Garanties d'approximation pour la résolution de POMDP via échantillonnage et réseaux de neurones convexes

Alix Danvy[1,2], Jilles S. Dibangoye[3], Erwan Escudie[3], Alexandre Niveau[2], Bruno Zanuttini[2]

[1] ENS Rennes, France
[2] Université Caen Normandie, ENSICAEN, CNRS, Normandie Univ ;
GREYC UMR6072, F-14000 Caen, France
[3] Bernoulli Institute, University of Groningen, Nijenborgh 4,
NL-9747 AG Groningen, Netherlands

alix.danvy@ens-rennes.fr, j.s.dibangoye@rug.nl, e.c.escudie@rug.nl,
alexandre.niveau@unicaen.fr, bruno.zanuttini@unicaen.fr,

Résumé

Nous présentons une nouvelle approche de résolution pour les processus décisionnels de Markov partiellement observables. Cette approche échantillonne l'espace d'états de croyance à la manière de PBVI, et approxime la fonction de valeur avec des réseaux de neurones convexes tout en contenant leur constante de Lipschitz. Nous donnons des garanties formelles d'approximation.

Mots-clés

Processus décisionnel de Markov partiellement observable, POMDP, Réseau de neurones convexe en l'entrée, ICNN, Itération de la valeur par échantillonnage, PBVI

Abstract

We present a novel solving approach for Partially Observable Markov Decision Processes. This approach samples the belief state space in the manner of PBVI, and approximates the value function using input-convex neural networks while containing their Lipschitz constant. We provide formal approximation guarantees.

Keywords

Partially Observable Markov Decision Process, POMDP, Input-Convex Neural Network, ICNN, Point-Based Value Iteration, PBVI

1 Introduction

La prise de décision séquentielle dans des environnements incertains et partiellement observables reste un des défis importants de l'intelligence artificielle. Le formalisme des Processus Décisionnels de Markov Partiellement Observables (POMDP) permet de modéliser ces problèmes en maintenant une distribution de probabilité continue sur les états possibles, appelée état de croyance (*belief state*). Pour résoudre mathématiquement un POMDP, on le reformule généralement comme un Processus Décisionnel de Markov (MDP) continu parfaitement observable sur l'espace des belief state, communément appelé *Belief-MDP*. Historiquement, il a été prouvé que pour un horizon fini, la fonction de valeur optimale de ce *Belief-MDP* est convexe [24] et linéaire par morceaux (PWLC) [18], pouvant être représentée exactement par un ensemble fini d'hyperplans, ou α-vecteurs. Dans le cas d'un horizon infini, la fonction de valeur conserve sa stricte convexité sur l'espace des croyances, mais peut nécessiter d'une infinité d'hyperplans pour être représentée de manière exacte [20].

Les algorithmes de planification de l'état de l'art, à l'instar de Point-Based Value Iteration (PBVI) [14], exploitent cette propriété géométrique en restreignant les mises à jour de Bellman à un ensemble fini de points de croyance échantillonnés. Bien que PBVI offre de solides garanties théoriques avec une borne d'erreur proportionnelle à la densité de l'échantillonnage, il se heurte à un goulot d'étranglement calculatoire majeur : la taille de l'ensemble d'hyperplans généré est proportionnelle au nombre de points échantillonnés, le maintien des hyperplans est coûteux, ce qui limite son application à des espaces restreints.

Face à cette explosion combinatoire, l'utilisation de réseaux de neurones profonds a récemment émergé comme une alternative prometteuse, remplaçant les calculs algébriques discrets par des approximateurs continus hautement paramétrés. Des travaux de planification neurale comme Neural Value Iteration [23] ou BetaZero [13] ont illustré la capacité de ces architectures à briser la malédiction de la dimensionnalité sur l'espace des croyances. Plus récemment, des approches issues de l'Apprentissage par Renforcement Profond (Deep RL), telles que "Convex is back" [10], ont commencé à réintroduire le biais inductif de convexité dans la résolution des Belief-MDPs. Néanmoins, ces approches sacrifient généralement les garanties de convergence et souffrent d'instabilité lors des mises à jour de l'approximateur. De plus, elles reposent principalement sur l'échantillonnage de trajectoires et n'exploitent donc pas le modèle complet du problème (les probabilités exactes de transition et d'observation) à son plein potentiel pour calculer des es-

pérances de Bellman complètes.
Pour combler ce fossé entre scalabilité empirique et garanties analytiques, nous introduisons **ICNN-PBVI**, une nouvelle méthode de planification qui intègre la robustesse algorithmique de PBVI au sein des Réseaux de Neurones Convexes (ICNN) [1]. Nos contributions principales se déclinent ainsi :

- **Backup neuronal supervisé par Bellman** : Nous substituons l'expansion géométrique des α-vecteurs par l'entraînement d'un ICNN. L'opérateur de Bellman exact, calculé à partir du modèle connu, génère des cibles de régression stables pour l'optimisation du réseau sur les points de croyance.
- **Architecture GL-ICNN (Globally Lipschitz ICNN)** : Afin de garantir que notre approximateur respecte la contrainte de régularité lipschitzienne inhérente à la fonction de valeur, nous concevons une couche de normalisation globale différentiable. En couplant une borne supérieure dynamique issue de l'algorithme AutoLip [16] avec une activation lisse *SmoothMax* et un redimensionnement des récompenses cibles (Value Target Rescaling) [15], notre modèle évite l'effondrement du gradient tout en respectant strictement la constante de Lipschitz théorique du problème.
- **Preuve formelle de la borne d'erreur** : Nous établissons une analyse de convergence généralisée démontrant que la fonction de valeur approchée par ICNN-PBVI conserve une borne d'erreur théorique solide, dépendante de l'erreur empirique de *fitting* neuronale et de la densité de l'échantillonnage de l'espace des croyances.

Dans la section 2, nous rappelons les fondements des POMDP et passons en revue les travaux connexes pertinents. La section 3 détaille la formulation de l'algorithme ICNN-PBVI et l'architecture spécifique du GL-ICNN. La section 4 est consacrée à notre analyse de convergence mathématique, suivie de la section 5 qui expose notre implémentation et notre plan d'expériences. Enfin, la section 6 conclut l'article et dessine nos perspectives de recherche futures.

2 Préliminaires et État de l'art

2.1 Processus Décisionnels de Markov Partiellement Observables (POMDP)

Un POMDP est un tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, P, Z, R, \gamma)$ modélisant un problème de décision séquentielle où l'état de l'environnement n'est pas directement observable. Il est défini par un ensemble fini d'états $\mathcal{S}$, d'actions $\mathcal{A}$ et d'observations $\mathcal{O}$. La dynamique est régie par la fonction de transition $P(s'|s,a)$, la fonction d'observation $Z(o|s',a)$ et la fonction de récompense $R(s,a)$. L'agent cherche à maximiser la somme espérée de ses récompenses futures, pondérées par un facteur d'escompte $\gamma \in [0,1)$.
Puisque l'état s est caché, l'agent maintient un état de croyance (*belief state*) $b \in \mathcal{B}$, une distribution de probabilité sur $\mathcal{S}$ mise à jour par filtrage bayésien à chaque transition. Le POMDP est ainsi reformulé en un *Belief-MDP* continu. L'opérateur de Bellman, noté $\mathcal{T}_{\mathcal{B}}$ est un opérateur de mise à jour de la fonction de valeur :

$$(\mathcal{T}_{\mathcal{B}}V)(b) = \max_{a\in\mathcal{A}} \left[r(b,a) + \gamma \sum_{o\in\mathcal{O}} p(o|b,a)V(b_{a,o}) \right], \tag{1}$$

où $r(b,a)$ est la récompense espérée sous la croyance b, $p(o|b,a)$ est la probabilité de l'observation o, et $b_{a,o}$ est la croyance actualisée après avoir fait a et observé o. La fonction de valeur optimale V^* est le point fixe de cet opérateur, i.e. $V^* = \mathcal{T}_{\mathcal{B}}V^*$. Un résultat fondamental de la théorie des POMDP stipule que pour un horizon fini, $V^*(b)$ est une fonction convexe et linéaire par morceaux (PWLC), représentable par l'enveloppe supérieure d'un ensemble de vecteurs appelés α-vecteurs.

2.2 Planification Point-Based et ses Limites

L'algorithme exact d'itération de la valeur souffre d'une explosion combinatoire, le nombre d'hyperplans augmentant de manière exponentielle à chaque étape. L'algorithme Point-Based Value Iteration (PBVI) [14] contourne ce problème en restreignant l'application de l'opérateur de Bellman à un ensemble fini de points de croyance $\mathcal{B}_{pts}$ atteignables depuis la croyance initiale.
PBVI garantit une convergence avec une borne d'erreur conditionnée par la densité ϵ_{B} de l'échantillonnage $\mathcal{B}_{\text{pts}} \subset \mathcal{B}$. Cependant, la méthode souffre toujours de la malédiction de la dimensionnalité, qui se manifeste par un double goulet d'étranglement : la mémoire et le temps de calcul. D'une part, l'empreinte mémoire sature rapidement, car la dimension de chaque α-vecteur est égale au nombre d'états physiques $|\ \mathcal{S}\ |$ et la taille de l'ensemble d'hyperplans maintenu reste proportionnelle à $|\ \mathcal{B}_{\text{pts}}\ |$. D'autre part, le coût computationnel d'une mise à jour ponctuelle (*backup*) croît dramatiquement avec la taille du problème. Le calcul exact nécessitant de projeter les vecteurs à travers les matrices de transition P et d'observation Z induit une complexité temporelle de l'ordre de $O(|\ \mathcal{S}\ |^2 \cdot |\ \mathcal{A}\ | \cdot |\ \mathcal{O}\ | \cdot |\ \mathcal{B}_{\text{pts}}\ |)$ à chaque itération. Bien que des algorithmes ultérieurs (comme SARSOP [8] ou HSVI [19]) aient optimisé les heuristiques de sélection des points pour limiter la croissance de l'ensemble $\mathcal{B}_{\text{pts}}$, ils reposent sur cette même mécanique algébrique exacte. Ils finissent inévitablement par s'effondrer face à des environnements de grande dimension, limitant de facto leur application à des espaces d'états de taille modeste.

2.3 Apprentissage par Renforcement Profond et Croyances

Pour surmonter ces limites computationnelles, la littérature s'est donc naturellement tournée vers l'utilisation d'approximateurs continus paramétrés par des réseaux de neurones. Des travaux comme *Neural Value Iteration* [23] démontrent empiriquement que des réseaux de neurones peuvent approximer la propriété PWLC pour résoudre des POMDP massifs hors d'atteinte des méthodes exactes. De

même, des algorithmes de planification en ligne comme BetaZero [13] utilisent des réseaux profonds comme estimateurs de valeur pour guider la recherche arborescente.

En parallèle, des approches Model-Free et hybrides tentent d'exploiter la structure des POMDP. L'algorithme "Convex is back" [10] applique l'apprentissage par différence temporelle (TD-Learning) [21] sur le Belief-MDP en utilisant des architectures Dueling DQN [22] modifiées pour encourager la convexité par l'ajout d'un objectif d'apprentissage, ou en le forçant par l'utilisation d'un ICNN [1]. Toutefois, cette méthode échantillonne l'opérateur de Bellman via un *replay buffer* au lieu d'effectuer un *backup* complet via le modèle du problème. En abandonnant la planification itérative stricte, ces approches perdent les garanties théoriques de convergence globale. De plus, sans contrainte formelle sur la régularité du réseau (sa constante de Lipschitz), elles s'exposent à de graves instabilités lors de l'apprentissage.

2.4 Réseaux de Neurones Convexes et Régularisation Lipschitzienne

Les Réseaux de Neurones Convexes par Entrée (ICNN) [1] imposent une architecture où les poids de connexion entre les couches cachées sont strictement non-négatifs, garantissant que la sortie du réseau est une fonction convexe de ses entrées. Théoriquement, [4] ont prouvé théoriquement que les ICNN avec des fonctions d'activation ReLU étaient au moins aussi expressifs qu'une fonction PWLC pour un même nombre de paramètres donné, avec une capacité à représenter une classe de fonction PWLC avec exponentiellement moins de paramètres qu'une énumération exhaustive d'hyperplans, justifiant leur supériorité structurelle sur les α-vecteurs. Cependant, la contrainte de positivité des poids dans le papier original provoque de grandes instabilités à l'entrainement, pouvant mener à des problèmes récurrents de disparition des gradients [9]. Nous adoptons donc l'approche par reparamétrisation *Softplus* de [11] pour optimiser ces poids sans *clipping* destructeur.

Enfin, garantir la stabilité lors des mises à jour récursives de la fonction de valeur (le *bootstrapping*, où la cible d'apprentissage dépend des propres prédictions temporelles du réseau) requiert une régularisation stricte. En Deep RL, [6] ont démontré que borner la constante de Lipschitz des réseaux *Critic* via la normalisation spectrale prévient la divergence due à la surestimation temporelle, permettant par la même occasion d'entraîner des réseaux significativement plus profonds [2]. Néanmoins, la normalisation spectrale par couche s'avère souvent trop rigide et bride l'expressivité locale. Inspirés par les travaux récents en contrôle continu [17], notre approche utilise l'algorithme AutoLip [16] couplé à une projection différentiable pour contraindre globalement la constante de Lipschitz de l'ICNN, préservant ainsi la contraction de l'opérateur de Bellman tout en conservant une optimisation fluide.

3 Méthodologie : ICNN-PBVI

Notre approche, **ICNN-PBVI**, fusionne la rigueur de la planification *model-based* de PBVI avec la puissance de représentation et la scalabilité des réseaux de neurones profonds. L'idée fondamentale est de conserver la structure algorithmique globale de PBVI (alternance entre l'amélioration de la valeur sur un ensemble de points et l'expansion de cet ensemble), mais de remplacer la représentation PWLC par un approximateur de fonction neuronal V_θ qui garantit la convexité par construction : un ICNN [1].

Au lieu de calculer un nouvel ensemble d'hyperplans Γ_{k+1} à chaque itération, nous entraînons un réseau de neurones $V_{\theta_{k+1}}$ pour qu'il approxime l'application de l'opérateur de Bellman $\mathcal{T}_\mathcal{B}$ sur le réseau de l'itération précédente V_{θ_k}.

3.1 L'Opérateur de Backup Neuronal

Le cœur de l'Itération de la Valeur est l'assignation récursive $V_{k+1} \leftarrow \mathcal{T}_\mathcal{B} V_k$. Dans notre cadre continu, nous définissons ce *backup* comme un problème de régression supervisée. Pour stabiliser l'apprentissage, à l'instar des algorithmes comme DQN [12] [22], nous utilisons deux réseaux distincts : un réseau en ligne (*online network*) V_θ que nous entraînons, et un réseau cible (*target network*) V_{θ^-} qui représente la fonction de valeur de l'itération précédente et reste figé durant la phase d'optimisation.

Pour chaque point de croyance $b \in \mathcal{B}_{\text{pts}}$, nous définissons la **cible de Bellman** y_b :

$$y_b = (\mathcal{T}_\mathcal{B} V_{\theta^-})(b). \tag{2}$$

L'objectif est d'ajuster les paramètres θ du réseau en ligne en minimisant une fonction de perte (typiquement l'erreur quadratique moyenne - MSE, ou la perte de Huber) sur l'ensemble des points échantillonnés :

$$\mathcal{L}(\theta) = \mathbb{E}_{b \sim \mathcal{B}_{\text{pts}}} \left[l\left(y_b, V_\theta(b)\right) \right]. \tag{3}$$

3.2 Architecture GL-ICNN : Reparamétrisation et Contrainte Globale

Pour que l'opérateur d'ajustement empirique par descente de gradient converge et respecte la théorie de la γ-contraction, la fonction apprise doit être strictement lipschitzienne. Nous proposons une architecture modifiée, le GL-ICNN (*Globally Lipschitz ICNN*), qui assure à la fois la convexité stricte et le contrôle de la constante de Lipschitz.

Architecture de base (FICNN) : Puisque notre fonction de valeur V doit être convexe par rapport à l'entrée entière (l'état de croyance b), nous utilisons un *Fully Input-Convex* (FICNN) de [1], par opposition à la variante *Partially Input-Convex* (PICNN) où seule une partie des variables d'entrée doit induire la convexité. Un FICNN à k couches calcule une fonction scalaire $f_\theta(y)$ à partir de son entrée y (dans notre cas, l'état de croyance b) via la récurrence suivante :

$$z_{i+1} = g_i \left(W_i^{(z)} z_i + W_i^{(y)} y + c_i \right), \ \forall i \in [\![0, k-1]\!], \tag{4}$$

avec $z_0 \equiv 0$ et $f_\theta(y) = z_k$. Pour garantir que la fonction finale soit convexe par rapport à y, les fonctions d'activation g_i (telles que ReLU) doivent être convexes et non-décroissantes, et les poids récurrents $W_i^{(z)}$ agissant sur les activations z_i doivent être contraints à la non-négativité ($W_i^{(z)} \geq 0$). Les poids d'entrée $W_i^{(y)}$ (les *skip-connections*) ne sont soumis à aucune restriction de signe.

Les vecteurs z_i correspondent aux activations de la i-ème couche cachée, avec $z_0 \equiv 0$ et $z_k = f_\theta(y)$ la sortie scalaire du réseau.

Convexité par Reparamétrisation (Softplus) : L'architecture standard des ICNN exige que les poids agissant sur les activations des couches cachées, notés $W^{(z)}$, soient non-négatifs [1]. Imposer cette contrainte par projection brutale (*clipping*) empêche l'optimisation de s'effectuer correctement et crée des instabilités à l'entrainement et des problèmes de convergence. Suivant [11], nous utilisons une reparamétrisation lisse. Les paramètres libres $\phi^{(z)} \in \mathbb{R}$ sont optimisés sans contrainte, et les poids effectifs du réseau sont calculés via la fonction *Softplus* :

$$W^{(z)} = \text{Softplus}(\phi^{(z)}) = \ln(1 + \exp(\phi^{(z)})). \quad (5)$$

Cette transformation maintient $W^{(z)} > 0$ tout en restant différentiable, évitant ainsi le blocage de l'optimiseur.

Contrainte Lipschitzienne Globale (GL-ICNN) : Lorsqu'il est couplé à un approximateur neuronal, l'opérateur de Bellman ne garantit une convergence stable que si la fonction d'approximation possède une pente strictement limitée. Pour forcer une constante de Lipschitz bornée par la limite théorique $L_{\text{target}} = \frac{R_{\max} - R_{\min}}{1-\gamma}$, nous intégrons une normalisation globale directement dans la *forward pass* du réseau.

Soit $f_{\text{raw},\theta}(b)$ la sortie brute de l'ICNN et $\hat{L}_\theta$ la borne supérieure de sa constante de Lipschitz locale calculée dynamiquement par l'algorithme AutoLip [16] (voir Annexe A). Pour éviter la disparition totale des gradients lorsque la contrainte est respectée, nous lissons l'opération de projection avec une fonction *SmoothMax* contrôlée par une température τ :

$$V_\theta(b) = \frac{f_{\text{raw},\theta}(b)}{\text{SmoothMax}_\tau\left(1, \frac{\hat{L}_\theta}{L_{\text{target}}}\right)}, \quad (6)$$

où $\text{SmoothMax}_\tau(1, x) = \tau \ln\left(e^{\frac{1}{\tau}} + e^{\frac{x}{\tau}}\right)$. Cette opération assure rigoureusement que si le réseau tend à dépasser L_{target}, sa sortie est redimensionnée à la baisse, préservant ainsi la borne maximale théorique.

3.3 Conditionnement Numérique : Rescaling des Cibles

L'amplitude des récompenses d'un POMDP peut causer de violents décalages d'échelle (scale shifts), particulièrement dommageables pour les ICNN dont les poids asymétriques amplifient la variance [9]. S'inspirant des techniques de stabilisation en RL profond [15], nous appliquons un redimensionnement des valeurs cibles.

Plutôt que d'apprendre directement V^*, le réseau apprend une fonction normalisée $\widetilde{V}^* \in [-1, 1]$. En définissant le centre $R_{\text{shift}} = \frac{R_{\max} + R_{\min}}{2}$ et l'amplitude maximale théorique $V_{\text{scale}} = \frac{R_{\max} - R_{\min}}{2(1-\gamma)}$, nous transformons les récompenses du modèle :

$$\widetilde{R}(s, a) = \frac{R(s, a) - R_{\text{shift}}}{V_{\text{scale}}}. \quad (7)$$

Ce *rescaling* affine possède un atout architectural majeur pour le GL-ICNN : dans cet espace normalisé, la constante de Lipschitz cible devient unitaire ($L_{\text{target}} = 1$). L'ICNN doit simplement apprendre des pentes douces, s'accordant parfaitement avec l'initialisation standard des paramètres autour de zéro. De plus, comme l'ont démontré des architectures récentes telles que DreamerV3 [7] via la transformation *symlog*, normaliser l'échelle des valeurs cibles rend l'algorithme d'apprentissage intrinsèquement plus robuste au choix des hyperparamètres, et ce indépendamment de la dynamique des récompenses de l'environnement.

3.4 L'Algorithme ICNN-PBVI

L'algorithme global (voir Algorithme 1) s'articule autour de deux phases répétées : IMPROVE (ajustement du GL-ICNN aux cibles de Bellman) et EXPAND (échantillonnage de nouveaux points de croyance pour densifier l'espace d'approximation). Durant IMPROVE, la perte est minimisée itérativement avec le réseau cible figé, jusqu'à ce que la distance maximale entre le nouveau réseau et l'ancien (la norme L_∞ empirique) descende sous un seuil de tolérance ϵ_{vi}, marquant l'atteinte du point fixe.

4 Analyse de Convergence Théorique

Dans cette section, nous établissons une borne d'erreur généralisée pour l'itération de la valeur approximée sur l'espace des croyances, et nous démontrons comment cette borne s'applique à PBVI et à notre algorithme ICNN-PBVI.

4.1 Lemmes Intermédiaires

L'analyse repose sur la relation entre les différentes normes vectorielles. Pour tout vecteur v, nous avons la hiérarchie suivante : $\|v\|_\infty \leq \|v\|_p \leq \|v\|_1$. Dans le cadre des POMDP, la fonction de récompense $R(s, a)$ est bornée par $R_{\max}$ et $R_{\min}$. La fonction de récompense espérée sur la croyance, $r(b, a) = \sum_s b(s)R(s, a)$, admet une constante de Lipschitz $L_R \leq R_{\max} - R_{\min}$ par rapport à la norme L_1. À partir de cette propriété, nous énonçons deux lemmes sur l'opérateur de Bellman $\mathcal{T}_\mathcal{B}$. Ces résultats sont des adaptations de propriétés classiques [14] au cadre continu sur l'espace des croyances. Leurs preuves complètes sont reportées en annexe B par souci de complétude.

Lemme 4.1. *Lipschitz par opérateur de Bellman*
Soit $\hat{V}$ une fonction $L_{\hat{V}}$-Lipschitzienne sur l'espace des croyances $\mathcal{B}$. Alors, la fonction résultant de l'application de l'opérateur de Bellman, $\mathcal{T}_\mathcal{B}\hat{V}$, est une fonction $(L_R + \gamma L_{\hat{V}})$-Lipschitzienne.

Lemme 4.2. *γ-contraction de Bellman*
L'opérateur de Bellman $\mathcal{T}_\mathcal{B}$ est une γ-contraction en norme L_∞. Pour toutes fonctions de valeur U, V sur $\mathcal{B}$:

$$\|\mathcal{T}_\mathcal{B}U - \mathcal{T}_\mathcal{B}V\|_\infty \leq \gamma \|U - V\|_\infty$$

.

4.2 Borne d'Erreur Généralisée

Pour formaliser l'approximation neuronale, nous définissons un opérateur d'ajustement empirique $\Pi_{\mathcal{B}_{\text{pts}}}$. Cet opéra-

Algorithm 1 ICNN-PBVI : Squelette de l'algorithme

Require: Modèle POMDP, $N_{\text{expansions}}$ (nombre d'itérations), ϵ_{vi} (seuil de convergence), ϵ_{fit} (seuil de *loss*)

Initialiser les poids θ du GL-ICNN (réseau en ligne) et $\theta^- \leftarrow \theta$ (réseau cible)
Initialiser l'ensemble de croyances $\mathcal{B}_{\text{pts}} \leftarrow \{b_0\}$
for $k = 1$ **to** $N_{\text{expansions}}$ **do**
    $\Delta_{vi} \leftarrow \infty$ ▷ **Phase 1 :** Improve (Convergence de l'itération de la valeur)
    while $\Delta_{vi} > \epsilon_{vi}$ **do**
        $y_b \leftarrow (\mathcal{T}_{\mathcal{B}} V_{\theta^-})(b), \quad \forall b \in \mathcal{B}_{\text{pts}}$ ▷ Générer cibles de Bellman avec le modèle exact
        repeat ▷ Boucle d'optimisation (Descente de gradient)
            $\mathcal{L}(\theta) \leftarrow \mathbb{E}_{b \in \mathcal{B}_{\text{pts}}}[l(y_b, V_\theta(b))]$
            $\theta \leftarrow \text{OptimizerStep}(\theta, \nabla_\theta \mathcal{L}(\theta))$
        until $\mathcal{L}(\theta) < \epsilon_{fit}$ **ou** détection d'un plateau
        $\Delta_{vi} \leftarrow \max_{b \in \mathcal{B}_{\text{pts}}} |V_\theta(b) - V_{\theta^-}(b)|$ ▷ Vérification de la contraction
        $\theta^- \leftarrow \theta$ ▷ Mise à jour du réseau cible ($V_k \leftarrow V_{k+1}$)
    end while
    $\mathcal{B}_{\text{new}} \leftarrow \text{ExpandSampling}(\mathcal{B}_{\text{pts}}, V_\theta)$ ▷ **Phase 2 :** Expand (Recherche de nouveaux points)
    $\mathcal{B}_{\text{pts}} \leftarrow \mathcal{B}_{\text{pts}} \cup \mathcal{B}_{\text{new}}$
end for
return V_θ

teur prend une fonction initiale $\hat{V}_t$ issue d'un espace d'hypothèses $\mathcal{H}$ (ex : la classe des fonctions représentables par un GL-ICNN) et retourne une nouvelle fonction $\hat{V}_{t+1}$ visant à minimiser l'erreur maximale sur l'ensemble fini de points $\mathcal{B}_{\text{pts}}$.

Nous définissons l'erreur de fitting résiduelle $\epsilon^{\text{fit}}_{\mathcal{B}_{\text{pts}}}$ comme l'écart maximal sur l'ensemble $\mathcal{B}_{\text{pts}}$ entre la fonction retournée par l'opérateur d'ajustement et la cible exacte de Bellman :

$$\epsilon^{\text{fit}}_{\mathcal{B}_{\text{pts}}} = \max_{b \in \mathcal{B}_{\text{pts}}} |\Pi_{\mathcal{B}_{\text{pts}}}(\hat{V}_t, \mathcal{T}_{\mathcal{B}} \hat{V}_t)(b) - (\mathcal{T}_{\mathcal{B}} \hat{V}_t)(b)|. \quad (8)$$

Cette erreur englobe à la fois l'erreur d'approximation de l'espace $\mathcal{H}$ et la sous-optimalité de l'optimisation locale.

Théorème 4.3. *Borne d'erreur généralisée pour l'itération de valeur approximée*

Soit $\mathcal{B}_{pts} \subset \mathcal{B}$ un échantillonnage de l'espace des croyances, de densité $\epsilon_B = \max_{b \in \mathcal{B}} \min_{b' \in \mathcal{B}_{pts}} \|b - b'\|_1$.

Soit $\Pi_{\mathcal{B}_{pts}}$ l'opérateur d'ajustement empirique sur des fonctions $L_{\hat{V}}$-Lipschitziennes. Supposons que l'algorithme ait convergé vers une fonction de valeur $\hat{V}$ formant un point fixe sous notre procédure d'optimisation locale, telle que $\hat{V} = \Pi_{\mathcal{B}_{pts}}(\hat{V}, \mathcal{T}_{\mathcal{B}} \hat{V})$. L'erreur totale par rapport à la fonction de valeur optimale V^* est bornée sur $\mathcal{B}$ par :

$$\begin{aligned} \left\| \hat{V} - V^* \right\|_\infty &\leq \frac{\epsilon^{fit}_{\mathcal{B}_{pts}}}{1-\gamma} + \frac{L_R + (1+\gamma) L_{\hat{V}}}{1-\gamma} \epsilon_B \\ &\leq \frac{\epsilon^{fit}_{\mathcal{B}_{pts}}}{1-\gamma} + \frac{(R_{max} - R_{min}) + (1+\gamma) L_{\hat{V}}}{1-\gamma} \epsilon_B. \end{aligned}$$

Démonstration. On cherche à borner $\epsilon_{\text{tot}} = \left\| \hat{V} - V^* \right\|_\infty$ (sur $\mathcal{B}$). On commence par l'ajout de zéro suivi d'une inégalité triangulaire :

$$\begin{aligned} \epsilon_{\text{tot}} &= \left\| \hat{V} - V^* \right\|_\infty \\ &= \left\| \hat{V} - \mathcal{T}_{\mathcal{B}} \hat{V} + \mathcal{T}_{\mathcal{B}} \hat{V} - V^* \right\|_\infty \\ &\leq \left\| \hat{V} - \mathcal{T}_{\mathcal{B}} \hat{V} \right\|_\infty + \left\| \mathcal{T}_{\mathcal{B}} \hat{V} - V^* \right\|_\infty \end{aligned}$$

Or, par la définition de la fonction de valeur optimale $V^* = \mathcal{T}_{\mathcal{B}} V^*$. On peut donc utiliser la propriété de γ-contraction de l'opérateur $\mathcal{T}_{\mathcal{B}}$ démontrée précédemment :

$$\begin{aligned} \left\| \mathcal{T}_{\mathcal{B}} \hat{V} - V^* \right\|_\infty &= \left\| \mathcal{T}_{\mathcal{B}} \hat{V} - \mathcal{T}_{\mathcal{B}} V^* \right\|_\infty \\ &\leq \gamma \cdot \left\| \hat{V} - V^* \right\|_\infty \\ &= \gamma \cdot \epsilon_{\text{tot}} \end{aligned}$$

On peut donc simplifier ϵ_{tot} :

$$\begin{aligned} \epsilon_{\text{tot}} &\leq \left\| \hat{V} - \mathcal{T}_{\mathcal{B}} \hat{V} \right\|_\infty + \gamma \cdot \epsilon_{\text{tot}} \\ \iff \epsilon_{\text{tot}} &\leq \frac{\left\| \hat{V} - \mathcal{T}_{\mathcal{B}} \hat{V} \right\|_\infty}{1-\gamma} \end{aligned}$$

On regarde $\left\| \hat{V} - \mathcal{T}_{\mathcal{B}} \hat{V} \right\|_\infty$. Soit b et $\hat{b}$ tel que :

$$\begin{aligned} b &= \arg\max_{b' \in \mathcal{B}} |\hat{V}(b') - \mathcal{T}_{\mathcal{B}} \hat{V}(b')| \\ \hat{b} &= \arg\min_{b' \in \mathcal{B}_{\text{pts}}} \|b - b'\|_1 \end{aligned}$$

Par ajouts successifs de zéros et inégalité triangulaire sur $A = \left\| \hat{V} - \mathcal{T}_{\mathcal{B}} \hat{V} \right\|_\infty$:

$$
\begin{aligned}
A &= |\hat{V}(b) - \mathcal{T}_{\mathcal{B}}\hat{V}(b)| \\
&= |\hat{V}(b) - \mathcal{T}_{\mathcal{B}}\hat{V}(b) + \hat{V}(\hat{b}) - \hat{V}(\hat{b})| \\
&\leq |\hat{V}(b) - \hat{V}(\hat{b})| + |\hat{V}(\hat{b}) - \mathcal{T}_{\mathcal{B}}\hat{V}(b)| \\
&= |\hat{V}(b) - \hat{V}(\hat{b})| \\
&\quad + |\hat{V}(\hat{b}) - \mathcal{T}_{\mathcal{B}}\hat{V}(b) + \mathcal{T}_{\mathcal{B}}\hat{V}(\hat{b}) - \mathcal{T}_{\mathcal{B}}\hat{V}(\hat{b})| \\
&\leq |\hat{V}(b) - \hat{V}(\hat{b})| + |-\mathcal{T}_{\mathcal{B}}\hat{V}(b) + \mathcal{T}_{\mathcal{B}}\hat{V}(\hat{b})| \\
&\quad + |\hat{V}(\hat{b}) - \mathcal{T}_{\mathcal{B}}\hat{V}(\hat{b})| \\
&= \underbrace{|\hat{V}(b) - \hat{V}(\hat{b})|}_{(i)} + \underbrace{|\mathcal{T}_{\mathcal{B}}\hat{V}(b) - \mathcal{T}_{\mathcal{B}}\hat{V}(\hat{b})|}_{(ii)} \\
&\quad + \underbrace{|\hat{V}(\hat{b}) - \mathcal{T}_{\mathcal{B}}\hat{V}(\hat{b})|}_{(iii)}
\end{aligned}
$$

On borne les 3 termes indépendamment :

(i) On sait que $\hat{V}$ est $L_{\hat{V}}$-Lipschitzienne et $\left\|b - \hat{b}\right\|_1 \leq \epsilon_{\text{B}}$, alors :

$$
\begin{aligned}
|\hat{V}(b) - \hat{V}(\hat{b})| &\leq L_{\hat{V}} \cdot \left\|b - \hat{b}\right\|_1 \\
&\leq L_{\hat{V}} \cdot \epsilon_{\text{B}}
\end{aligned}
$$

(ii) On sait que $\mathcal{T}_{\mathcal{B}}\hat{V}$ est $(L_R + \gamma L_{\hat{V}})$-Lipschitzienne et $\left\|b - \hat{b}\right\|_1 \leq \epsilon_{\text{B}}$, on a donc :

$$
\begin{aligned}
|\mathcal{T}_{\mathcal{B}}\hat{V}(b) - \mathcal{T}_{\mathcal{B}}\hat{V}(\hat{b})| &\leq (L_R + \gamma L_{\hat{V}}) \cdot \left\|b - \hat{b}\right\|_1 \\
&\leq (L_R + \gamma L_{\hat{V}}) \cdot \epsilon_{\text{B}}
\end{aligned}
$$

(iii) Par l'hypothèse l'algorithme a convergé vers un point fixe $\hat{V} = \Pi_{\mathcal{B}_{\text{pts}}}(\hat{V}, \mathcal{T}_{\mathcal{B}}\hat{V})$. Or $\hat{b} \in \mathcal{B}_{\text{pts}}$, on peut donc borner la différence par l'erreur maximale de fitting $B = |\hat{V}(\hat{b}) - (\mathcal{T}_{\mathcal{B}}\hat{V})(\hat{b})|$:

$$
\begin{aligned}
B &= |\Pi_{\mathcal{B}_{\text{pts}}}(\hat{V}, \mathcal{T}_{\mathcal{B}}\hat{V})(\hat{b}) - (\mathcal{T}_{\mathcal{B}}\hat{V})(\hat{b})| \\
&\leq \max_{b' \in \mathcal{B}_{\text{pts}}} |\Pi_{\mathcal{B}_{\text{pts}}}(\hat{V}, \mathcal{T}_{\mathcal{B}}\hat{V})(b') - (\mathcal{T}_{\mathcal{B}}\hat{V})(b')| \\
&= \epsilon^{\text{fit}}_{\mathcal{B}_{\text{pts}}}
\end{aligned}
$$

Ce qui donne donc :

$$
\epsilon_{\text{tot}} \leq \frac{\epsilon_{\text{B}} \cdot ((1+\gamma)L_{\hat{V}} + L_R) + \epsilon^{\text{fit}}_{\mathcal{B}_{\text{pts}}}}{1-\gamma}
$$

□

Remarque : Cette borne met en évidence les trois leviers pour minimiser l'erreur totale :

— **Densifier l'échantillonnage ($\epsilon_{\mathbf{B}} \downarrow$)** : Augmenter le nombre de points de croyance réduit mécaniquement le terme d'erreur lié à l'échantillonage. Néanmoins, un ensemble $\mathcal{B}_{\text{pts}}$ plus gros peut complexifier le *fitting*.

— **Régulariser $\hat{V}$ ($L_{\hat{V}} \downarrow$)** : Réduire la constante de Lipschitz limite l'amplification des erreurs. Toutefois, une régularisation excessive peut brider la capacité d'expression du modèle, i.e. peut augmenter $\epsilon^{\text{fit}}_{\mathcal{B}_{\text{pts}}}$.

— **Capacité de représentation ($\epsilon^{\text{fit}}_{\mathcal{B}_{\text{pts}}} \downarrow$)** : Un approximateur plus riche permet de réduire l'erreur d'ajustement aux cibles de Bellman. Peut s'obtenir au prix d'une optimisation plus difficile, soit une convergence plus lente vers le point fixe de Bellman.

4.3 Application aux algorithmes (PBVI et ICNN-PBVI)

Les preuves de ces corollaires sont en annexes C.

Corollaire 4.3.1. *Application sur PBVI*
L'opérateur de backup de PBVI étant de nature analytique exacte sur $\mathcal{B}_{pts}$, on a $\epsilon^{fit}_{\mathcal{B}_{pts}} = 0$. En sachant que la constante de Lipschitz des hyperplans de PBVI est bornée par $\frac{R_{max} - R_{min}}{1-\gamma}$, l'application du théorème donne :

$$
\left\|\hat{V} - V^*\right\|_\infty \leq \frac{2 \cdot (R_{max} - R_{min})}{(1-\gamma)^2} \cdot \epsilon_B. \tag{9}
$$

Ce qui correspond, à un facteur constant près, à la borne historique établie par [14].

Corollaire 4.3.2. *Application sur ICNN-PBVI*
Considérons l'algorithme ICNN-PBVI utilisant un réseau GL-ICNN. Par construction, le GL-ICNN garantit strictement que la constante de Lipschitz du réseau est bornée par $L_{target} = \frac{R_{max} - R_{min}}{1-\gamma}$. En remplaçant $L_{\hat{V}}$ et L_R on a pour pour ICNN-PBVI :

$$
\left\|\hat{V} - V^*\right\|_\infty \leq \frac{\epsilon^{fit}_{\mathcal{B}_{pts}}}{1-\gamma} + \frac{2 \cdot (R_{max} - R_{min})}{(1-\gamma)^2} \cdot \epsilon_B. \tag{10}
$$

Ce corollaire montre que sous réserve que l'optimisation neuronale minimise efficacement l'erreur résiduelle ($\epsilon^{\text{fit}}_{\mathcal{B}_{\text{pts}}} \approx 0$), ICNN-PBVI bénéficie des mêmes garanties d'erreur pire-cas que PBVI, tout en s'affranchissant de la contrainte du maintien couteux de l'ensemble des hyperplans.

5 Conclusion et Perspectives

Cet article aborde le fossé entre la planification formelle (POMDP) et l'Apprentissage par Renforcement Profond. Les méthodes exactes telles que PBVI offrent de fortes garanties mais explosent en complexité, tandis que les approches neuronales passent à l'échelle au détriment de la rigueur mathématique. Pour réconcilier ces deux paradigmes, nous avons introduit ICNN-PBVI et l'architecture GL-ICNN. En couplant la reparamétrisation Softplus pour une stricte convexité et la régularisation SmoothMax sur AutoLip pour une contrainte Lipschitzienne globale, notre réseau permet d'approximer stablement l'opérateur de Bellman. Notre contribution théorique majeure réside dans la preuve formelle que cette hybridation conserve une borne d'erreur généralisée compétitive face à PBVI.
À court terme, notre perspective principale est la validation empirique de ces garanties théoriques. Pour ce faire, nous développons actuellement une architecture logicielle vectorisée sous JAX [3], permettant d'optimiser à la fois notre méthode et les baselines en exploitant l'accélération matérielle. Les premières expérimentations sur des environnements canoniques de petite dimension (comme le problème

Tiger) valident d'ores et déjà la viabilité de notre contrainte Lipschitzienne en pratique. Nos futures expériences à plus grande échelle visent à répondre à quatre grandes questions de recherche : la convergence empirique vers la borne théorique démontrée, la stabilité de l'optimisation lors du bootstrapping, l'expressivité de l'ICNN en grande dimension, et sa compétitivité face aux algorithmes de Deep RL qui exploitent la convexité et ceux qui ne l'exploitent pas.
Enfin, à plus long terme, l'extension d'ICNN-PBVI au cadre multi-agents (Dec-POMDP), domaine s'appuyant également sur des méthodes *point-based* et la convexité de la valeur centralisée [5], constitue une voie de recherche extrêmement prometteuse.

Remerciements

Ce travail s'inscrit dans le cadre du Master 2 MINMACS de Normandie Université.
Ces travaux ont également bénéficié de discussions enrichissantes avec les membres des équipes MAD et IMAGE du GREYC.

Références

[1] Brandon Amos, Lei Xu, and J. Zico Kolter. Input convex neural networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 146–155. PMLR, 06–11 Aug 2017.

[2] Johan Bjorck, Carla P. Gomes, and Kilian Q. Weinberger. Towards deeper deep reinforcement learning with spectral normalization. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2021. Curran Associates Inc.

[3] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX : composable transformations of Python+NumPy programs, 2018.

[4] Yize Chen, Yuanyuan Shi, and Baosen Zhang. Optimal control via neural networks : A convex approach. In *International Conference on Learning Representations*, 2019.

[5] Jilles Steeve Dibangoye, Christopher Amato, Olivier Buffet, and François Charpillet. Optimally solving Dec-POMDPs as continuous-state MDPs. *Journal of Artificial Intelligence Research*, 55 :443–497, 2016.

[6] Florin Gogianu, Tudor Berariu, Mihaela C Rosca, Claudia Clopath, Lucian Busoniu, and Razvan Pascanu. Spectral normalisation for deep reinforcement learning : An optimisation perspective. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 3734–3744. PMLR, 18–24 Jul 2021.

[7] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, pages 1–7, 2025.

[8] Wee Sun Lee Hanna Kurniawati, David Hsu. SARSOP : Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Proceedings of Robotics : Science and Systems IV*, Zurich, Switzerland, June 2008.

[9] Pieter-Jan Hoedt and Günter Klambauer. Principled weight initialisation for input-convex neural networks. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[10] Daniel Koutas, Daniel Hettegger, Kostas G. Papakonstantinou, and Daniel Straub. Convex Is Back : Solving Belief MDPs With Convexity-Informed Deep Reinforcement Learning. (arXiv :2502.09298), March 2025.

[11] Ashok Makkuva, Amirhossein Taghvaei, Sewoong Oh, and Jason Lee. Optimal transport mapping via input convex neural networks. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 6672–6681. PMLR, 13–18 Jul 2020.

[12] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. 2013. cite arxiv :1312.5602Comment : NIPS Deep Learning Workshop 2013.

[13] Robert J. Moss, Anthony Corso, Jef Caers, and Mykel J. Kochenderfer. BetaZero : Belief-State Planning for Long-Horizon POMDPs using Learned Approximations. In *Reinforcement Learning Conference (RLC)*, 2024.

[14] Joelle Pineau, Geoff Gordon, and Sebastian Thrun. Point-based value iteration : an anytime algorithm for POMDPs. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, IJCAI'03, page 1025–1030, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc.

[15] Tobias Pohlen, Bilal Piot, Todd Hester, Mohammad Gheshlaghi Azar, Dan Horgan, David Budden, Gabriel Barth-Maron, Hado Van Hasselt, John Quan, Mel Večerík, et al. Observe and look further : Achieving consistent performance on atari. *arXiv preprint arXiv :1805.11593*, 2018.

[16] Kevin Scaman and Aladin Virmaux. Lipschitz regularity of deep neural networks : analysis and efficient estimation. In *Proceedings of the 32nd International Conference on Neural Information Processing*

Systems, NIPS'18, page 3839–3848, Red Hook, NY, USA, 2018. Curran Associates Inc.

[17] Jaeyong Shin, Woohyun Cha, Donghyeon Kim, Junhyeok Cha, and Jaeheung Park. Spectral normalization for lipschitz-constrained policies on learning humanoid locomotion. *arXiv preprint arXiv :2504.08246*, 2025.

[18] Richard D. Smallwood and Edward J. Sondik. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, 21(5) :1071–1088, 1973.

[19] Trey Smith and Reid Simmons. Heuristic search value iteration for POMDPs. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, UAI '04, page 520–527, Arlington, Virginia, USA, 2004. AUAI Press.

[20] Edward J. Sondik. The optimal control of partially observable markov processes over the infinite horizon : Discounted costs. *Operations Research*, 26(2) :282–304, 1978.

[21] Richard Sutton. Learning to predict by the method of temporal differences. *Machine Learning*, 3 :9–44, 08 1988.

[22] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1995–2003, New York, New York, USA, 20–22 Jun 2016. PMLR.

[23] Yang You, Ufuk Çakır, Alex Schutz, and Nick Hawes. Neural value iteration. *Computing Research Repository*, arXiv :2511.08825, 11 2025.

[24] K.J Åström. Optimal control of Markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, 10(1) :174–205, 1965.

A Calcul de la constante de Lipschitz (AutoLip)

L'algorithme d'estimation de la borne de Lipschitz locale, adapté des travaux de [16] pour les réseaux ICNN, propage récursivement la constante à travers le graphe de calcul. En utilisant la norme matricielle de Frobenius $\|\cdot\|_F$ (qui borne supérieurement la norme spectrale), nous garantissons un calcul rapide et différentiable sans casser le graphe du gradient.

Algorithm 2 Estimation Lipschitz FICNN (AutoLip)

Require: Paramètres $\theta = \{W_i^{(z)}, W_i^{(y)}\}_{i=0}^{k-1}$, Norme de Frobenius $\|\cdot\|_F$
Ensure: $\hat{L}_\theta$: Borne supérieure globale de la constante de Lipschitz
1: $L_0 \leftarrow 0$ ▷ L'état caché initial $z_0 \equiv 0$
2: **for** $i = 0$ **to** $k-1$ **do**
3: $N_z \leftarrow \|W_i^{(z)}\|_F$ ▷ Norme de la matrice des connexions internes
4: $N_y \leftarrow \|W_i^{(y)}\|_F$ ▷ Norme de la matrice des connexions d'entrée (*skip-connections*)
5: $L_{i+1} \leftarrow N_z \cdot L_i + N_y$ ▷ Propagation (les activations comme ReLU ont une pente ≤ 1)
6: **end for**
7: **return** L_k

B Preuves lemmes intermédiaires

Démonstration. Lipschitz par opérateur de Bellman
Soient $b, b' \in \mathcal{B}$. Par définition de l'opérateur de Bellman :

$$
\begin{aligned}
&|(\mathcal{T}_{\mathcal{B}}\hat{V})(b) - (\mathcal{T}_{\mathcal{B}}\hat{V})(b')| \\
&= \left| \max_a \left[R(b,a) + \gamma \sum_o P(o|b,a)\hat{V}(b_{a,o}) \right] \right. \\
&\quad \left. - \max_{a'} \left[R(b',a') + \gamma \sum_o P(o|b',a')\hat{V}(b'_{a',o}) \right] \right| \\
&\leq \max_a \left| R(b,a) - R(b',a) \right. \\
&\quad \left. + \gamma \sum_o \Big(P(o|b,a)\hat{V}(b_{a,o}) - P(o|b',a)\hat{V}(b'_{a,o}) \Big) \right| \\
&\leq \max_a |R(b,a) - R(b',a)| \\
&\quad + \gamma \max_a \left| \sum_o P(o|b,a)\hat{V}(b_{a,o}) - \sum_o P(o|b',a)\hat{V}(b'_{a,o}) \right|.
\end{aligned}
$$

Le premier terme est borné par la constante de Lipschitz de la récompense : $|R(b,a) - R(b',a)| \leq L_R \|b - b'\|_1$.
Pour le second terme, on évalue la différence d'espérance sur les observations. On utilise la définition de la mise à jour bayésienne non-normalisée, qui donne : $P(o|b,a)b_{a,o}(s) = P(o|s,a)\sum_{s'} P(s|s',a)b(s')$.
On peut alors développer la somme des normes L_1 sur les observations (voir [14]) :

$$
\begin{aligned}
&\sum_o \|P(o|b,a)b_{a,o} - P(o|b',a)b'_{a,o}\|_1 \\
&= \sum_{o,s} \left| P(o|b,a)b_{a,o}(s) - P(o|b',a)b'_{a,o}(s) \right| \\
&= \sum_{o,s} P(o|s,a) \left| \sum_{s'} P(s|s',a)\big(b(s') - b'(s')\big) \right| \\
&= \sum_s \underbrace{\Big(\sum_o P(o|s,a) \Big)}_{=1} \left| \sum_{s'} P(s|s',a)\big(b(s') - b'(s')\big) \right| \\
&\leq \sum_{s,s'} P(s|s',a)\,|b(s') - b'(s')| \quad \text{(inég. triang.)} \\
&= \sum_{s'} |b(s') - b'(s')| \underbrace{\sum_s P(s|s',a)}_{=1} \\
&= \|b - b'\|_1.
\end{aligned}
$$

Puisque $\hat{V}$ est $L_{\hat{V}}$-Lipschitzienne par rapport à la norme L_1, on applique ce résultat :

$$
\begin{aligned}
&\max_a \left| \sum_o P(o|b,a)\hat{V}(b_{a,o}) - \sum_o P(o|b',a)\hat{V}(b'_{a,o}) \right| \\
&\leq \max_a \sum_o L_{\hat{V}} \|P(o|b,a)b_{a,o} - P(o|b',a)b'_{a,o}\|_1 \\
&\leq L_{\hat{V}} \|b - b'\|_1.
\end{aligned}
$$

En rassemblant le tout, on a donc : $|(\mathcal{T}_{\mathcal{B}}\hat{V})(b) - (\mathcal{T}_{\mathcal{B}}\hat{V})(b')| \leq (L_R + \gamma L_{\hat{V}})\|b - b'\|_1$. □

Démonstration. γ-contraction de l'opérateur de Bellman
Pour tout point de croyance $b \in \mathcal{B}$:

$$
\begin{aligned}
&|(\mathcal{T}_{\mathcal{B}}U)(b) - (\mathcal{T}_{\mathcal{B}}V)(b)| \\
&= \left| \max_a \left[R(b,a) + \gamma \sum_o P(o|b,a)U(b_{a,o}) \right] \right. \\
&\quad \left. - \max_{a'} \left[R(b,a') + \gamma \sum_o P(o|b,a')V(b_{a',o}) \right] \right| \\
&\leq \max_a \left| \gamma \sum_o P(o|b,a)\big(U(b_{a,o}) - V(b_{a,o})\big) \right| \\
&\leq \gamma \max_a \sum_o P(o|b,a)\, \|U - V\|_\infty \\
&= \gamma \|U - V\|_\infty \quad (\text{car } \textstyle\sum_o P(o|b,a) = 1).
\end{aligned}
$$

□

C Preuves des corollaires

Démonstration. Application du théorème sur PBVI
L'opérateur de backup de PBVI étant de nature analytique,

on a $\epsilon^{\text{fit}}_{\mathcal{B}_{\text{pts}}} = 0$. Ensuite :

$$\begin{aligned}
&(1+\gamma)L_{\hat{V}} + L_R \\
&\leq \frac{(1+\gamma)(R_{\max} - R_{\min})}{1-\gamma} + R_{\max} - R_{\min} \\
&= \frac{(1+\gamma+1-\gamma)(R_{\max} - R_{\min})}{1-\gamma} \\
&= \frac{2(R_{\max} - R_{\min})}{1-\gamma}.
\end{aligned}$$

En le réintroduisant dans l'équation de base :

$$\left\|\hat{V} - V^*\right\|_\infty \leq \frac{2(R_{\max} - R_{\min})}{(1-\gamma)^2}\epsilon_{\text{B}}.$$

□

Démonstration. Application du théorème sur ICNN-PBVI
Par définition de notre architecture GL-ICNN, nous avons la garantie stricte que $L_{\hat{V}} \leq \frac{R_{\max} - R_{\min}}{1-\gamma}$. De plus, nous avons $L_R \leq R_{\max} - R_{\min}$.
Remplaçons ces termes dans le numérateur du terme de généralisation de notre théorème de base :

$$\begin{aligned}
&(1+\gamma)L_{\hat{V}} + L_R \\
&\leq (1+\gamma)\left(\frac{R_{\max} - R_{\min}}{1-\gamma}\right) + (R_{\max} - R_{\min}) \\
&= (R_{\max} - R_{\min})\left[\frac{1+\gamma}{1-\gamma} + \frac{1-\gamma}{1-\gamma}\right] \\
&= \frac{2(R_{\max} - R_{\min})}{1-\gamma}.
\end{aligned}$$

En réintroduisant ce résultat dans l'équation de base du théorème général :

$$\begin{aligned}
\left\|\hat{V} - V^*\right\|_\infty &\leq \frac{\epsilon^{\text{fit}}_{\mathcal{B}_{\text{pts}}}}{1-\gamma} + \frac{\frac{2(R_{\max} - R_{\min})}{1-\gamma}}{1-\gamma}\epsilon_{\text{B}} \\
&= \frac{\epsilon^{\text{fit}}_{\mathcal{B}_{\text{pts}}}}{1-\gamma} + \frac{2(R_{\max} - R_{\min})}{(1-\gamma)^2}\epsilon_{\text{B}}.
\end{aligned}$$

□