'à présent. Ainsi, TS permet de capitaliser sur les fortes machines tout en explorant les machines possédant du potentiel grâce à son aspect non-déterministe. Ce n'est d'ailleurs pas le cas des approches UCB, qui sont déterministe quant à la sélection d'actions. Autrement dit, pour un historique d'observation fixe, on peut calculer directement l'action qui va être sélectionnée par UCB, ce qui n'est pas le cas avec TS.

4.3 La robustesse de TS et Bayes-UCB à des mises à jour en *batch*

Dans cette section, nous étudions la robustesse de TS et Bayes-UCB à des mises à jour en *batch*. Pour ce faire, nous utilisons les mêmes paramètres que précédemment afin de mieux étudier l'impact du paramètre L, qui représente la taille du buffer. Nous considérons 100 instances à $K = 2$ actions générées en échantillonnant les espérances μ_k d'une distribution Beta(2,5). Nous exécutons chaque stratégie durant $T = 1000$ pas de temps sur chaque instance et utilisons les priors parfaits et fixons la valeur de l'hyperparamètre $c = 5$. Les résultats obtenus sont présentés dans les figures ci-dessous.

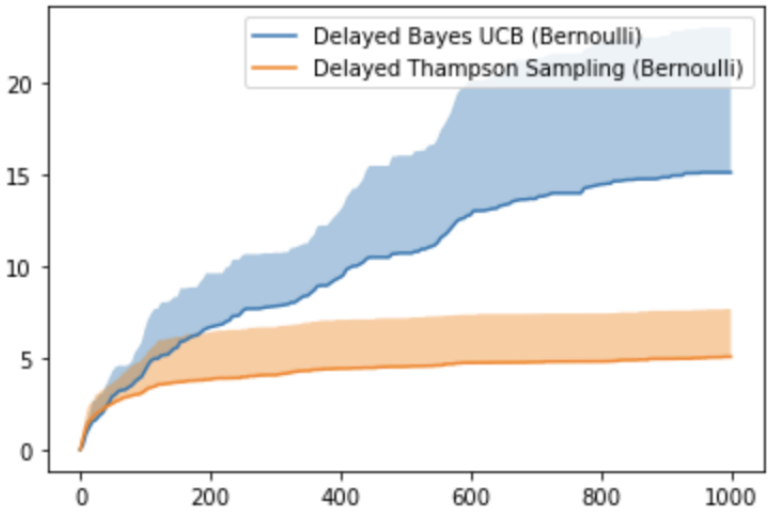

Figure 7: Évolution du pseudo regret cumulatif pour TS et Bayes-UCB avec récompenses à retardement pour L=3

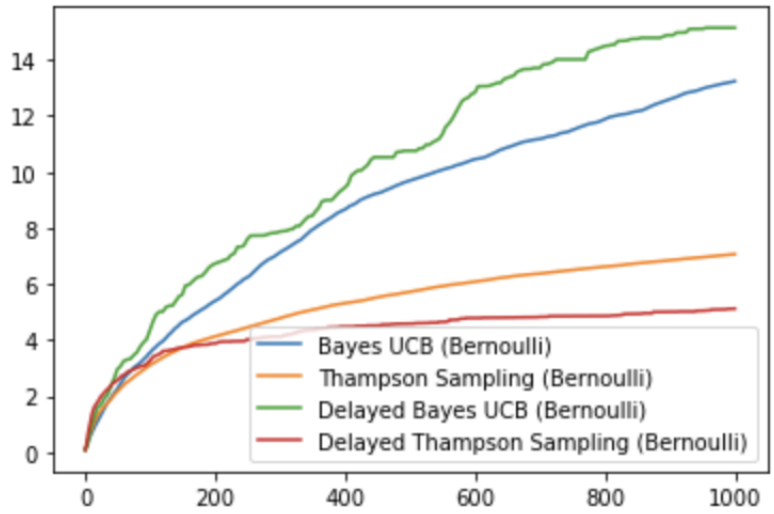

Figure 8: Évolution du pseudo regret cumulatif pour TS et Bayes-UCB avec et sans retardement des récompenses

Dans les expériences résumées dans les figures 8 et 7, nous avons considéré $L = 3$. Dans la figure 8 nous comparons les deux méthodes avec mises à jours retardées et dans la seconde nous comparons les versions originales aux versions avec délai. Par souci de lisibilité, nous avons supprimé les variances dans la deuxième figure afin de mieux observer les résultats. Les figurent montrent que les délais introduits détériorent considérablement Bayes-UCB tandis que ça ne semble pas affecter TS, au contraire les résultats obtenus avec la stratégies dont les mises à jours sont retardées semblent meilleures...

Afin de mieux analyser ces résultats et observer l'impact du paramètre L, nous enregistrons cette fois les performances des méthodes avec mises à jours retardées en considérant différentes valeurs de la tailles du *buffer* $L \in \{3, 5, 15\}$.

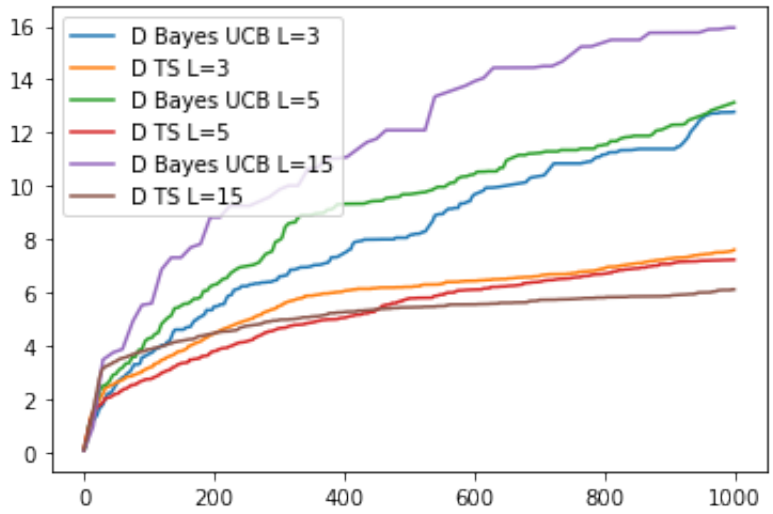

Figure 9: Évolution du pseudo regret cumulatif pour TS et Bayes-UCB avec récompenses à retardement pour différentes tailles du buffer L

Les résultats obtenus lors de nos expériences confirme ceux de Chapelle and Li [2] qui affirment que TS serait plus robuste qu'UCB à des mises à jour en *batch*. En effet, en s'appuyant sur les figures ci-dessous, en particulier la figure 8, nous pouvons constater que TS est très robuste aux mises à jours à retardement puisque les performances sont inchangées avec l'introduction du délai (voire meilleures pour $L = 15$). En revanche, Bayes-UCB enregistre sa pire performance pour $L = 15$ et s'améliore avec la réduction du délai. Cela va dans le même sens que cette citation issue du papier venant des auteurs de la question précédente: *"An interesting quantity in this simulation is the relative regret of UCB and Thompson sampling. It appears that Thompson sampling is more robust than UCB when the delay is long. Thompson sampling alleviates the influence of delayed feedback by randomizing over actions; on the other hand, UCB is deterministic and suffers a larger regret in case of a sub-optimal choice."*.

Ainsi, la stochasticité de Thompson Sampling permet d'atténuer l'effet du retard tandis que pour UCB, qui est déterministe, les délais entraînent la sélection des action sous-optimale pendant les L pas de temps nécessitant de remplir le buffer puisque les observations restent inchangées et que le processus est déterministe et se traduit ainsi par l'augmentation du pseudo-regret cumulatif au cours du temps.

5 L'algorithme Kernel-TS pour les bandits contextuels

On considère la classe d'environnements de bandits contextuels à actions disjointes tel que pour chaque action k, l'espérance de récompense pour un contexte s est donnée par la fonction:

$$f_k(s) =< \theta_k, \phi(s) >,$$

où: $\phi(s) = [1, s, s^2, s^3, s^4]$ dénote les features du contexte s, et que la récompense obtenue en jouant l'action k_t dans le contexte s_t est donnée par $r_t \sim \mathcal{N}(f_{k_t}(s_t)), \sigma^2$. La variance du bruit sur les observations σ^2 est inconnue, mais on suppose que l'on connait une borne supérieure $R^2 \geq \sigma^2$.

5.1 Visualisation des fonctions de récompenses associées à chaque action sur l'espace des contextes

Dans un premier temps, nous générons aléatoirement une instance de l'environnement décrit ci-haut avec $K = 10$ actions. pour chaque action k, nous générons un paramètre vectoriel θ_k tel que chaque composante du vecteur est échantillonnée uniformément sur l'intervalle $[-1, 1]$ et que $\|\theta_k\|_2 = 1$ pour tout k. Les fonctions de récompenses, obtenues, associées à chaque action sur l'espace des contextes sont présentées dans la figure ci-dessous:

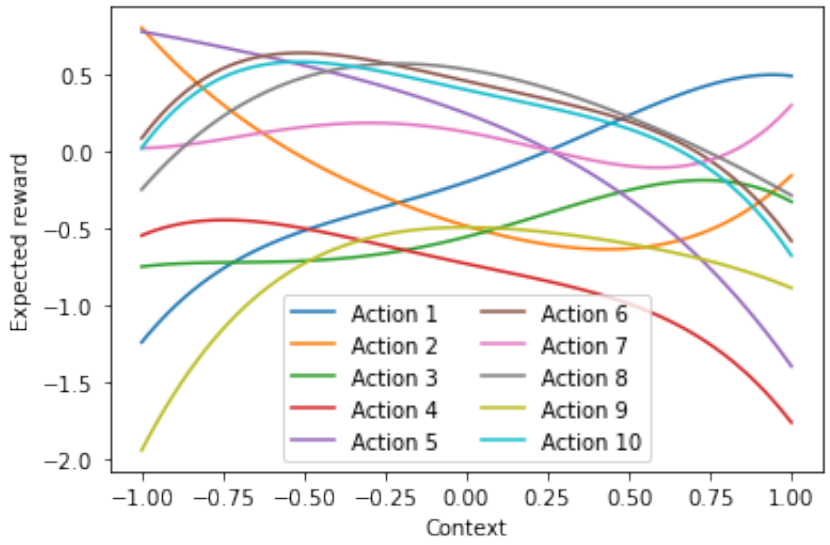

Figure 10: Visualisation des fonctions de récompenses associées à chaque action sur l'espace des contextes

5.2 Algorithme Kernel-TS

Le pseudo-code de l'algorithme Kernel-TS pour l'environnement de bandits contextuels à actions disjointes est donné ci-dessous.

Algorithme 5: Kernel-TS

Entrée: Taille de l'horizon T, nombre d'actions K, l'espace des contextes $\mathcal{S}$, paramètre de régularisation λ, borne supérieure de la variance R^2

Initialisation des $N_k(t) = 0, \forall k \in \{1, \dots, K\}$

Initialiser les paramètres des modèles associés à chaque action *(on aura un modèle par action étant donné qu'on travaille sur des bandits contextuels à actions disjointes)*

Pour $t = 1, \dots, T$ **faire**

- Observer le contexte s_t
- **Pour** *k=1, ..., K* **faire**
  - Calculer la moyenne et la covariance sur les $t-1$ observations précédentes:
  - $\hat{f}_{k,t-1} = \left(\hat{f}_{k,t-1}(s)\right)_{s \in \mathcal{S}}$
  - $\hat{\Sigma}_{k,t-1} = \frac{R^2}{\lambda}\left(K_{\lambda,k,t-1}(s,s')\right)_{s,s' \in \mathcal{S}}$, K étant la fonction noyau
  - Échantillonner une fonction $\tilde{f}_{k,t} \sim \mathcal{N}_{|\mathcal{S}|}(\hat{f}_{k,t-1}, \hat{\Sigma}_{k,t-1})$
- Sélectionner $k_t = \underset{k}{\operatorname{argmax}} \tilde{f}_{k,t}(s_t)$
- Jouer l'action k_t et observer la récompense r_t
- Mettre à jour les observations associées à l'action k_t (la récompense r_t associée au contexte s_t)

5.3 L'impact du paramètre R^2

Dans le problème énoncé ci-haut, la variance σ^2 est inconnue, mais on suppose connaître une borne supérieure R^2. Dans ce qui suit, on va évaluer l'impact de ce dernier en effectuant des expériences sur des instances de la classe donnée pour $\sigma^2 = 0.25$. Nous considérons les valeurs suivantes $R \in \{0.5, 1, 2, 5, 10\}$. Pour chacune de ces valeurs, nous effectuons $N = 10$ répétitions de $T = 250$ pas de temps sur une instance donnée. Les résultats obtenus pour notre première instance sont preésentés dans la figure 11.

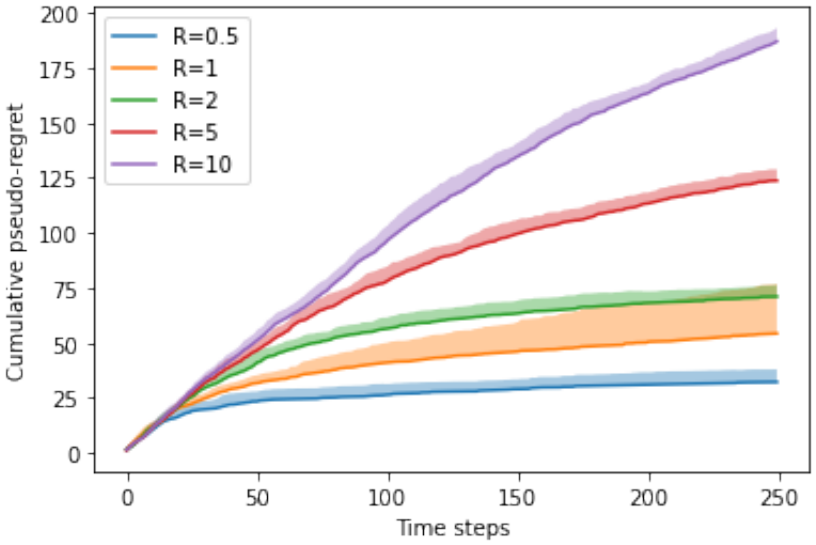

Figure 11: Évolution du pseudo regret cumulatif pour Kernel-TS pour différentes valeurs de R évalué sur une seule instance

Cette figure montre que plus R s'éloigne de la vraie valeur, plus les performances sont détériorées. Autrement dit, plus on sur-estime R, ce qui se traduit par une méconnaissance de l'environnement ou une incertitude, plus on sur-explore. En effet, l'augmentation de R^2 implique l'augmentation de $\hat{\Sigma}_{k,t-1} = \frac{R^2}{\lambda}\left(K_{\lambda,k,t-1}(s,s')\right)_{s,s' \in \mathcal{S}}$ ce qui mène à une sur-exploration, qui se traduit par l'augmentation du pseudo-regret cumulatif au cours du temps.

Dans un second temps, nous avons répété l'expérience précédente sur $N = 5$ instances différentes et avons enregistré leurs performances (La moyenne et la variance relative aux différentes instances) en gardant les mêmes paramètres que précédemment. Les résultats sont présentés dans la figure ci-dessous.

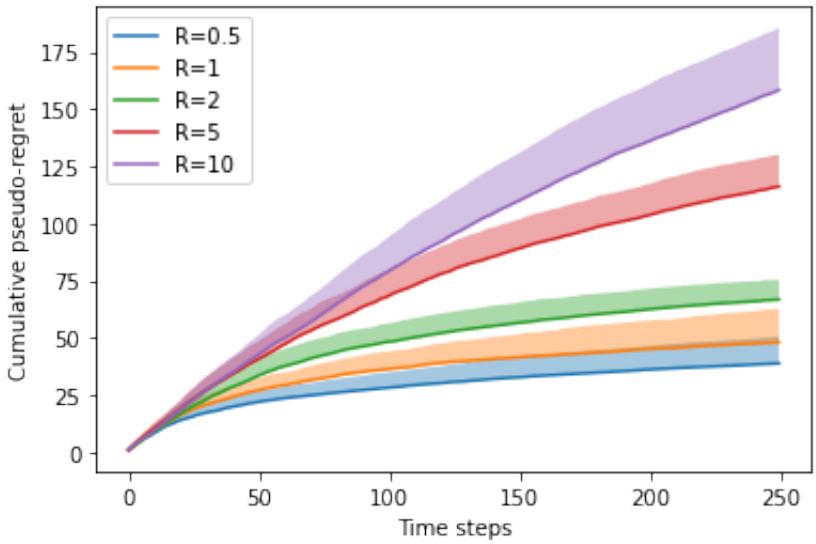

Figure 12: Évolution du pseudo regret cumulatif pour Kernel-TS pour différentes valeurs de R évalué sur plusieurs instance

Cette figure confirme les résultats discutés ci-haut puisqu'on enregistre la pire performance pour $R = 10$ qui surestime excessivement σ tandis que les meilleurs résultats sont obtenus pour $R = \sigma$. On constate également une grande variance dans les résultats observés pour $R = 10$ ce qui pourrait s'expliquer encore une fois par une valeure élevée de $\hat{\Sigma}_{k,t-1}$ qui introduit plus de stochasticité dans la procédure (ce qui se traduit aussi par une sur-exploration)"

Références

[1] Shipra Agrawal and Navin Goyal. Further optimal regret bounds for thompson sampling. In Carlos M. Carvalho and Pradeep Ravikumar, editors, *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, volume 31 of *Proceedings of Machine Learning Research*, pages 99–107, Scottsdale, Arizona, USA, 29 Apr–01 May 2013. PMLR.

[2] Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[3] Tor Lattimore. *Bandit algorithms / Tor Lattimore (deepMind) and Csaba Szepesvari (University of Alberta).* Cambridge University Press, Cambridge, United Kingdom ;, 2020 - 2020.