LSTM Path-Maker : une stratégie à base de réseau de neurones LSTM pour la patrouille multiagent

Mehdi William Othmani-Guibourg, Amal El Fallah Seghrouchni, Jean-Loup Farges

▶ **To cite this version:**

Mehdi William Othmani-Guibourg, Amal El Fallah Seghrouchni, Jean-Loup Farges. LSTM Path-Maker : une stratégie à base de réseau de neurones LSTM pour la patrouille multiagent. Revue Ouverte d'Intelligence Artificielle, 2022, 3 (3-4), pp.345-372. 10.5802/roia.34 . hal-03649519

HAL Id: hal-03649519
https://hal.science/hal-03649519

Submitted on 27 Apr 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Revue Ouverte d'Intelligence Artificielle
Volume 3, nº 3-4, 2022, 345-372

LSTM Path-Maker: une stratégie à base de réseau de neurones LSTM pour la patrouille multiagent

Mehdi William Othmani-Guibourg[a], Amal El Fallah Seghrouchni[b], Jean-Loup Farges[a]

[a] ONERA Dept. traitement de l'information et systèmes 2 avenue Edouard Belin 31400 Toulouse, France
E-mail : william.othmani@gmail.com

[b] Ai movement - Université Mohammed 6 Polytechnique et LIP6 Sorbonne Université 75005 Paris, France
URL *:* http://digital-ia.org/
E-mail : amal.elfallah@lip6.fr, farges@onera.fr.

Résumé. — Depuis plus d'une décennie, la tâche de la patrouille multiagent a attiré l'attention de la communauté multiagent de manière croissante, notamment en raison de son grand nombre d'applications potentielles. Cependant, les algorithmes à base de méthodes d'apprentissage profond pour traiter cette tâche ont jusqu'à présent peu été développés. Dans cet article nous proposons d'intégrer un réseau de neurones récurrent (RNN) à une stratégie de patrouille multiagent en introduisant un modèle formel de stratégie multiagent basée sur l'architecture de RNN LSTM que nous avons nommé *LSTM-Path-Maker*. Le réseau LSTM est entraîné sur des traces de simulation d'une stratégie coordonnée et centralisée, puis embarqué dans chaque agent en vue de patrouiller une zone le plus efficacement possible sans communication. Enfin, cette nouvelle stratégie basée sur l'architecture LSTM est évaluée en simulation et comparée à deux stratégies représentatives de référence: d'une part une stratégie coordonnée et d'autre part une stratégie réactive. Les résultats indiquent que la stratégie proposée, comparable à une stratégie réactive car non coordonnée, est meilleure que la stratégie réactive.

Mots-clés. — Systèmes multiagents, Réseaux de neurones artificiels, Réseaux récurrents à mémoire court et long terme.

1. Introduction

La détection d'événements survenant dans un environnement est nécessaire dans un grand nombre de domaines. Par exemple, dans le cadre de la sécurité civile, il peut y avoir nécessité à détecter un départ de feu dans une forêt ou des individus ayant besoin d'assistance. Dans le domaine du gardiennage ou dans un cadre militaire, il peut être nécessaire de détecter des intrus. Lorsque l'environnement ne peut pas être couvert

par un système d'observation fixe, une solution alternative est la patrouille, c'est-à-dire le déplacement dans l'environnement d'agents visant à *couvrir* chaque partie de la zone à patrouiller aussi souvent que possible. Par exemple, un drone peut être utilisé pour observer une zone géographique donnée tandis qu'il se meut. La tâche générique consistant à patrouiller peut naturellement être distribuée dans le temps et dans l'espace entre plusieurs agents. Pour les domaines évoqués précédemment, il est possible d'utiliser plusieurs drones qui doivent alors être coordonnés. De manière générale, la gestion d'un tel système correspond au problème de la patrouille multiagent [13, 14].

Une caractéristique que le problème de la patrouille multiagent partage avec d'autres problèmes complexes est la difficulté à obtenir des résultats analytiques à partir des équations qui le formulent ; en cela il peut être considéré comme un problème complexe. Il apparaît alors que la manière privilégiée de prévoir le comportement du système relatif au problème est de simuler les interactions entre les différents composants de ce système ; on parle alors de simulation à base d'agents. La qualité d'une stratégie de patrouille est concrètement évaluée en simulation à l'aide de différentes métriques, chacune mesurant une propriété spécifique de la distribution des visites générées par la stratégie. De manière informelle, il y a consensus sur le fait qu'une bonne stratégie minimise l'intervalle temporel entre deux passages au même emplacement, ceci pour tous les emplacements de la zone à patrouiller.

Ces quinze dernières années, différents types de stratégies d'agents ont été proposés pour le problème de la patrouille multiagent : des stratégies centralisées [13], émergentes [13], à base d'oisiveté [13], heuristiques prenant en compte l'oisiveté et la distance pour le choix d'un nœud cible et le calcul du chemin vers celui-ci [1], des stratégies reposant sur le calcul d'un cycle hamiltonien [9], sur des heuristiques de résolution du problème du voyageur de commerce [6], sur de l'apprentissage par renforcement [22], et même sur un mécanisme d'enchères [15]. Dans ce contexte, il est pertinent de distinguer, d'une part, les agents qui agissent sur la base de leur information personnelle des agents qui agissent sur la base d'une information partagée, et, d'autre part, les agents qui réagissent selon un comportement similaire à des réflexes, dits *réactifs*, des agents qui agissent en poursuivant un but explicite, dits *cognitifs*.

Jusqu'à ce jour peu de travaux ont abordé l'utilisation de réseaux de neurones artificiels pour l'apprentissage de comportements coopératifs dans le cadre de la patrouille multiagent. Guo *et al.* [11] se sont penchés sur la planification de chemins de couverture complète d'une zone à patrouiller par des méthodes non hiérarchiques basées sur des réseaux de neurones pour lesquels chaque neurone code une région spécifique de l'espace. Un autre exemple est l'apprentissage multiagent coopératif où chaque robot embarque un réseau de neurones connecté avec celui des autres robot. L'évolution des poids de connexion entre les différents réseaux de neurones permet alors l'apprentissage d'un langage de communication entre robots [7]. Néanmoins, aucun travail ne s'était encore intéressé à dépasser les performances des stratégies existantes en utilisant des réseaux de neurones artificiels profonds. En particulier, les capacités d'apprentissage des réseaux de neurones artificiels pourraient permettre de capturer des comportements coopératifs issus de stratégies centralisées afin de les exploiter

postérieurement dans un contexte où les agents ne peuvent pas communiquer entre eux et ainsi surpasser les stratégies sans communication entre agents de l'état de l'art.

Cet article propose donc, dans le cadre du problème de la patrouille multiagent, d'utiliser l'architecture de réseaux de neurones récurrents à *à mémoire court et long terme*, en anglais *Long Short-Term Memory* (LSTM). Les réseaux de neurones récurrents, très efficients dans l'apprentissage de séquences temporelles, sont pertinents pour ce problème dans la mesure où ils peuvent être interprétés comme des machines de prise de décision dans le temps. La stratégie proposée ici est basée sur l'architecture LSTM. Dans cette stratégie, le réseau de neurones artificiels est utilisé comme un générateur de chemins pour que les agents patrouilleurs naviguent dans un graphe aussi efficacement que possible, sans aucune communication. Chaque réseau de neurones est entraîné hors-ligne en utilisant des données générées au préalable à cette fin. Il est ensuite embarqué dans les agents qui l'utilisent pour sélectionner le prochain nœud à visiter étant donné le chemin déjà parcouru. Finalement, les performances de cette stratégie sont évaluées en utilisant des critères d'évaluation, ou métriques, classiques pour ce type de recherches.

La section 2 présente des recherches antérieures sur la patrouille multiagent, certaines architectures de réseaux neuronaux et des travaux sur l'utilisation des réseaux neuronaux pour la patrouille multiagent. Cette présentation est utile pour comprendre les développements proposés ici tout en les positionnant par rapport à l'existant. Le lecteur familier avec la patrouille multiagent et les réseaux de neurones pourra passer directement aux sections suivantes. La section 3 introduit le générateur de chemins à base de LSTM, en anglais *Random-Next-Neighbour-LSTM-Path-Maker* (RLPM), la nouvelle stratégie introduite dans le présent article. Dans la section 4 les résultats d'apprentissage sont analysés et la nouvelle stratégie évaluée. La section 5 conclut ce travail en mettant en lumière certaines limites de cette nouvelle stratégie et indique des pistes pour de futurs travaux.

2. Recherches antérieures pertinentes

Cette section vise à fournir des informations sur la patrouille multiagent et l'architecture neuronale LSTM.

2.1. La patrouille multiagent

2.1.1. *Définition formelle*

Formellement, le modèle de la patrouille multiagent repose sur une société d'agents $\mathbf{A}$, capables de se déplacer dans un environnement avec les mêmes paramètres de mobilité, et sur un graphe $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ représentant de manière abstraite une discrétisation de la zone à patrouiller. $\mathbf{V}$ est l'ensemble des nœuds, c'est-à-dire l'ensemble des lieux à visiter, il comprend N éléments et ceux-ci peuvent être identifiés par leur indice : $\mathbf{V} = \{1, \ldots, N\}$. L'ensemble des arcs de $\mathbf{G}$, noté $\mathbf{E}$, est tel que $\mathbf{E} \subset \mathbf{V}^2$. Chaque arc, $\{v, w\}$, correspond à un trajet direct possible entre l'emplacement correspondant au nœud v et celui correspondant au nœud w. Le temps de parcours de ce trajet est $c_{v,w}$.

L'état du système est caractérisé d'une part par la position des différents agents sur **G** et d'autre part par l'oisiveté des différents nœuds de **G**. L'oisiveté d'un nœud v à l'instant t, $i_t(v)$, correspond au temps écoulé depuis le dernier passage de tout agent [6]. Chaque fois qu'un agent arrive sur un nœud v, il doit décider du prochain arc à parcourir parmi les arcs $\{v, w\}$; il choisit le nœud suivant w parmi les nœuds voisins de v. Finalement, au démarrage d'une instance du problème de la patrouille multiagent, les agents de **A** sont positionnés sur certains nœuds de **G** et les oisivetés sont initialisées à 0.

2.1.2. *Les critères d'évaluation*

L'oisiveté instantanée à l'instant t du nœud v, $i_t(v)$, conduit à l'*oisiveté moyenne* (Iav), qui n'est autre que la moyenne des oisivetés instantanées pour tous les nœuds et tous les instants de la mission, et à la *pire oisiveté* (WI), également évaluée sur l'ensemble de la mission [14]. Soit τ la durée d'exécution d'une mission, on a alors :

$$\text{Iav} = \frac{1}{\tau \times |\mathbf{V}|} \sum_{t=1}^{\tau} \sum_{v \in \mathbf{V}} i_t(v) \tag{2.1}$$

et :

$$\text{WI} = \max_{t \in \{1,\dots\tau\}, v \in \mathbf{V}} \{i_t(v)\} \tag{2.2}$$

Sampaio *et al.* [21] ont défini des critères d'évaluation pertinents pour la réalisation de statistiques sur la base non pas de l'oisiveté mais sur celle plus intuitive de l'intervalle entre des visites des agents sur un nœud. Pour cette classe de critères d'évaluation, la valeur de l'intervalle d'un nœud est calculée en mémorisant la valeur de l'oisiveté juste avant le passage d'un agent. Soit pour le nœud v l'ensemble des intervalles $\mathbf{I}(v) = \{i_t(v) \mid t \in \{1, \dots \tau - 1\} \wedge i_{t+1}(v) = 0\} \cup i_\tau(v)$. Tous les intervalles de tous les nœuds sont agrégés sur la durée d'une mission pour obtenir d'une part l'*Intervalle Moyen* (IM) et d'autre part l'*Intervalle Quadratique Moyen* (IQM). IM est la moyenne des intervalles et IQM est la racine carrée de la moyenne des carrés des intervalles. On a :

$$\text{IM} = \frac{1}{\sum_{v \in \mathbf{V}} |\mathbf{I}(v)|} \sum_{v \in \mathbf{V}, x \in \mathbf{I}(v)} x \tag{2.3}$$

et :

$$\text{IQM} = \sqrt{\frac{1}{\sum_{v \in \mathbf{V}} |\mathbf{I}(v)|} \sum_{v \in \mathbf{V}, x \in \mathbf{I}(v)} x^2} \tag{2.4}$$

De par son caractère quadratique IQM illustre mieux les différences entre intervalles que ne le fait IM. Cet indicateur traduit le fait que les visites sur les nœuds sont équilibrées dans le temps et dans l'espace. Bien que IM et IQM soient des indicateurs plus intuitifs que Iav et que IQM pallie en partie aux défauts de IM, il est important de noter que WI correspond également au pire intervalle, c'est à dire à la limite quand p tend vers l'infini de la racine $p^{\text{ème}}$ de la moyenne des intervalles élevés à la puissance p. WI traduit donc mieux que IQM l'équilibre entre les visites aux noeuds.

De manière à évaluer la contribution de chaque agent lorsque la taille de $\mathbf{A}$ varie, les critères d'évaluation sont normalisés en les multipliant par le nombre d'agents N_a tel que $N_a = |\mathbf{A}|$.

2.1.3. *Les stratégies*

Une stratégie d'agents est donc un algorithme qui permet à chaque agent de prendre une décision chaque fois qu'il arrive sur un nœud. Pour le problème de la patrouille multiagent, quelle que soit la stratégie considérée, chaque agent réalise des actions sur la base de sa connaissance de l'état du système. Parmi la grande variété de stratégies, deux sont ici utilisées comme références : la stratégie réactive consciencieuse, en anglais *Conscientious Reactive* (CR) [14], et la stratégie cognitive et coordonnée avec recherche de chemin heuristique, en anglais *Heuristic Pathfinder Cognitive Coordinated* (HPCC) [1]. La stratégie à cycle unique [6] et l'utilisation des processus décisionnels de Markov partiellement observables et distribués [3] sont également présentés ici.

L'algorithme de CR consiste à sélectionner, parmi les nœuds voisins de celui où se trouve l'agent, le nœud présentant l'oisiveté la plus élevée. Pour CR les agents ne communiquent pas entre eux, chaque agent utilise donc un majorant de l'oisiveté calculé sur la base de son propre parcours et non pas l'oisiveté résultant du mouvement de l'ensemble des agents. CR peut être vu comme une stratégie représentante type des stratégies décentralisées et réactives. Elle peut donc être utilisée comme stratégie de référence pour les stratégies décentralisées.

Pour HPCC, la communication entre agents est parfaite : les oisivetés des nœuds sont calculées par un coordinateur sur la base des déplacements de tous les agents ; elles peuvent donc être considérées comme partagées par tous dans la mesure où les agents peuvent communiquer à tout instant avec le coordinateur. Lorsqu'un agent arrive sur un nœud v_0, le processus de décision permettant de sélectionner le nœud suivant w comprend deux étapes :

(1) Méthode *Heuristic* : sélection d'un nœud cible qui n'est pas nécessairement un nœud voisin de v_0 et,
(2) Méthode *Pathfinder* : calcul d'un chemin entre v_0 et ce nœud cible.

w est alors le premier nœud sur le chemin calculé. La sélection du nœud cible, via l'algorithme *Heuristic*, prend en compte non seulement son oisiveté normalisée mais aussi le temps de parcours pour y arriver en partant de v_0, également normalisé. Le temps de parcours entre les nœuds v_0 et v de $\mathbf{V}$, $d(v_0, v)$, est le plus court chemin calculé sur la base des temps de parcours des arcs reliant des nœuds voisins. Il s'agit donc du chemin le plus rapide. L'oisiveté et la durée du chemin le plus rapide sont normalisés en les réduisant à une valeur entre 0 et 1. En supposant que les oisivetés ne sont pas toutes identiques, la valeur 0 est attribuée à l'oisiveté maximale et la valeur 1 à l'oisiveté minimale. Les valeurs intermédiaires sont calculées par interpolation entre

ces deux *extrema*. L'oisiveté normalisée d'un nœud v de $\mathbf{V}$, $\bar{i_t}(v)$, est alors donnée par :

$$\bar{i_t}(v) = \frac{\max_{w \in \mathbf{V}}\{i_t(w)\} - i_t(v)}{\max_{w \in \mathbf{V}}\{i_t(w)\} - \min_{w \in \mathbf{V}}\{i_t(w)\}} \tag{2.5}$$

Cette quantité est d'autant plus petite que l'oisiveté du nœud v est élevée. L'objectif du problème de la patrouille multiagent étant de minimiser l'oisiveté des nœuds, les nœuds à oisiveté élevée devront donc être visités en premier. La durée du chemin le plus rapide est normalisée suivant le même principe. Néanmoins, afin de favoriser les nœuds les plus proches, la valeur 0 est attribuée au nœud le plus proche et la valeur 1 au nœud le moins rapidement atteignable. Pour un nœud v différent de v_0, la valeur normalisée du chemin le plus rapide entre v_0 et v, $\bar{d}(v_0, v)$, est alors donnée par :

$$\bar{d}(v_0, v) = \frac{d(v_0, v) - \min_{w \in \mathbf{V}\setminus\{\mathbf{v_0}\}}\{d(v_0, w)\}}{\max_{w \in \mathbf{V}}\{d(v_0, w)\} - \min_{w \in \mathbf{V}\setminus\{\mathbf{v_0}\}}\{d(v_0, w)\}} \tag{2.6}$$

Le nœud cible, $\hat{v}$, pour l'agent se trouvant au nœud v_0 est alors donné par minimisation d'un critère pondérant les deux variables normalisées :

$$\hat{v} = \text{argmin}\{r_H \bar{i_t}(v) + (1 - r_H)\bar{d}(v_0, v) : v \in \mathbf{V}\setminus(\{v_0\} \cup \mathbf{C}_t)\} \tag{2.7}$$

La pondération $r_H \in [0, 1]$ doit être choisie par le concepteur de la stratégie. $\mathbf{C}_t$ correspond à l'ensemble des nœuds qui à l'instant t sont nœuds cibles des autres agents. Il est important de noter que lorsque plusieurs agents arrivent simultanément sur leurs nœuds cibles respectifs, *Heuristic* traitera leurs requêtes de prochain nœuds cible dans l'ordre de leur identifiant. $\mathbf{C}_t$ est donc propre à chaque agent considéré. Cela permet aux agents d'avoir tout le temps des nœuds cible différents. La détermination du chemin vers le nœud cible prend en compte l'oisiveté des nœuds sur les chemins possibles entre v_0 et le nœud cible. Ce chemin optimal est déterminé via l'algorithme *Pathfinder* qui réévalue les arcs des chemins allant du nœud v au nœud w en fonction de l'oisiveté des nœuds traversés. Dans cet algorithme un arc est évalué comme suit :

$$\gamma_{v,w}(r_P) = r_P \bar{i_t}(w) + (1 - r_P)\bar{c}_{v,w} \tag{2.8}$$

où la pondération $r_P \in [0, 1]$ doit également être choisie par le concepteur de la stratégie et le temps de trajet normalisé de l'arc $\bar{c}_{v,w}$ est donné par :

$$\bar{c}_{v,w} = \frac{c_{v,w} - \min_{\{x,y\} \in \mathbf{E}} c_{x,y}}{\max_{\{x,y\} \in \mathbf{E}} c_{x,y} - \min_{\{x,y\} \in \mathbf{E}} c_{x,y}} \tag{2.9}$$

Lorsque tous les arcs présentent le même temps de trajet, une valeur arbitraire de $\frac{1}{2}$ est attribuée à tous les $\bar{c}_{v,w}$. Le calcul du chemin avec des poids donnés par l'équation (2.8) fait faire aux agents des détours passant par des nœuds d'oisiveté élevée au lieu d'aller plus rapidement vers le nœud cible. HPCC est donc une stratégie centralisée et omnisciente.

En ce qui concerne la complexité algorithmique, pour le coordinateur, qui est le processus le plus coûteux de cette stratégie, elle est de [19] :

- $N_a \times \dfrac{N}{N + |\mathbf{E}| + \left\lfloor \bar{l} \right\rfloor}\, O(N \log(N))$, pour le calcul du prochain nœud cible dans tout le graphe ainsi que son chemin pour tous les agents requéreurs,
- $2N_a \times \dfrac{N}{N + |\mathbf{E}| + \left\lfloor \bar{l} \right\rfloor}$, pour la communication avec ces agents requéreurs,

où $\bar{l}$ est la longueur moyenne des arêtes. Cette complexité est suffisamment faible pour permettre un fonctionnement en ligne de HPCC.

Par ailleurs, il a été montré que c'est l'une des meilleures stratégies en ligne [2, 19]. Elle peut donc être considérée comme stratégie de référence en ce qui concerne les stratégies en ligne. Néanmoins, reposant sur les oisivetés réelles des nœuds, et donc sur les mouvements de tous les agents à tout instant, elle a pour principal défaut d'être centralisée, et ce faisant, non résiliente.

La stratégie à cycle unique consiste à calculer hors ligne un cycle optimal ou quasi optimal passant par tous les nœuds puis à faire circuler tous les agents sur ce cycle. Un théorème permet alors de borner l'oisiveté maximale des nœuds pour cette stratégie [6]. Cette borne comprend deux parties : la première correspond au temps de parcours du cycle divisé par le nombre d'agents et la seconde fait apparaître le plus long temps de parcours entre deux nœuds consécutifs du cycle. Cette seconde partie de la borne confirme l'intuition d'inadaptation d'un cycle unique pour des graphes où plusieurs groupes de nœuds sont séparés par de longues distances. La stratégie proposée ici vise à remédier à ce défaut en apprenant un comportement d'agent en fonction du graphe à patrouiller et du nombre total d'agents impliqués dans la mission.

La stratégie basée sur l'utilisation des processus décisionnels de Markov partiellement observables et distribués considère une action qui est le produit croisé des actions des différents agents et calcule une politique qui indique à chaque agent l'action qu'il doit entreprendre étant donné l'observation de sa propre position [3]. Bien que la prise de décision soit distribuée – mais calculée hors ligne –, ce travail se distingue de la méthode proposée ici par le fait que tous les agents de la patrouille ont la même stratégie alors que, pour l'application des processus décisionnels de Markov partiellement observables et distribués, chaque agent a une stratégie qui lui est propre. On peut penser que l'approche proposée ici est plus robuste à la défaillance d'un agent. En effet, bien que pour les deux approches la stratégie soit calculée pour un nombre donné d'agents, l'approche proposée ici, contrairement à l'approche basée sur les processus décisionnels de Markov, ne peut attribuer un rôle à un agent, par exemple définir un ensemble de noeuds patrouillés uniquement par cet agent. En cas de panne de cet agent la stratégie basée sur les processus décisionnels de Markov présenterait une performance déplorable du fait que certains noeuds ne seraient plus du tout visités.

2.2. Quelques éléments sur les réseaux neuronaux

2.2.1. *L'encodage 1 parmi n*

L'encodage 1 parmi n consiste à traduire une variable discrète pouvant prendre n valeurs par un ensemble de variables Booléennes, toutes égales à zéro sauf celle

dont l'indice correspond à la valeur de la variable discrète qui est égale à un. Bien que courant dans le domaine de l'apprentissage automatique, cet encodage n'est pas spécifique aux réseaux de neurones : on le retrouve depuis longtemps dans d'autres domaines comme par exemple la synthèse de circuits intégrés [4].

2.2.2. *L'architecture LSTM*

Les réseaux de neurones récurrents sont des réseaux de neurones qui traitent successivement chaque élément d'une séquence d'entrées. Ils maintiennent dans leurs couches cachées un vecteur d'état, appelé état caché, qui contient de l'information sur le passé de la séquence. Àun instant t, les sorties des cellules cachées h_t dépendent de de leur sortie au temps précédent h_{t-1}. Cet état caché peut être considéré comme une mémoire qui permet de traiter une information donnée au fur et à mesure : l'information stockée dans l'état caché permet de trouver des corrélations entre des événements séparés par plusieurs périodes d'échantillonnage.

Les LSTM correspondent à une architecture spécifique de réseau récurrent conçue pour prendre en compte des dépendances de long terme [12, 10]. La cellule élémentaire de cette architecture est présentée sur la Figure 2.1. À l'origine, Hochreiter *et al.* ont défini l'architecture LSTM comme un réseau récurrent avec une couche d'entrée, une couche cachée entièrement connectée contenant des cellules de mémoire, c sur la figure, des portes, i, f et o sur la figure, et une couche de sortie. Chaque cellule mémoire est un neurone présentant une récurrence temporelle locale. Àl'instant t, le vecteur des états des cellules de mémoire de la couche cachée est noté c_t et parfois appelé état des cellules du réseau. Les vecteurs de signaux des portes d'entrée, i_t, d'oubli, f_t, et de sortie, o_t, contrôlent à différents niveaux par des multiplications le flux allant de l'entrée à la sortie en passant par la cellule c_t. Ces portes constituent une analogie continue des opérations discrètes d'écriture, de lecture et de mise à zéro. L'état des cellules correspond donc à la mémoire accumulée par le LSTM grâce à ses portes d'entrée, d'oubli et de sortie. Le fait qu'il y ait d'une part l'état des cellules de mémoire, c_t, et d'autre part l'état des sorties des cellules cachées, h_t, permet aux LSTM de présenter des modes de mémorisation supplémentaires par rapport à d'autres réseaux de neurones récurrents.

Les réseaux LSTM profonds présentent plusieurs couches. On utilise alors l'exposant l pour indiquer les signaux et paramètres de la $l^{\text{ème}}$ couche. Une couche LSTM est alors décrite par les équations suivantes :

$$i_t^l = \sigma(W_{xi}^l x_t^l + W_{hi}^l h_{t-1} + b_i^l) \tag{2.10}$$

$$f_t^l = \sigma(W_{xf}^l x_t^l + W_{hf}^l h_{t-1} + b_f^l) \tag{2.11}$$

$$o_t^l = \sigma(W_{xo}^l x_t^l + W_{ho}^l h_{t-1} + b_o^l) \tag{2.12}$$

$$c_t^l = f_t^l * c_{t-1}^l + i_t^l * \tanh(W_{xc}^l x_t^l + W_{hc}^l h_{t-1} + b_c^l) \tag{2.13}$$

$$h_t^l = o_t^l * \tanh(c_t^l) \tag{2.14}$$

où σ est la fonction sigmoïde, $*$ symbolise la multiplication terme à terme et :

- W_{xi}^l, W_{hi}^l, W_{xf}^l, W_{hf}^l, W_{xo}^l, W_{ho}^l, W_{xc}^l et W_{hc}^l sont des matrices de poids et
- b_i^l, b_f^l, b_o^l et b_c^l sont des vecteurs de biais.

Tous ces paramètres doivent être appris. Les couches s'enchaînent de proche en proche comme suit :

$$x_t^{l+1} = h_t^l \tag{2.15}$$

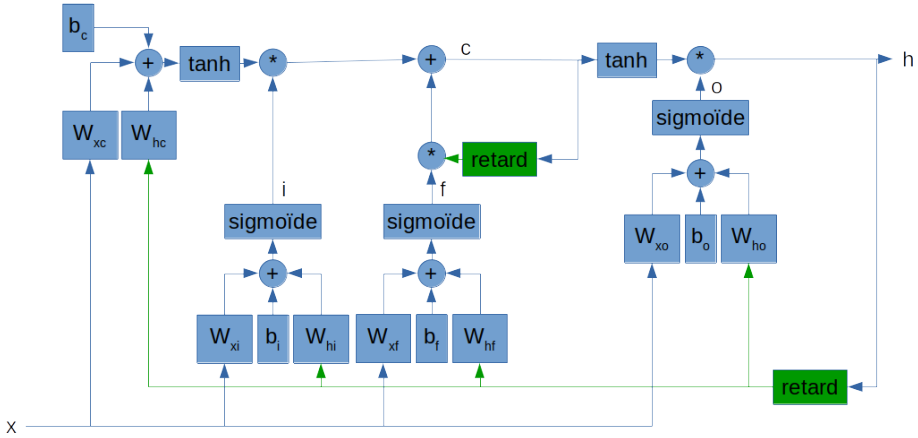

FIGURE 2.1. Couche d'un réseau LSTM ; les fonctions, retards et multiplications s'appliquent élément par élément aux vecteurs considérés

2.2.3. *Les couches softmax*

Une couche *softmax* d'entrée x et de sortie h est définie par :

$$h_j = \frac{e^{(W_s x + b_s)_j}}{\sum_{k=1}^{k=\dim(h)} e^{(W_s x + b_s)_k}} \tag{2.16}$$

où les indices j et k représentent respectivement le $j^{\text{ème}}$ et $k^{\text{ème}}$ éléments d'un vecteur, W_s est une matrice de poids et b_s un vecteur de biais. Ce type de couche permet d'obtenir un vecteur h présentant les caractéristiques d'une distribution de probabilité. En effet, par construction, la somme des éléments de h vaut 1 et chaque élément de h est positif [5].

2.3. UTILISATION DE RÉSEAUX DE NEURONES POUR LA PATROUILLE

Jusqu'à ce jour, peu de travaux se sont intéressés à l'utilisation de réseaux de neurones artificiels pour la patrouille multiagent. Guo *et al.* [11] ont utilisé des méthodes basées sur des réseaux de neurones artificiels pour planifier un chemin de patrouille assurant une couverture complète. Dans ce travail, la zone à surveiller est discrétisée

en disques de rayon donné qui peuvent correspondre à des nœuds à visiter. Chaque neurone du réseau est associé à une région et est activé négativement ou positivement en fonction respectivement de la présence d'un obstacle ou d'une absence de visite par la patrouille. Finalement, l'ensemble des activations des neurones du réseau constitue un paysage dynamique de sorte que les régions non visitées attirent le robot globalement sur tout l'espace, tandis que les obstacles le repoussent pour éviter des collisions. Le type de réseau de neurones utilisé n'est pas pertinent pour traiter la problématique de cet article, c'est-à-dire apprendre des chemins. Il faut également noter que dans ce travail un réseau de neurones unique est utilisé concomitamment par tous les agents, alors que dans le présent travail les agents ne doivent pas communiquer entre eux.

D'Ambrosio *et al.* [7] ont proposé un système de communication multiagent basé sur des réseaux de neurones embarqués dans chaque agent. Les agents communiquent entre eux par l'intermédiaire de connexions neuronales : certaines cellules de la couche de sortie du réseau d'un agent correspondent à de l'information à transmettre aux autres agents et certaines cellules de la couche d'entrée du réseau d'un agent correspondent à de l'information reçue des autres agents. Les poids des connections relatives à l'information transmise d'un robot à l'autre sont appris par algorithme évolutionnaire en utilisant les résultats d'une simulation robotique en ce qui concerne les performances. Dans la perspective qui est la nôtre, ces travaux présentent la même hypothèse restrictive que les travaux précédents : les agents communiquent entre eux non seulement lors de l'apprentissage en simulation mais aussi lors de l'exécution en opération. Ils sont néanmoins une source d'inspiration en cela que chaque agent utilise un réseau de neurones.

Les travaux de Sales *et al.* [20] sont également liés à notre problématique. En effet, ils présentent un système autonome de patrouille composé de quatre robots intelligents qui peuvent se déplacer librement dans un bâtiment pour y détecter des intrus. Les robots ont un système de localisation et de navigation qui comprend un réseau de neurones artificiels et une machine à états dont les états correspondent à des caractéristiques clefs de l'environnement. La machine à états génère conjointement deux séquences : une séquence d'actions à exécuter et une séquence d'états. Le réseau de neurones traite les données des capteurs pour identifier et classifier les états courants et futurs de la machine à états en vue de déterminer les actions à entreprendre. Après avoir été entraînés hors ligne à identifier les caractéristiques principales de l'environnement tels que les couloirs, les intersections et les virages, les réseaux sont ensuite utilisés en ligne. Nous avons retenu de ces travaux le principe d'entraîner les réseaux de neurones hors ligne en vue de les préparer pour une mission spécifique dans laquelle les agents naviguent le plus efficacement possible sans communiquer. Néanmoins, contrairement à notre travail, dans celui de Sales *et al.* chaque robot calcule un plus court chemin, tenant compte des autres robots, afin d'atteindre la position de l'intrus lorsqu'il est détecté, tandis que dans notre cas le réseau de neurones est utilisé pour sélectionner le nœud suivant parmi les voisins du nœud courant tout en tenant compte de la séquence des nœuds déjà parcourus, ainsi que de ce qui a été appris durant l'entraînement.

Les réseaux de neurones sont également utilisés comme modèles statistiques permettant d'estimer les oisivetés réelles à partir des oisivetés individuelles [16]. La

stratégie de l'agent utilise alors les méthodes de HPCC avec les oisivetés estimées à la place des oisivetés réelles permettant ainsi d'éviter la communication entre agents. L'apprentissage des réseaux de neurones est réalisée à partir d'enregistrements de séquences d'oisiveté réelle et individuelle couplées produites par HPCC. Pour différentes configurations, plusieurs modèles statistiques sont entraînés via la méthode des moindres carrés. Contrairement aux travaux d'estimation des oisivetés réelles à partir des oisivetés individuelles, le travail présenté ici se base sur des séquences de nœuds et non sur des séquences de couples d'oisivetés.

Finalement, des travaux récents [17, 18] ont jeté les premières bases pour l'approche ici proposée. Dans ces travaux, les structures de réseaux LSTM sont optimisées pour chaque graphe et chaque nombre d'agents. Les résultats obtenus indiquent des problèmes de robustesse dans les décisions prises par les agents. Ces problèmes se traduisent pour certains graphes par des performances sur l'intervalle quadratique moyen inférieures à celle de CR. L'apport du travail présenté ici par rapport à ces travaux récents utilisant des réseaux LSTM est :

- Une formalisation plus rigoureuse du problème abordé. En effet, les approximations successives réalisées pour passer d'une stratégie centralisée à une stratégie décentralisée basée sur un réseau LSTM sont exposées avec plus de clarté et formalisées plus rigoureusement que précédemment.
- Une initialisation analytique des réseaux LSTM basée sur la structure du graphe.
- L'utilisation d'une base de données d'apprentissage plus pertinente.
- Une évaluation utilisant des critères plus adaptés. En effet, les évaluations antérieures de stratégies basées sur des réseaux LSTM portent sur IM et IQM [17, 18]. Ici l'évaluation porte sur Iav et WI.

3. Une stratégie basée sur l'architecture LSTM

Cette section présente une stratégie d'agent décentralisée qui est basée sur une architecture neuronale récurrente apprenant à naviguer, dans un graphe donné et pour un nombre d'agents donné, à partir d'un ensemble de traces résultant de nombreuses exécutions d'une stratégie d'agents coordonnés, en pratique HPCC. La principale hypothèse émise ici est que si les agents apprennent à se comporter *en moyenne*, *i.e.* tendanciellement, comme s'ils étaient coordonnées, ou pour le mieux dire, comme s'ils se coordonnaient implicitement, ils pourront atteindre des performances proches de celle d'une stratégie où les agents seraient effectivement coordonnées.

3.1. Type et structure d'approximation

Une stratégie centralisée est en capacité de rassembler l'ensemble des composants de l'état du système, et lorsqu'un agent a doit choisir le prochain nœud à visiter $\hat{v}^a_{t+1}$ la procédure de décision peut prendre en compte non seulement le nœud où se trouve l'agent, $\hat{v}^a_t$, mais aussi l'ensemble de l'état du système à l'instant τ^a_t où l'agent a en est arrivé au $(t+1)^{\text{ème}}$ nœud de son parcours. Cet état, $S_{\tau^a_t}$ correspond aux positions

des autres agents et aux oisivetés de tous les nœuds. Formellement, la procédure de choix de cette stratégie centralisée peut être notée :

$$\hat{v}^a_{t+1} = \hat{f}_{\mathbf{G},N_a}(\hat{v}^a_t, S_{\tau^a_t}) \tag{3.1}$$

Notons que cette manière de formaliser une stratégie est volontaire et cohérente avec la définition d'une stratégie en Programmation Dynamique comme une fonction donnant l'action en fonction de l'état. Par exemple, on trouve dans la littérature une formalisation de stratégie d'agent comme une fonction de t vers $\hat{v}^a_{t+1}$ [6]. Cette formulation alternative présente le désavantage de confondre une stratégie et sa trace. Pour un graphe et un nombre donné d'agents, le réseau de neurones récurrent est entraîné à déterminer le prochain nœud à visiter en fonction des nœuds précédents $\hat{v}^a_t, \hat{v}^a_{t-1} \dots \hat{v}^a_0$ d'un chemin parcouru par un agent a, ceci pour tous les agents, toutes les longueurs de chemin et plusieurs exécutions de simulation. On a donc une première approximation qui consiste à considérer uniquement le chemin parcouru par l'agent pour déterminer le nœud suivant $\tilde{v}^a_{t+1}$. Néanmoins, étant donné qu'une information propre à l'agent est utilisée en lieu et place de l'état global du système, il est en pratique très difficile, voire impossible, d'avoir une relation déterministe entre la séquence des nœuds précédents et le nœud suivant. En effet :

- Les agents ne connaissent plus la position des autres agents.
- L'identité de l'agent n'intervient pas explicitement dans la relation entre le chemin parcouru et le nœud suivant qui est la même pour tous les agents.
- L'absence de prise en compte de l'identité de l'agent peut mener à des raisonnements circulaires lorsqu'on impose que la forme de la relation soit une fonction.

On postule donc l'existence d'une distribution de probabilité sur le nœud suivant, cette distribution étant conditionnée par la séquence des nœuds précédents. Soit

$$p(\tilde{v}^a_{t+1} \mid \hat{v}^a_t, \hat{v}^a_{t-1} \dots \hat{v}^a_0) = \left(p(\tilde{v}^a_{t+1} = 1 \mid \hat{v}^a_t, \hat{v}^a_{t-1} \dots \hat{v}^a_0),\ \dots,\ p(\tilde{v}^a_{t+1} = N \mid \hat{v}^a_t, \hat{v}^a_{t-1} \dots \hat{v}^a_0)\right)^T$$

cette distribution de probabilités sur le nœud suivant, on a :

$$p(\tilde{v}^a_{t+1} \mid \hat{v}^a_t, \hat{v}^a_{t-1}, \dots, \hat{v}^a_0) = \tilde{f}_{\mathbf{G},N_a}(\hat{v}^a_t, \hat{v}^a_{t-1}, \dots, \hat{v}^a_0) \tag{3.2}$$

Cette première approximation est relative au choix des données d'apprentissage. Ce choix a été fait d'une part parce que l'agent a a uniquement accès à ses propres données, et d'autre part parce qu'il a été décidé de se concentrer sur les nœuds parcourus qui reflètent, dans une certaine mesure, la valeur des oisiveté individuelles. En effet l'agent pourrait calculer ses oisivetés individuelles en fonction des nœuds qu'il a parcourus. Une fois ce choix effectué, on obtient alors une distribution de probabilité sur le nœud suivant au lieu d'une fonction quant au processus de décision. Intuitivement, il est possible de considérer l'intérêt de cette distribution de probabilité par analogie avec l'intérêt des stratégies mixtes basées sur des distributions de probabilité en théorie des jeux. Finalement, ces séquences étant apprises par un réseau récurrent de type LSTM propre au graphe et au nombre d'agents, la dépendance par rapport à la séquence de nœuds est remplacée par une dépendance par rapport au dernier nœud et par rapport aux

états h et c du réseau. Les états du réseau constituant sa mémoire, on peut considérer que la liste des nœuds parcourus est retenue, et cela au moins de manière vague pour les nœuds qui ont été parcourus longtemps auparavant. Soit v^a_{t+1} la variable aléatoire estimée par le réseau, on a alors avec les mêmes notations que précédemment :

$$p(v^a_{t+1} \mid \hat{v}^a_t, h^1_t, \ldots, h^L_t, c^1_t, \ldots, c^L_t, \theta) = f_{\mathbf{G},N_a}(\hat{v}^a_t, h^1_t, \ldots, h^L_t, c^1_t, \ldots, c^L_t, \theta) \quad (3.3)$$

où L est le nombre de couches LSTM dans le réseau et θ l'ensemble des poids et biais des différentes couches de celui-ci.

Le type de réseau utilisé est présenté sur la Figure 3.1. Son entrée est traitée par une couche d'encodage binaire qui est l'entrée de la première couche LSTM comprenant H cellules. D'autres couches LSTM avec le même nombre de cellules suivent. La dernière couche LSTM alimente une couche softmax sans biais présentant autant de cellules qu'il y a de nœuds dans le graphe. On a donc :

- Pour la couche d'entrée, les matrices W^1_{xi}, W^1_{xf}, W^1_{xo} et W^1_{xc} possèdent $N.H$ paramètres, les matrices W^1_{hi}, W^1_{hf}, W^1_{ho} et W^1_{hc} ont H^2 paramètres et les biais b^1_i, b^1_f, b^1_o et b^1_c ont H paramètres, soit un nombre total de paramètres égal à $4H^2 + 4NH + 4H$.
- Pour une couche intermédiaire l, les matrices W^l_{xi}, W^l_{xf}, W^l_{xo}, W_{xc} W^l_{hi}, W^l_{hf}, W^l_{ho}, et W^l_{hc} possèdent toutes H^2 paramètres et les biais b^l_i, b^l_f, b^l_o et b^l_c ont tous H paramètres. Soit un nombre total de paramètres de $8H^2 + 4H$.
- Pour la couche softmax, la matrice W^l_s a $N.H$ paramètres.

Étant donné qu'il y a $L-1$ couches intermédiaires la dimension de θ est $\dim(H, L, N) = 4(2L-1)H^2 + (4L+5N)H$ et le réseau est noté $L-H$.

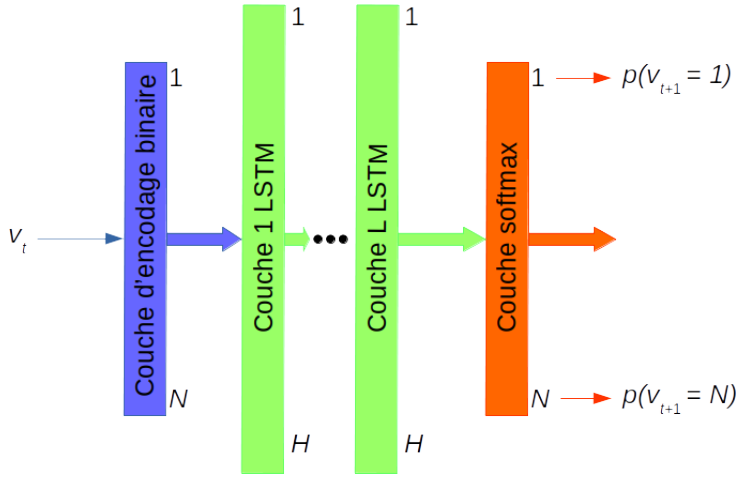

FIGURE 3.1. Réseau utilisé

3.2. Apprentissage

Un apprentissage de type supervisé est utilisé pour entraîner les réseaux LSTM sur des traces d'une stratégie centralisée $\hat{f}_{\mathbf{G},N_a}$ exécutée sur un graphe $\mathbf{G}$ par N_a agents. Ces traces sont constituées de M séquences comprenant chacune T nœuds. Elles résultent de plusieurs simulations dans lesquelles les N_a agents sont positionnés initialement sur des nœuds sélectionnés aléatoirement.

Pour ces traces un critère d'entropie croisée est minimisé. c'est-à-dire :

$$\min_{\theta} - \sum_{m=1}^{M} \sum_{t=1}^{T} \log p \left(v_{t+1}^{a(m)} = \hat{v}_{t+1}^{a(m)}(m) \mid \hat{v}_t^{a(m)}(m), h_t^1, \dots h_t^L, c_t^1, \dots c_t^L, \theta \right) \quad (3.4)$$

où $a(m)$ est l'agent concerné par la $m^{\text{ème}}$ séquence et $\hat{v}_t^{a(m)}(m)$ est le $t^{\text{ème}}$ nœud généré par la stratégie centralisée dans la $m^{\text{ème}}$ séquence. Les valeurs des états h et c du réseau sont considérés après application de la séquence $\{\hat{v}_0^{a(m)}(m), \dots \hat{v}_{t-1}^{a(m)}(m)\}$ à celui-ci avec un état initial nul. Notons que l'identité $a(m)$ de l'agent ayant parcouru la séquence n'a aucune influence sur le résultat de l'apprentissage. On s'interdit donc d'apprendre des rôles d'agents et l'affectation de groupes de nœuds à un agent étant donné son rôle dans la patrouille. Pour l'affectation au moment de l'exécution on compte uniquement sur le tirage aléatoire dans la distribution issue du LSTM.

3.2.1. *Prise en compte de la topologie du graphe*

Il est important de noter que l'apprentissage réalisé sur la base du critère donné par l'équation (3.4) considère uniquement les traces d'exécution de la stratégie. De ce fait, il n'est pas capable de distinguer un arc $\mathbf{G}$ qui n'est pas parcouru par la stratégie de l'absence de connexion entre deux nœuds de $\mathbf{G}$. Préalablement à cet apprentissage, en vue d'initialiser le réseau de neurones, une phase de pré-apprentissage vise donc à apprendre la topologie de $\mathbf{G}$ indépendamment de la la stratégie et du nombre d'agents. En effet, quels que soient la stratégie et le nombre d'agents, la stratégie produit un nœud suivant qui est voisin du nœud courant.

Pour capturer la topologie, on cherche à se donner une fonction de $\mathbf{V}$ dans $\mathbb{R}^N$ qui produise, pour un nœud v en entrée, un vecteur dont les composantes valent :

- 0 lorsque la composante ne correspond pas à un voisin de v et
- l'inverse du nombre de voisins de v lorsque la composante correspond à un voisin de v.

Les deux méthodes pouvant être utilisées pour le pré-apprentissage feront l'objet des sous-sections suivantes.

3.2.2. *Pré-apprentissage empirique*

Ici une base d'apprentissage constituée de M_p couples $\{\hat{v}_0^{a(m)}(m), \hat{v}_1^{a(m)}(m)\}$ est produite en tirant aléatoirement $\hat{v}_0^{a(m)}(m)$ parmi les nœuds de $\mathbf{G}$, puis en tirant

aléatoirement $\hat{v}_1^{a(m)}(m)$ parmi les voisins de $\hat{v}_0^{a(m)}(m)$. Le pré-apprentissage consiste alors à minimiser un critère d'entropie sur cette base. c'est-à-dire :

$$\min_{\theta} - \sum_{m=1}^{M_P} \log p \left(v_1^{a(m)} = \hat{v}_1^{a(m)}(m) \mid \hat{v}_0^{a(m)}(m), 0, \ldots 0, 0, \ldots 0, \theta \right) \tag{3.5}$$

Notons que la notation $a(m)$ est totalement arbitraire et est utilisée uniquement pour des raisons de compatibilité avec les notations précédentes. Lorsque ce type de pre apprentissage est utilisé la stratégie est appelée *Random- Next-Neighbour-LSTM-Path-Maker* (RLPM).

3.2.3. *Pré-apprentissage analytique*

Pour un réseau du type de la Figure 3.1 pour lequel $H = N$ une initialisation analytique partant de la sortie et remontant vers l'entrée est possible. Le principe est ici de capturer $\mathbf{G}$ avec la dernière couche LSTM et de fixer les couches précédentes par induction arrière de manière à ce qu'elles se comportent comme la fonction identité. Lorsque cette procédure de pré-apprentissage est appliquée la stratégie résultante est appelée *RAndom Multiagent PAtrollinG LSTM-Path-MakER* (RAMPAGER).

En fixant la matrice de la dernière couche softmax à $k \cdot I_N$, où k est grand et positif, on obtient une normalisation de la couche de sortie qui conserve l'ordre de ses composants. Dans la pratique une valeur de k supérieure ou égale à 5 permet d'obtenir un ordre sur les composants suffisamment discriminant.

Pour toutes les couches LSTM, l'influence de la sortie précédente est neutralisée en fixant à 0 les matrices utilisées pour le contrôle de toutes les portes par la sortie et ainsi que celle utilisée pour impacter l'état interne. De plus, les biais sont fixés à 0.

Pour la dernière couche LSTM, les composantes du vecteur de sortie correspondant à un voisin doivent avoir une valeur élevée pour un voisin du nœud courant, et une valeur faible sinon. La porte de sortie doit par conséquent être ouverte seulement pour ces voisins, conduisant alors à une matrice de contrôle de la porte de sortie par l'entrée W_{xo} :

$$W_{xo}(v,u) \approx \begin{cases} +\infty & \text{si } v \text{ est voisin de } u, \\ -\infty & \text{sinon.} \end{cases} \tag{3.6}$$

Les composantes du flux d'entrée arrivant à l'état c correspondant aux voisins du nœud courant doivent également être positifs. De manière à éviter un demi-tour de l'agent, il est nécessaire d'assurer que le bloc ne génère pas une distribution comprenant le nœud courant à la prochaine étape. Par conséquent, le flux d'entrée vers l'état c correspondant au nœud courant doit être négatif. Pour obtenir ceci, il est nécessaire que la sortie de l'activation tanh de l'équation (2.13) soit proche de -1 pour le nœud courant. En considérant que l'entrée est codée en 1 parmi N, on déduit la matrice

W^{xc} d'alimentation de l'état par l'entrée :

$$W_{xc}(v,u) \approx \begin{cases} -\infty & \text{si } u = v, \\ +\infty & \text{si } v \text{ est voisin de } u, \\ 0 & \text{sinon.} \end{cases} \tag{3.7}$$

En ce qui concerne la matrice W^{xi} qui permet le contrôle de la porte d'entrée par l'entrée, on se base sur le fait que cette porte doit être ouverte pour les voisins du nœud courant. En considérant de nouveau le codage 1 parmi N de l'entrée, on trouve $W_{xi}(v,u) \approx +\infty$ quand v est voisin de u. Finalement, en ce qui concerne la matrice W_{xf} qui permet le contrôle de la porte d'oubli par l'entrée, de manière à éviter une visite de nœuds récemment visités le flux du nœud courant doit également passer au travers de la porte d'oubli pour les nœuds voisins. Ces considérations conduisent alors à :

$$W_{xi}(v,u) = W_{xf}(v,u) \approx \begin{cases} +\infty & \text{si } v \text{ est voisin de } u, \\ 0 & \text{sinon.} \end{cases} \tag{3.8}$$

Notons que ce mécanisme pondère le nœud précédemment visité par $\tanh(1/2)$ alors que les autres voisins sont pondérés par $\tanh(1)$. Il ne s'agit donc pas d'une interdiction de faire demi-tour, ce qui serait problématique pour des graphes présentant des impasses, mais simplement d'une initialisation de la recherche de solution déconseillant les demi-tours. Il est néanmoins possible, lorsque les traces présentent des demi-tours, que la solution après apprentissage génère des probabilités de demi-tour.

Pour une architecture avec plus d'une couche LSTM, l'initialisation consiste à fixer les blocs des couches LSTM intermédiaires de manière à ce qu'elles se comportent comme la fonction identité. Pour obtenir ce résultat, toutes les portes sont ouvertes pour le nœud courant et fermées pour les autres nœuds :

$$W_{xo}(v,u) = W_{xi}(v,u) = W_{xf}(v,u) \approx \begin{cases} +\infty & \text{si } u = v, \\ -\infty & \text{sinon.} \end{cases}$$

Pour assurer que la fonction tangente hyperbolique de l'équation de sortie soit proche de 1, la cellule doit produire la plus haute valeur possible pour le nœud courant. Par conséquent, l'activation de la tangente hyperbolique de l'équation (2.13) doit être la plus proche possible de 1 pour le nœud courant. De plus, pour les autres nœuds la tangente hyperbolique doit être proche de 0. Une solution est :

$$W_{xc}(v,u) \approx \begin{cases} +\infty & \text{si } u = v, \\ 0 & \text{sinon.} \end{cases} \tag{3.9}$$

Dans la pratique, des valeurs suffisamment élevées pour saturer $\sigma()$ ou $\tanh()$ sont utilisées en place de $+\infty$ et $-\infty$ dans les expressions.

3.3. Utilisation du réseau pour la décision d'un agent

En général, malgré le pré-apprentissage, il n'y a pas de garantie que le support de $f_{\mathbf{G},N_a}(v_t^a, h_t^1, \dots h_t^L, c_t^1, \dots c_t^L, \theta)$ se résume au voisinage de v_t^a dans $\mathbf{G}$. L'agent a qui

se trouve au nœud v_t^a commence donc par limiter la distribution de probabilité générée par le réseau au voisinage du nœud courant :

$$\bar{f}_{\mathbf{G},N_a}(v_t^a, h_t^1, \dots h_t^L, c_t^1, \dots c_t^L, \theta) = R(f(v_t^a, h_t^1, \dots h_t^L, c_t^1, \dots c_t^L, \theta), v_t^a) \quad (3.10)$$

où $R : [0,1]^N \times \mathbf{V} \to [0,1]^N$ est la fonction mettant à zéro les valeurs des composantes du vecteur ne correspondant pas à un nœud du voisinage du nœud donné en entrée. Ensuite $\bar{f}_{\mathbf{G},N_a}(v_t^a, h_t^1, \dots h_t^L, c_t^1, \dots c_t^L, \theta)$ est normalisé en divisant chacune de ses composantes par leur somme. Finalement, l'agent tire aléatoirement le nœud suivant v_{t+1}^a dans cette distribution normalisée. Cette manière de procéder, c'est-à-dire réaliser un tirage aléatoire au lieu de choisir le nœud avec la probabilité la plus élevée, a été déterminée suite à de mauvaises performances constatées pour l'utilisation de cette dernière méthode.

4. Expérimentation et résultats

4.1. Outils

L'évaluation des stratégies présentées en section 3 utilise une suite logicielle[(1)] comprenant :

- PyTrol, un simulateur, facile d'usage et codé en Python, adéquat non seulement pour étudier des stratégies de patrouille mais aussi pour générer rapidement des traces d'exécution ;
- MAPTrainer, un environnement de développement basé sur PyTorch pour entraîner des modèles d'apprentissage automatique dans le contexte de la patrouille multiagent ;
- MAPTor, un outil annexe pour :
  - préparer les configurations de patrouille multiagent, les scénarios et les exécutions,
  - traiter les données générées par PyTrol de manière à les préparer pour l'apprentissage de modèles par MAPTrainer,
  - réaliser des analyses statistiques sur les traces de simulation générées par PyTrol,
  - tracer des statistiques pour chaque scénario de patrouille multiagent étudié.

Pytrol est un simulateur à temps discret : la progression de la mission de la patrouille est simulée en enchaînant des incréments temporels de durée constante appelés pas de temps. Dans un pas de temps :

- l'oisiveté de chaque nœud est incrémentée,
- chaque agent progresse le long de son arc courant,

(1) `https://github.com/mothguib/`

- les agents arrivant sur un nœud après avoir parcouru un arc prennent une décision quant au prochain nœud voisin à visiter en appliquant leur stratégie ; concrètement, dans le simulateur ils se voient attribuer un nouvel arc menant vers le prochain nœud voisin,
- les quantités nécessaires au calcul des critères sont stockées et les oisivetés des nœuds où se trouvent les agents sont remises à zéro.

Notons que les temps de parcours des arcs $c_{v,w}$ doivent être des multiples de la durée de l'incrément temporel. Finalement, suivant la nature de la stratégie, l'application de celle-ci par les agents peut se traduire ou non par des échanges entre agents ou avec un coordinateur.

4.2. Scénarios

Les expérimentions numériques sont réalisées en faisant varier le graphe, le nombre d'agents et la stratégie. Un jeu de 6 graphes possédant 50 nœuds, nommés respectivement Map A, Map B, Circular, Corridor, Islands et Grid, est utilisé pour l'évaluation des stratégies de patrouille multiagent [2]. Le graphe Map B présente un nœud par lequel les agents sont obligés de passer pour changer de groupe de nœuds. Les graphes Circular et Corridor correspondent à des séquences de nœuds. Ces situations présentant peu d'intérêt pratique, dans la mesure où pour ces graphes il est direct que la meilleure stratégie de patrouille à employer est une stratégie à cycle unique, les graphes considérés ici sont Islands, Map A et Grid et sont présentées sur la Figure 4.1.

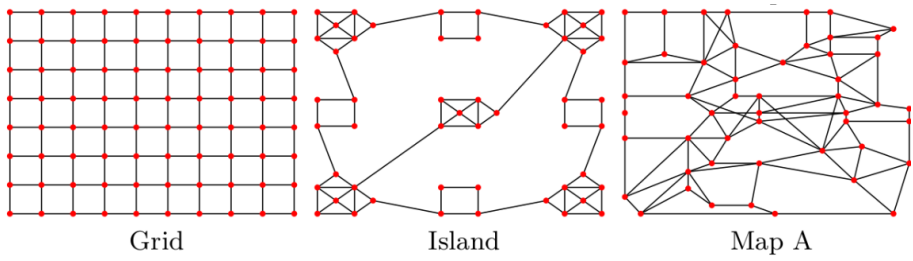

Figure 4.1. Graphes utilisés lors des expérimentations

Il est possible de constater que Grid correspond à une situation sans obstacle où les points d'intérêt sont régulièrement répartis. À l'opposé Islands correspond à un environnement encombré où les points d'intérêt sont localisés par groupes. Ceci évoque des zones boisées dans un massif montagneux. Le graphe Map A correspond à une situation intermédiaire entre les deux graphes précédents. Les scénarios sont définis pour 5, 10, 15 et 25 agents. En effet, des nombres d'agents plus faibles que 5 limitent l'aspect multiagent et, étant donné les graphes considérés, des nombres d'agents plus élevés que 25 conduisent à des problèmes pour lesquels le nombre de nœuds par agent est très faible. Les stratégies considérées sont :

- HPCC telle que décrite en section 2.1.3, une stratégie centralisée de haute performance typique, avec $r = 0.5$;
- CR telle que décrite en section 2.1.3, une stratégie distribuée sans communication entre agents typique ;
- RPLM telle que décrite en section 3.2.2, c'est-à-dire un réseau LSTM utilisé comme prédicteur de nœud avec un pré-apprentissage empirique ;
- RAMPAGER telle que décrite en section 3.2.3, c'est-à-dire un réseau LSTM utilisé comme prédicteur de nœud avec un pré-apprentissage analytique.

Chaque simulation dure 3 000 pas de temps de 1 seconde chacun. Lorsque la simulation est utilisée pour effectuer une évaluation, les critères de pire oisiveté (WI) et d'oisiveté moyenne (Iav) sont considérés dans la mesure où WI reflète le principal défaut dans le système de patrouille et Iav indique l'efficacité globale.

4.3. Apprentissage

Pour les stratégies RPLM, un ensemble de modèles LSTM déjà appris, sur une base de données générée par des exécutions de HPCC avec $r = r_H = r_P = 0.2$, et déjà évalués en matière de performance de la stratégie correspondante pour les critères d'intervalle moyen (IM) et d'intervalle quadratique moyen (IQM) [17, 18], sont utilisés directement dans les stratégies.

Table 4.1. Critères, normalisés et moyennés sur les graphes Map A, Islands et Grid ainsi que sur 5, 10, 15 et 25 agents, pour différentes configurations de HPCC

r	Iav	IM	IQM	WI
0.1	940	**147**	489	31 188
0.2	546	149	397	30 025
0.5	**233**	184	**293**	**1 216**
0.8	319	194	353	4 923
0.9	329	199	365	3 378

Le tableau 4.1 suggère que le choix de 0.2 pour r n'est pas des plus judicieux lorsqu'une optimisation de la régularité des visites est recherchée. Par conséquent, pour RAMPAGER les modèles sont appris à partir d'une base de données construite avec des données issues de l'exécution de HPCC avec une valeur de 0.5 pour r. Notons que le choix de HPCC comme stratégie de référence pourrait être mis en question. En effet, il existe une borne maximale théorique de WI pour la stratégie à cycle unique. Néanmoins le choix de continuer à utiliser HPCC peut se justifier par :

- le fait que HPCC est la meilleure stratégie en ligne connue et surpasse la stratégie à cycle unique sur des graphes du type de Islands [2],

- le fait qu'une stratégie à cycle unique ne nécessite pas *a priori* de mécanisme complexe pour être distribuée. En effet, chaque agent peut mémoriser le cycle comme une fonction du nœud précédent et du nœud courant vers le nœud suivant,
- l'existence d'une borne maximale théorique ne garantit pas que la stratégie à cycle unique présente les meilleures performances.

Les données utilisées consistent en 10 000 séquences, *i.e.* traces, générées pour chaque scénario, conduisant à :

- 2 000 exécutions pour un scénario avec 5 agents,
- 1 000 exécutions pour un scénario avec 10 agents,
- 667 exécutions pour un scénario avec 15 agents,
- 400 exécutions pour un scénario avec 25 agents,

cela pour chaque graphe étudié.

Par conséquent, le volume de données ne dépend pas du nombre d'agents N_a mais est identique pour tous les scénarios. Finalement, cette base de données inclut des données pour les trois graphes. Néanmoins, l'apprentissage est réalisé pour chaque graphe et chaque nombre d'agents en extrayant de cette base les données correspondantes.

4.3.1. *Choix d'une architecture pour RAMPAGER*

En utilisant la notation de la section 3.1, les architectures LSTM considérées ici en termes de *L-H* sont 1-50, 2-50, 3-50 et 4-50, conduisant aux variantes correspondantes de la stratégie RAMPAGER. L'entraînement de toutes ces architectures pour les trois graphes nécessitant un temps de traitement important, l'apprentissage de ces différentes architectures LSTM a d'abord été réalisé sur un seul graphe, puis la meilleure architecture a été retenue pour être évaluée pour tous les scénarios. C'est le graphe Map A qui a été choisi pour déterminer la meilleure architecture car il présente une topologie intermédiaire entre l'extrême régularité du graphe Grid et l'isolation de groupes de nœuds du graphe Islands. Après apprentissage des différentes architectures LSTM, les stratégies correspondantes sont donc évaluées sur le graphe de Map A, pour 5, 10, 15 et 25 agents, puis finalement la meilleure architecture est sélectionnée pour être entraînée pour les autres scénarios. Le Tableau 4.2 présente les critères WI et Iav, moyennés sur les nombres d'agents, pour les quatre réseaux LSTM. On constate que 2-50 est le plus performant pour les deux critères, bien que les autres réseaux conduisent à des valeurs à peine supérieures. Étant donné que, d'une part, les nombres de paramètres pour les architectures 1-50, 2-50, 3-50 and 4-50 sont respectivement de 22 700, 42 900, 63 100 et 83 300 et que, d'autre part, l'apprentissage est réalisé sur 10 000 séquences, il est légitime de supposer que l'architecture 1-50 ne présente pas assez de paramètres pour réaliser une approximation suffisamment précise et que les architectures 3-50 et 4-50 induisent du sur-apprentissage. Par conséquent, 2-50 est choisi pour être entraîné pour tous les graphes et expérimenté en simulation. L'évaluation concerne donc RAMPAGER 2-50.

TABLE 4.2. Critères d'évaluation normalisés, moyennés sur les nombres d'agents, pour quelques architectures LSTM simulées sur le graphe Map A

Architecture	WI	Iav
1-50	5 590	391
2-50	**5 363**	**380**
3-50	5 493	390
4-50	5 453	389

4.3.2. *Apprentissage pour RAMPAGER*

12 scénarios résultant de la combinaison des trois graphes et des quatre tailles de population d'agents étudiés sont ici éprouvés, ce qui implique d'apprendre et d'évaluer en simulation 12 architectures LSTM 2-50. L'apprentissage est effectué en utilisant l'algorithme Adagrad [8] pour 500 époques avec un taux d'apprentissage de 0.1. Avant de commencer l'étape d'apprentissage, l'entropie croisée calculée du réseau LSTM sur la base de données entière et l'exactitude correspondante sont de 2.65 et 20 % respectivement. La Figure 4.2 présente l'entropie croisée calculée sur la base de données de validation pour le réseau LSTM 2-50 pendant son entraînement sur {HPCC 0.5, Map A, 15}. L'exactitude est également calculée sur la base de données de validation. L'entropie croisée s'étend de 1.09, après la première époque d'entraînement, à 0.86, après la dernière, et l'exactitude de 56 % à 64 %. De plus, suivant la Figure 4.2, le modèle converge après 200 époques approximativement.

FIGURE 4.2. Coût de validation du réseau LSTM 2-50 entraîné sur des données générées par exécution de HPCC avec $r = 0.5$ sur le graphe Map A avec 15 agents

Durant l'entraînement, les paramètres de la couche intermédiaire ont légèrement varié. Tous les paramètres des matrices de poids de la dernière couche LSTM sont approximativement centrés autour de quelques valeurs. Après l'apprentissage les distributions de ces paramètres présentent le même profil en étant centrées autour des

mêmes valeurs. Cependant, elles s'étalent sur un intervalle plus grand et leur pic est moins élevé. Cette évolution est le signe de la tendance de l'apprentissage à légèrement modifier les paramètres initialisés analytiquement pour intégrer les chemins, séquences temporelles de nœuds, présentés au réseau. En ce qui concerne les vecteurs de biais, leur distribution a évolué sporadiquement sans valeurs centrales spécifiques. Les profils de la première et dernière distributions sont totalement différents. Ceci conduit à la conclusion que les biais ont beaucoup changé et, contrairement aux poids, ont été impactés par l'apprentissage.

4.4. Résultats

Les résultats présentés ici se focalisent sur l'effet de l'initialisation analytique. Les Figures 4.3 à 4.8 montrent la performance sur WI et Iav, pour les graphes Islands, Map A et Grid, respectivement, de RAMPAGER et de la meilleure variante RLPM, moyennée sur 100 exécutions pour chaque scénario. Sur toutes ces figures, les écart-types sont indiqués par des lignes verticales, comme il est possible de le constater pour la Figure 4.3 pour le meilleur RLPM. Néanmoins, pour un grand nombre de valeurs moyennes l'écart-type est trop petit pour être observé.

Globalement, HPCC avec $r = 0.5$ est toujours la meilleure stratégie pour les deux critères. Il y a une différence significative entre ces performances et les performances des autres stratégies. Ce résultat est probablement la conséquence de la différence de système de communication entre agents : il n'y a pas de contrainte de communication pour HPCC alors que pour les autres stratégies les agents ne communiquent pas entre eux. Par ailleurs, il est possible de constater que, pour tous les graphes et pour les deux critères, les courbes relatives à HPCC ne sont pas croissantes. La borne supérieure théorique de WI pour la stratégie à cycle unique présente un terme divisé par le nombre d'agents et un terme constant [6]. Cette borne exprimée en critère normalisé se traduirait donc par une fonction affine croissante et l'avantage qu'elle confère à la stratégie à cycle unique parait par conséquent surfait.

Suivant les Figures 4.3 et 4.4, RAMPAGER est largement meilleur que le meilleur variant RLPM pour le graphe Islands. Pour WI il est légèrement moins performant que CR, alors que sur Iav il se situe entre CR et HPCC.

Ensuite, les Figures 4.5 et 4.6 montrent que pour le graphe Map A, RAMPAGER est meilleur que le meilleur variant RLPM pour les deux critères, en particulier pour 25 agents. En ce qui concerne WI il est légèrement moins performant que CR, excepté pour 15 agents où il est meilleur. Pour le critère Iav RAMPAGER est toujours plus performant que CR.

Pour le graphe Grid, les Figures 4.7 et 4.8 mettent en évidence le fait que RAMPAGER est meilleur que le meilleur variant RLPM. Pour WI, il présente des performances proches de celles de CR pour 5 et 10 agents, égales à celle de CR pour 15 agents, et meilleures pour 25 agents. Il est possible de constater que RAMPAGER est globalement équivalant à CR pour WI, avec un WI moyen, tous nombres d'agents confondus, de 4 313 pour RAMPAGER contre 4 450 pour CR ; pour le graphe Grid RAMPAGER

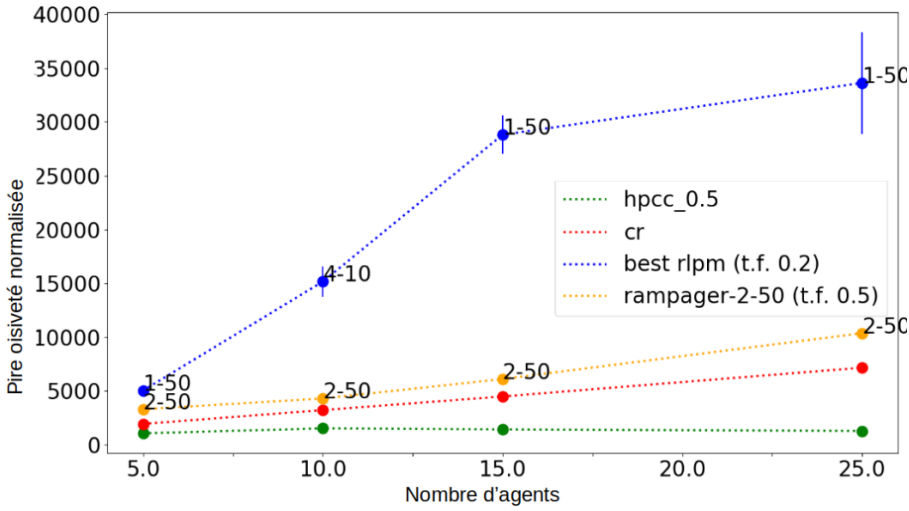

FIGURE 4.3. WI normalisé, moyenné sur 100 exécutions, de RAMPAGER 2-50 et du meilleur variant RLPM en fonction du nombre d'agents pour le graphe Islands

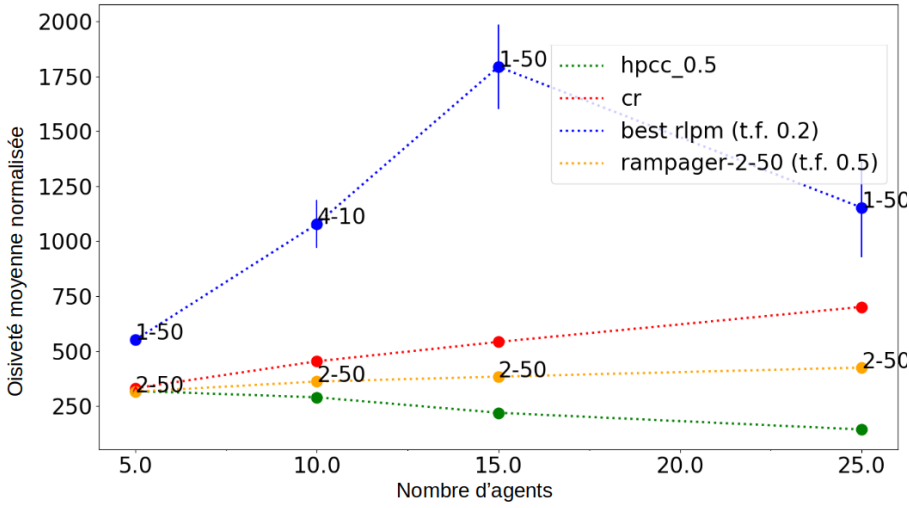

FIGURE 4.4. Iav normalisé, moyenné sur 100 exécutions, de RAMPAGER 2-50 et du meilleur variant RLPM en fonction du nombre d'agents pour le graphe Islands

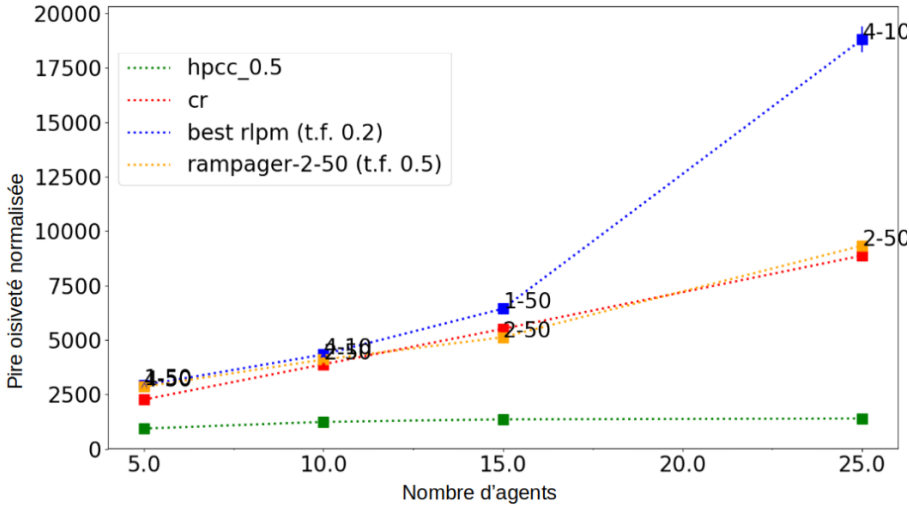

FIGURE 4.5. WI normalisé, moyenné sur 100 exécutions, de RAMPAGER 2-50 et du meilleur variant RLPM en fonction du nombre d'agents pour le graphe Map A

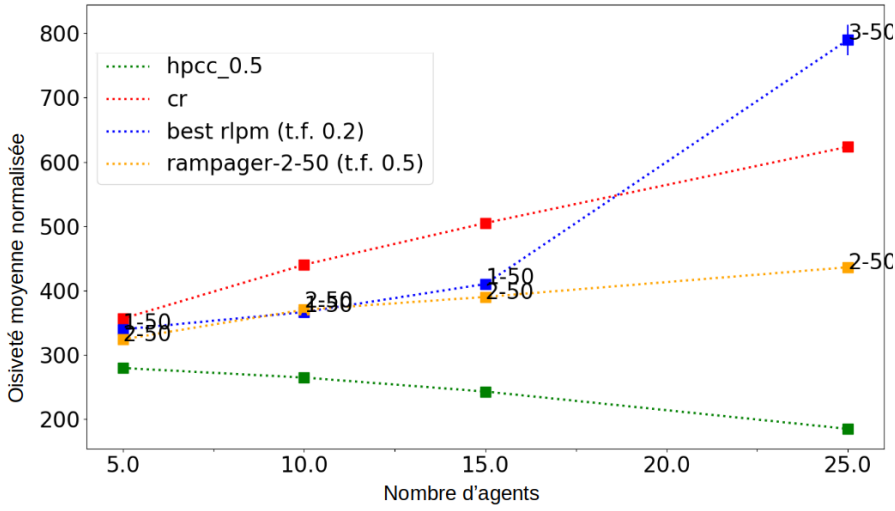

FIGURE 4.6. Iav normalisé, moyenné sur 100 exécutions, de RAMPAGER 2-50 et du meilleur variant RLPM en fonction du nombre d'agents pour le graphe Map A

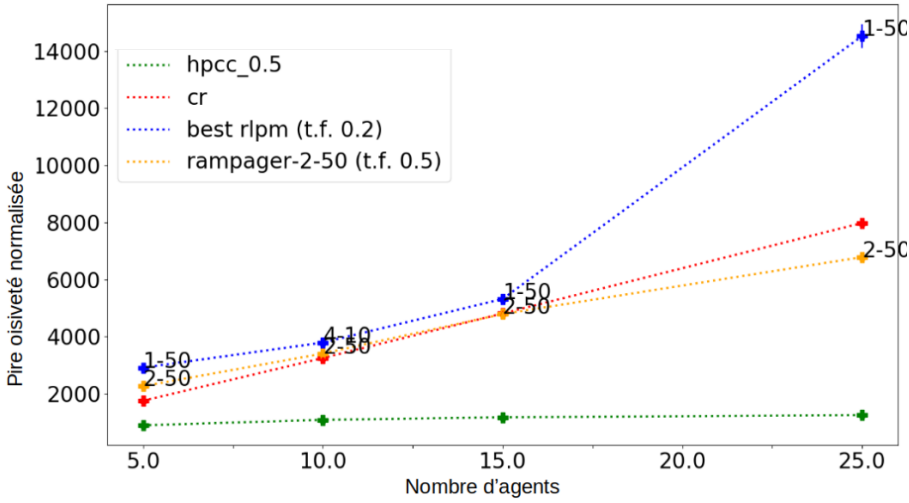

FIGURE 4.7. WI normalisé, moyenné sur 100 exécutions, de RAMPAGER 2-50 et du meilleur variant RLPM en fonction du nombre d'agents pour le graphe Grid

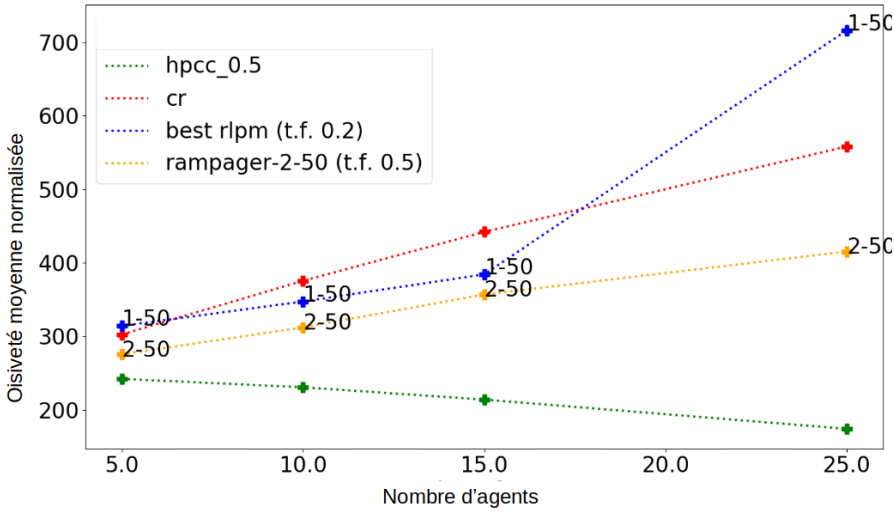

FIGURE 4.8. Iav normalisé, moyenné sur 100 exécutions, de RAMPAGER 2-50 et du meilleur variant RLPM en fonction du nombre d'agents pour le graphe Grid

est donc légèrement meilleur en moyenne que CR sur WI. Pour Iav, RAMPAGER est meilleur à la fois que CR et que le meilleur variant RLPM, en particulier pour ce dernier pour 25 agents.

La stratégie RLPM sélectionne le prochain nœud à visiter comme le résultat d'un tirage aléatoire dans une distribution de probabilité sur les nœuds, cette distribution étant générée par un réseau LSTM. RAMPAGER se base sur une procédure analytique pour l'initialisation du réseau LSTM, qui respecte les contraintes liées au graphe patrouillé, permettant ainsi que ce réseau soit configuré de telle sorte qu'il ne puisse prédire un nœud qui n'est pas voisin de celui fourni en entrée. Il est important de noter que cette procédure porte uniquement sur l'initialisation des paramètres du réseau et qu'il n'y a donc pas de contrainte sur les paramètres qui pourrait conduire à une perte de précision prédictive. L'application de cette initialisation guidée par la structure du graphe et l'utilisation d'une base de données plus pertinente que celle de RLPM a donc conduit à une stratégie, supérieure au meilleur variant RLPM. Cette stratégie présente des performances meilleures que celles de CR et est particulièrement bien adaptée aux missions pour lesquelles la communication est interdite ou impossible.

5. Conclusion et perspectives

L'intelligence artificielle distribuée, plus précisément les systèmes multiagents, associée à une approche d'apprentissage automatique ont été appliqués au cas d'étude de la patrouille multiagent qui est une référence générique pour l'étude des architectures multiagents. Une méthode utilisant l'apprentissage supervisé pour créer des stratégies multiagent distribuées à partir de traces d'exécution d'une stratégie centralisée performante a été proposée et évaluée dans le contexte de la patrouille multiagent.

L'approche considérée est l'apprentissage de la structure et des paramètres d'un prédicteur utilisé par un agent pour prédire le prochain nœud à visiter en considérant uniquement ses décisions passées.

Bien qu'une nouvelle stratégie ait été proposée et évaluée, de nombreux problèmes restent sans solution et suggèrent des pistes pour de futurs travaux à l'intersection de l'apprentissage automatique profond et de la patrouille multiagent.

HPCC présente de meilleures performances que les autres stratégies ici testées. Ceci est probablement dû au fait que, contrairement à HPCC, les autres stratégies considèrent une absence de communication et d'échange d'intentions. Une piste intéressante pour des recherches supplémentaires pourrait être de proposer des extensions de RAMPAGER pour le cas où les communications seraient contraintes, par exemple lorsque les agents sont capables de communiquer seulement lorsqu'ils sont à une distance inférieure à la portée de leur équipement radio. Ceci conduirait à une architecture neuronale plus sophistiquée et plus complexe que celle de RAMPAGER, pour laquelle les agents interagiraient et échangeraient de l'information. Finalement cette architecture pourrait être implémentée et évaluée.

Une autre voie d'exploration serait d'utiliser une fonction de coût différente de l'entropie croisée pour entraîner différemment les réseaux de neurones. Ceci pourrait

améliorer les critères d'évaluation de robustesse tels que WI, la pire oisiveté. Un système d'apprentissage basé sur de l'apprentissage par renforcement qui optimiserait directement le critère d'évaluation à optimiser, par exemple WI, pourrait aussi être considéré.

Le travail présenté dans cet article pourra également être évalué empiriquement par rapport à la stratégie à cycle unique et par rapport à l'utilisation des processus décisionnels de Markov partiellement observables et distribués.

Aussi, l'étude du problème de généralisation, que l'on retrouve pour tout modèle d'apprentissage automatique, pourrait être prolongée dans le contexte de la patrouille multiagent en évaluant la performance d'une architecture neuronale entraînée pour un nombre donné d'agents dans une situation où ce nombre est légèrement différent. Alternativement, les paramètres du réseau pourraient être appris non seulement pour un nombre donné d'agents mais aussi pour un ensemble de scénarios où ce nombre varierait légèrement.

Bibliographie

[1] A. Almeida, P. Castro, T. Menezes & G. Ramalho, « Combining idleness and distance to design heuristic agents for the patrolling task », in *II Brazilian Workshop in Games and Digital Entertainment* (Pestana Bahia Hotel, Salvador da Bahia, Brazil), 2003, p. 33-40.

[2] A. Almeida, G. Ramalho, H. Santana, P. Tedesco, T. Menezes, V. Corruble & Y. Chevaleyre, « Recent advances on multi-agent patrolling », in *Brazilian Symposium on Artificial Intelligence* (São Luis, Maranhão, Brazil), Springer, 2004, p. 474-483.

[3] A. Beynier, « Cooperative multiagent patrolling for detecting multiple illegal actions under uncertainty », in *2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*, IEEE, 2016, p. 9-16.

[4] R. K. Brayton, G. D. Hachtel, C. T. McMullen & A. L. Sangiovanni-Vincentelli, « Multiple-Valued Logic Minimization », in *Logic Minimization Algorithms for VLSI Synthesis*, Springer, 1984, p. 139-147.

[5] J. S. Bridle, « Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition », in *Neurocomputing*, Springer, 1990, p. 227-236.

[6] Y. Chevaleyre, « Theoretical analysis of the multi-agent patrolling problem », in *Intelligent Agent Technology, 2004.(IAT 2004). Proceedings. IEEE/WIC/ACM International Conference on*, IEEE, 2004, p. 302-308.

[7] D. B. D'Ambrosio, S. Goodell, J. Lehman, S. Risi & K. O. Stanley, « Multirobot behavior synchronization through direct neural network communication », in *International Conference on Intelligent Robotics and Applications*, Springer, 2012, p. 603-614.

[8] J. Duchi, E. Hazan & Y. Singer, « Adaptive subgradient methods for online learning and stochastic optimization », *Journal of machine learning research* **12** (2011), n° 7, p. 2121-2159.

[9] Y. Elmaliach, A. Shiloni & G. A. Kaminka, « Frequency-based multi-robot fence patrolling », *Bar Ilan Univ., Comput. Sci. Dept., Maverick Group, Tech. Rep* **1** (2008), p. 2008.

[10] A. Graves & J. Schmidhuber, « Framewise phoneme classification with bidirectional LSTM networks », in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 4, IEEE, 2005, p. 2047-2052.

[11] Y. Guo, L. E. Parker & R. Madhavan, « Collaborative robots for infrastructure security applications », in *Mobile robots : the evolutionary approach*, Springer, 2007, p. 185-200.

[12] S. Hochreiter & J. Schmidhuber, « Long short-term memory », *Neural computation* **9** (1997), n° 8, p. 1735-1780.

[13] A. Machado, A. Almeida, G. Ramaldo, J.-D. Zucker & A. Drogoul, « Multi-agent movement coordination in patrolling », in *Proceedings of the 3rd International Conference on Computer and Game*, 2002, p. 155-170.

[14] A. Machado, G. Ramalho, J.-D. Zucker & A. Drogoul, « Multi-agent patrolling : An empirical analysis of alternative architectures », in *International Workshop on Multi-Agent Systems and Agent-Based Simulation*, Springer, 2002, p. 155-170.

[15] T. Menezes, P. Tedesco & G. Ramalho, « Negotiator agents for the patrolling task », in *Advances in Artificial Intelligence-IBERAMIA-SBIA 2006*, Springer, 2006, p. 48-57.

[16] M. Othmani-Guibourg, A. El Fallah-Seghrouchni & J.-L. Farges, « Decentralized Multi-agent Patrolling Strategies Using Global Idleness Estimation », in *International Conference on Principles and Practice of Multi-Agent Systems*, Springer, 2018, p. 603-611.

[17] ———, « Path Generation with LSTM Recurrent Neural Networks in the context of the Multi-agent Patrolling », in *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, IEEE, 2018, p. 430-437.

[18] M. Othmani-Guibourg, A. E. F. Seghrouchni & J.-L. Farges, « LSTM Path-Maker : a new LSTM-based strategy for the multi-agent patrolling », in *52nd Annual Hawaii International Conference on System Sciences (HICSS)*, 2019, p. 616-625.

[19] C. Poulet, « Coordination dans les systèmes multi-agents : Le problème de la patrouille en système ouvert », Thèse, Paris 6, 2013.

[20] D. O. Sales, D. Feitosa, F. S. Osório & D. F. Wolf, « Multi-agent autonomous patrolling system using ANN and FSM control », in *2012 Second Brazilian Conference on Critical Embedded Systems*, IEEE, 2012, p. 48-53.

[21] P. A. Sampaio, G. Ramalho & P. Tedesco, « The gravitational strategy for the timed patrolling », in *2010 22nd IEEE International Conference on Tools with Artificial Intelligence*, vol. 1, IEEE, 2010, p. 113-120.

[22] H. Santana, G. Ramalho, V. Corruble & B. Ratitch, « Multi-agent patrolling with reinforcement learning », in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 3*, IEEE Computer Society, 2004, p. 1122-1129.

Abstract. — For over a decade, the multiagent patrol task has received a growing attention from the multiagent community due to its wide range of potential applications. However, the existing patrolling-specific algorithms based on deep learning algorithms are still in preliminary stages. In this paper, we propose to integrate a recurrent neural network as part of a multiagent patrolling strategy. Hence we proposed a formal model of an LSTM-based agent strategy named LSTM Path Maker. The LSTM network is trained over simulation traces of a coordinated strategy, then embedded on each agent of the new strategy to patrol efficiently without communicating. Finally this new LSTM-based strategy is evaluated in simulation and compared with two representative strategies : a coordinated one and a reactive one. Results indicate that the proposed strategy is better than the reactive one.

Keywords. — Multiagent Systems (MAS), Artificial Neural Networks (ANN), Long Short-Term Memory (LSTM)..

Manuscrit reçu le 28 mars 2021, révisé le 24 août 2021, accepté le 5 décembre 2021.