Actes de la 23e Conférence Nationale en Intelligence Artificielle

Isabelle Bloch

AfIA

Association française
pour l'Intelligence Artificielle

CNIA

Conférence Nationale en Intelligence Artificielle

PFIA 2020

Table des matières

Isabelle BLOCH
Éditorial 4

Comité de programme 5

S. Belabbes, S. Benferhat
Partially Preordered Inconsistent Lightweight Ontologies in Possibility Theory 6

M. F. Mbouopda, E. Mephu Nguifo
Classification des séries temporelles incertaines par transformation "Shapelet" 14

R. Châtel, A.-I. Mouaddib
MDP augmentés pour la résolution de jeux de sécurité stochastiques 22

A. Yaddaden, S. Harispe, M. Vasquez, M. Buljubašić
Apprentissage automatique pour l'optimisation combinatoire : Éude du problème du voyageur de commerce 30

R. Fontaine, N. Aky, R. Courdier, D. Payet
Vers une utilisation éco responsable des objets connectés par la mutualisation de leurs composants physiques : Une approche basée sur le concept d'artefact 38

Éditorial

Cette année, étrange à bien des égards, ne permet pas les rencontres et échanges habituels pendant la conférence. Les processus de soumission et de relecture ont pu cependant se dérouler normalement, et ces actes contiennent les cinq articles acceptés à la conférence CNIA 2020, hébergée par la plateforme PFIA. Ils sont présentés pendant la conférence, à distance.

La conférence comporte également des présentations d'articles acceptés à AAAI, ECAI, IJCAI, écrits par des équipes françaises. Ces articles pourront être trouvés dans les actes des conférences correspondantes.

Dans une période où il est beaucoup question de science, comme rarement auparavant, et où l'intelligence artificielle est souvent mise en avant, la conférence CNIA, et plus largement la plateforme PFIA, prend toute son importance, en illustrant les diverses facettes de l'IA et les avancées de la recherche dans ce vaste domaine.

Bonne conférence !

Isabelle BLOCH

Comité de programme

Président

— Isabelle Bloch, LTCI, Télécom Paris

Membres

— Meghyn Bienvenu (LABRI, Université de Bordeaux)
— Olivier Boissier (Ecole des Mines de Saint-Etienne)
— Elise Bonzon (LIPADE, Paris Descartes)
— Jean Charlet (LIMICS)
— Sylvie Coste-Marquis (CRIL, Université d'Artois)
— Célia Da Costa Pereira (Université de Nice)
— Séverine Dubuisson (LIS, université Aix-Marseille)
— Florence Dupin De Saint Cyr (IRIT, Toulouse)
— Pietro Gori (LTCI, Télécom Paris)
— Sébastien Konieczny (CRIL, CNRS)
— Florence Le Ber (ENGEES, Université de Strasbourg)
— Marie-Jeanne Lesot (LIP6, Sorbonne Université)
— Christophe Marsala (LIP6, Sorbonne Université)
— Nicolas Maudet (LIP6, Sorbonne Université)
— Abdel-Illah Mouaddib (GREYC, Université de Caen)
— Alasdair Newson (LTCI, Télécom Paris)
— Odile Papini (Université Aix-Marseille)
— Fabian Suchanek (LTCI, Télécom Paris)
— Bruno Zanuttini (GREYC, Université de Caen)
— Pierre Zweigenbaum (LIMSI, CNRS)

Ontologies Légères Inconsistantes Partiellement Pré-ordonnées en Théorie des Possibilités *

Sihem Belabbes, Salem Benferhat

CRIL, Univ. Artois & CNRS, France

{belabbes, benferhat}@cril.fr

Résumé

Nous proposons une extension de la logique DL-Lite possibiliste standard au cas où la base de connaissances est partiellement pré-ordonnée. Nous considérons que la base assertionnelle ABox est représentée par une base pondérée symbolique et nous supposons qu'un ordre partiel strict est appliqué aux poids. Nous introduisons une méthode traitable pour calculer une réparation unique pour une ABox pondérée partiellement pré-ordonnée. L'idée de base consiste à calculer cette réparation à partir des réparations possibilistes associées aux bases compatibles d'une ABox partiellement pré-ordonnée, qui intuitivement encodent toutes les extensions possibles d'un pré-ordre partiel. Nous proposons une caractérisation équivalente à travers la notion d'assertions π-acceptées, ce qui garantit que le calcul de la réparation possibiliste partielle se fasse en temps polynomial.

Mots-clés

Base de Connaissances Inconsistante, Logique DL-Lite, Théorie des Possibilités.

Abstract

This paper investigates an extension of standard possibilistic DL-Lite to the case where the knowledge base is partially preordered. We consider the assertional base ABox to be represented as a symbolic weighted base and we assume a strict partial order is applied to the weights. We introduce a tractable method for computing a single repair for a partially preordered weighted ABox. Basically, this repair is computed from possibilistic repairs associated with compatible bases of a partially preordered ABox, which intuitively encode all possible extensions of a partial order. We provide an equivalent characterization using the notion of π-accepted assertions, which ensures that the partial possibilistic repair is computed in polynomial time.

Keywords

Inconsistent Knowledge Base, DL-Lite, Possibility Theory.

*Cet article a été présenté à la conférence *International Symposium on Artificial Intelligence and Mathematics* (ISAIM 2020), Fort Lauderdale, USA, 6–8 janvier 2020.

1 Introduction

La théorie des possibilités a été largement étudiée depuis le travail précurseur de Zadeh [32]. Il s'agit d'une théorie de l'incertitude qui permet de gérer des informations incomplètes, incertaines, qualitatives et munies de priorités, ainsi que de raisonner en présence d'inconsistance [17, 19]. La théorie des possibilités a des liens forts avec les fonctions ordinales conditionnelles [27] ainsi qu'avec les fonctions de croyances consonantes [18, 14, 26].

La logique possibiliste standard [16] fournit un cadre naturel pour raisonner à partir d'informations inconsistantes, incertaines, et munies de priorités ordonnées selon un préordre total. En substance, la logique possibiliste standard est une logique pondérée qui utilise des formules de logique propositionnelle auxquelles sont attachés des poids. Ces derniers appartiennent à l'intervalle unitaire $[0, 1]$, considéré comme une échelle ordinale. Un poids (ou degré) constitue une borne inférieure sur le degré de certitude (ou de priorité) d'une formule.

Un domaine de recherche qui jouit d'un interêt considérable est celui de la gestion de l'inconsistance dans les ontologies formelles, en particulier celles spécifiées dans les fragments légers des logiques des descriptions, à savoir DL-Lite. Par exemple, des extensions floues ont été proposées pour les logiques des descriptions [11, 9, 29] et pour DL-Lite [24, 28]. De plus, des extensions possibilistes des logiques des descriptions [15, 25] ainsi que des extensions probabilistes [1, 10, 23] ont également été proposées.

Récemment, un cadre pour DL-Lite possibiliste a été proposé [6]. L'idée consiste à attacher des poids aux assertions de la ABox pour refléter le fait que certaines informations peuvent être plus fiables que d'autres. Une caractéristique intéressante de DL-Lite possibiliste est que le *query answering* est traitable. Donc, le fait que l'expressivité de DL-Lite standard soit augmentée par un pré-ordre total sur les assertions n'engendre pas de coût supplémentaire.

Cependant, dans de nombreuses applications et notamment les ontologies, la fiabilité est définie de manière partielle seulement. Cela est souvent dû au fait que l'information soit obtenue de sources multiples qui ne partagent pas la même opinion. Cela implique l'application d'un ordre partiel au lieu d'un ordre total sur les poids affectés aux formules ou les assertions. Il convient de mentionner que la

relation d'ordre appliquée aux poids est un ordre partiel strict, c'est-à-dire qu'il n'y a pas d'égalité entre les poids. En revanche, la relation d'ordre qui résulte de l'affectation de ces poids aux assertions est un pré-ordre partiel sur les assertions, puisque le même poids peut être affecté à plus d'une assertion (les égalités de poids entre les assertions sont permises). Dans ce cas, la ABox correspondante est partiellement pré-ordonnée.

Des extensions de la logique possibiliste standard ont été proposées pour permettre de raisonner avec des informations partiellement pré-ordonnées. Dans [7], les notions fondamentales de la logique possibiliste standard comme l'inférence possibiliste ont été revisitées. Il s'agit d'affecter à des formules de logique propositionnelle des degrés appartenant à une échelle d'incertitude partiellement ordonnée, au lieu de l'intervalle unitaire $[0, 1]$. Dans [5, 30], l'idée d'affecter des poids symboliques partiellement ordonnés à des croyances a également été étudiée en détail. L'inconvénient principal de telles approches est que leur complexité computationnelle est coûteuse (Δ_p^2-dure), ce qui les rend inappropriées dans des situations où les réponses aux requêtes doivent être obtenues efficacement.

Etant donné ce contexte, nous nous intéressons à proposer une extension de DL-Lite possibiliste standard [6] au cas où les connaissances sont partiellement pré-ordonnées, sans augmenter sa complexité. L'idée de gestion de l'inconsistance dans des ontologies légères partiellement pré-ordonnées a été récemment investiguée dans [3]. Une méthode efficace, appelée "Elect", a été proposée pour calculer une réparation unique pour une ABox partiellement pré-ordonnée. Elect offre une extension de la sémantique bien connue *Intersection of ABox Repair* (IAR) [22] pour le cas où la ABox est partiellement pré-ordonnée. Il s'agit d'interpréter une ABox partiellement pré-ordonnée comme une famille de ABox totalement pré-ordonnées, et pour lesquelles des réparations peuvent être calculées. Ainsi, l'intersection de ces réparations produit une réparation unique pour la ABox partiellement pré-ordonnée.

Une question naturelle qui se pose concerne le fait de pouvoir ou non étendre le cadre DL-Lite possibiliste à des ordres partiels de manière traitable (dans l'esprit de la méthode Elect). Cet article donne une réponse positive.

Pour cela, nous considérons une famille de ABox compatibles (qui sont des ABox DL-Lite possibilistes) et calculons la réparation possibiliste associée à chaque base compatible. Enfin, nous obtenons une réparation unique pour la ABox pondérée partiellement pré-ordonnée à partir de l'intersection de toutes les réparations possibilistes. Notre contribution principale est la proposition d'une caractérisation équivalente qui identifie les assertions acceptées, appelées π-acceptées, sans calculer explicitement toutes les bases compatibles. Nous montrons que l'ensemble des assertions π-acceptées est consistant et qu'il peut être calculé en temps polynomial. Par ailleurs, nous montrons que lorsque la relation de préférence est un ordre total, la réparation obtenue est en fait la réparation possibiliste calculée en DL-Lite possibiliste standard.

Cet article est structuré comme suit. Nous rappelons brièvement les principes de base de DL-Lite standard dans la Section 2, suivie par son extension à la logique possibiliste dans la Section 3. Nous introduisons notre méthode traitable pour calculer une réparation pour une ABox pondérée partiellement pré-ordonnée dans la Section 4. Nous concluons par une discussion des travaux futurs.

2 La Logique des Descriptions DL-Lite

La logique des descriptions DL-Lite [13] est une famille de langages de représentation des connaissances qui ont gagné en popularité dans de nombreux domaines d'application comme la formalisation d'ontologies légères, grâce à leur pouvoir expressif et leurs bonnes propriétés computationnelles. Par exemple, *query answering* à partir d'une base de connaissances DL-Lite peut s'effectuer de manière efficace en utilisant la reformulation de requêtes [21].

Nous présentons le dialecte DL-Lite$_R$ de DL-Lite. Nous considérons un ensemble fini de *noms de concepts* C, un ensemble fini de *noms de rôles* R et un ensemble fini de *noms d'individus* I, tels que C, R et I sont mutuellement disjoints. Soient $A \in \mathsf{C}$, $P \in \mathsf{R}$ et $P^- \in \mathsf{R}$ *l'inverse* de P. Le langage DL-Lite$_R$ est défini selon les règles suivantes :

$$\begin{array}{ll} R \longrightarrow P \mid P^- & E \longrightarrow R \mid \neg R \\ B \longrightarrow A \mid \exists R & C \longrightarrow B \mid \neg B \end{array}$$

où R est un *rôle de base*, E est un *rôle complexe*. De même, B est un *concept de base*, C est un *concept complexe*.

En ce qui concerne la sémantique, une interprétation est un tuplet $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, où $\Delta^{\mathcal{I}} \neq \emptyset$ et $\cdot^{\mathcal{I}}$ est une fonction d'interprétation qui associe un nom de concept A à un sous-ensemble $A^{\mathcal{I}}$ de $\Delta^{\mathcal{I}}$, un nom de rôle P à une relation binaire $P^{\mathcal{I}}$ sur $\Delta^{\mathcal{I}}$, et un nom d'individu a à un élément du domaine $\Delta^{\mathcal{I}}$. Nous avons également :

$$\begin{array}{l} (P^-)^{\mathcal{I}} = \{(y, x) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (x, y) \in P^{\mathcal{I}}\}; \\ (\exists R)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} \text{ t.q. } (x, y) \in R^{\mathcal{I}}\}; \\ (\neg B)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus B^{\mathcal{I}}; \\ (\neg R)^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \setminus R^{\mathcal{I}}. \end{array}$$

Un *axiome d'inclusion* sur des concepts (resp. des rôles) est un énoncé de la forme $B \sqsubseteq C$ (resp. $R \sqsubseteq E$). Une inclusion de concept est dite *axiome négatif d'inclusion* si elle contient le symbole "¬" à droite de l'inclusion, sinon elle est dite *axiome positif d'inclusion*. Une *assertion* est un énoncé de la forme $A(a)$ ou $P(a, b)$, où $a, b \in \mathsf{I}$.

Une interprétation $\mathcal{I}$ *satisfait* un axiome d'inclusion $B \sqsubseteq C$ (resp. $R \sqsubseteq E$), noté $\mathcal{I} \Vdash B \sqsubseteq C$ (resp. $\mathcal{I} \Vdash R \sqsubseteq E$), si $B^{\mathcal{I}} \subseteq C^{\mathcal{I}}$ (resp. $R^{\mathcal{I}} \subseteq E^{\mathcal{I}}$). $\mathcal{I}$ *satisfait* une assertion $A(a)$ (resp. $P(a, b)$), noté $\mathcal{I} \Vdash A(a)$ (resp. $\mathcal{I} \Vdash P(a, b)$), si $a^{\mathcal{I}} \in A^{\mathcal{I}}$ (resp. $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in P^{\mathcal{I}}$).

Une base de connaissances DL-Lite$_R$ est une paire $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, où $\mathcal{T}$ est un ensemble fini d'axiomes d'inclusion, dit TBox, et $\mathcal{A}$ est un ensemble fini d'assertions, dit ABox. Une interprétation $\mathcal{I}$ est un *modèle* d'une TBox $\mathcal{T}$ (resp. ABox $\mathcal{A}$), noté $\mathcal{I} \Vdash \mathcal{T}$ (resp. $\mathcal{I} \Vdash \mathcal{A}$), si $\mathcal{I} \Vdash \alpha$ pour tout

α dans $\mathcal{T}$ (resp. dans $\mathcal{A}$). L'interprétation $\mathcal{I}$ est un modèle de $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ si $\mathcal{I} \Vdash \mathcal{T}$ et $\mathcal{I} \Vdash \mathcal{A}$.
Une base de connaissances (KB) $\mathcal{K}$ est *consistante* si elle admet au moins un modèle, sinon elle est *inconsistante*.
Une TBox $\mathcal{T}$ est *incohérente* s'il y a un nom de concept $A \in \mathsf{C}$ tel que A est vide dans chaque modèle de $\mathcal{T}$, sinon elle est *cohérente*.
Dans la suite de cet article, nous notons DL-Lite$_R$ simplement par DL-Lite. Pour de plus amples détails sur la famille DL-Lite de logiques des descriptions, nous invitons le lecteur à consulter [13].
Tout au long de cet article, nous utiliserons l'exemple suivant et l'adapterons au besoin.

Exemple 1 *Soit $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ une KB DL-Lite.*
Soit $\mathcal{T} = \{A \sqsubseteq \neg B, B \sqsubseteq \neg C, C \sqsubseteq \neg D\}$ une TBox.
Soit $\mathcal{A} = \{A(a),\ A(b),\ B(a),\ B(c),\ C(a),\ C(b),\ D(a),\ D(b),\ D(c),\ E(a)\}$ une ABox non-ordonnée (sans poids sur les assertions).
Il est aisé de vérifier que $\mathcal{K}$ est inconsistante. Par exemple, l'individu 'a' appartient aux concepts A et B. Ceci contredit l'axiome négatif $A \sqsubseteq \neg B$. □

Plusieurs stratégies ont été proposées pour raisonner avec des KB inconsistantes (e.g. [2, 12, 8, 31]). L'idée principale repose sur le calcul de réparations pour la ABox, où une réparation est définie comme un sous-ensemble de la ABox maximal (pour l'inclusion ensembliste) qui est consistant avec la TBox. Parmi ces stratégies, on trouve la sémantique *ABox Repair* (AR) [22] dans laquelle une réponse à une requête est valide si elle s'ensuit de chaque réparation. On trouve également la sémantique *Intersection of ABox Repair* (IAR) [22] qui interroge une sous-base consistante de la ABox obtenue de l'intersection de toutes les réparations. Par ailleurs, la sémantique dite *non-contestée* [4] revient à une version prioritaire de la sémantique IAR.
Nous nous intéressons aux réparations possibilistes, en particulier dans le cas de bases de connaissances partiellement pré-ordonnées. Nous rappelons d'abord les notions de base de DL-Lite possibiliste standard.

3 Bases de Connaissances DL-Lite Possibilistes

Les logiques des descriptions possibilistes [20, 15] sont des extensions des logiques des descriptions standard basées sur la théorie des possibilités, qui permettent de raisonner avec des connaissances incertaines et inconsistantes. Des extensions DL-Lite possibilistes [6] ont récemment été proposées pour les fragments légers DL-Lite. L'idée principale consiste à affecter des degrés de priorité (ou poids) aux axiomes de la TBox et aux assertions de la ABox pour exprimer leur relative certitude dans une base de connaissances inconsistante. Le degré d'inconsistance d'une base de connaissances peut alors être calculé à partir de ces poids, permettant une inférence possibiliste.
Dans cette section, nous considérons une base de connaissances DL-Lite possibiliste finie, $\mathcal{WK} = \langle \mathcal{T}, \mathcal{WA} \rangle$, appelée KB pondérée. Nous supposons que les axiomes de la TBox $\mathcal{T}$ sont complètement certains alors que les assertions de la ABox pondérée $\mathcal{WA}$ sont munies de degrés de priorité définis sur l'intervalle unitaire $]0, 1]$, comme suit :

$$\mathcal{WA} = \{(f, \alpha) \mid f \text{ est une assertion DL-Lite, } \alpha \in]0, 1]\}.$$

Nous supposons qu'un degré de priorité unique α est affecté à chaque assertion f.
Les assertions de $\mathcal{WA}$ avec un degré de priorité $\alpha = 1$ sont considérées complètement certaines et ne sont pas discutables, alors que les assertions avec un degré de priorité $0 < \alpha < 1$ sont dites quelque peu certaines. Les assertions avec les degrés de priorité les plus élevés sont plus certaines que celles avec des degrés de priorité plus bas. Nous ignorons les assertions avec un degré $\alpha = 0$, donc seules les assertions quelque peu certaines sont énoncées explicitement.
Dans la suite de cet article, pour une base assertionnelle pondérée donnée $\mathcal{WB}$, nous notons par $\mathcal{WB}^*$ l'ensemble des assertions obtenues après leur avoir enlevé les degrés de priorité. Nous notons $\mathcal{WK}^*$ la KB dont la ABox est l'ensemble d'assertions $\mathcal{WB}^*$.
Par ailleurs, nous supposons que la TBox $\mathcal{T}$ est cohérente et n'évolue pas. Cependant les assertions de la ABox pondérée $\mathcal{WA}$ peuvent être discutables. Cela veut dire qu'il peut y avoir des conflits entre les assertions par rapport aux axiomes de $\mathcal{T}$. Dans ce cas, la KB pondérée $\mathcal{WK}$ est dite inconsistante.
Un conflit assertionnel est défini comme un sous-ensemble d'assertions minimal (pour l'inclusion ensembliste) qui est inconsistant avec la TBox, où l'inconsistance est comprise dans le sens de DL-Lite standard. Formellement :

Définition 1 *Soit $\mathcal{WK} = \langle \mathcal{T}, \mathcal{WA} \rangle$ une KB pondérée.*
Une sous-base $\mathcal{C} \subseteq \mathcal{WA}$ est un conflit assertionnel dans $\mathcal{WK}$ ssi :

— *$\langle \mathcal{T}, \mathcal{C}^* \rangle$ est inconsistant, et*
— *$\forall f \in \mathcal{C}^*$, $\langle \mathcal{T}, \mathcal{C}^* \setminus \{f\} \rangle$ est consistant.*

Nous notons $\mathcal{C}(\mathcal{WA})$ l'ensemble de tous les conflits assertionnels de $\mathcal{WA}$. Il est important de noter que le calcul de l'ensemble des conflits se fait en temps polynomial en DL-Lite [12]. De plus, nous supposons qu'il n'y a pas d'assertion $f \in \mathcal{WA}^*$ telle que $\langle \mathcal{T}, \{f\} \rangle$ est inconsistant. Donc les conflits assertionnels dans des bases de connaissances DL-Lite cohérentes sont binaires [12], i.e., $\forall \mathcal{C} \in \mathcal{C}(\mathcal{WA}), |\mathcal{C}| = 2$. Nous notons un conflit assertionnel comme une paire $\mathcal{C}_{ij} = \{(f_i, \alpha_i), (f_j, \alpha_j)\}$, où $(f_i, \alpha_i), (f_j, \alpha_j) \in \mathcal{WA}$. Les assertions $f_i, f_j \in \mathcal{WA}^*$ sont alors dites conflictuelles par rapport à $\mathcal{T}$.

Exemple 2 *Continuons l'Exemple 1 et munissons la ABox de poids. Soit $\mathcal{WK} = \langle \mathcal{T}, \mathcal{WA} \rangle$ la KB pondérée correspondante, avec $\mathcal{T} = \{A \sqsubseteq \neg B, B \sqsubseteq \neg C, C \sqsubseteq \neg D\}$, et la ABox pondérée est :*

$$\mathcal{WA} = \left\{ \begin{array}{l} (A(a), 0{,}9), (A(b), 0{,}9), (B(c), 0{,}8), \\ (E(a), 0{,}7), (D(b), 0{,}6), (C(a), 0{,}5), \\ (D(a), 0{,}4), (B(a), 0{,}3), (D(c), 0{,}3), \\ (C(b), 0{,}1) \end{array} \right\}$$

L'ensemble des conflits assertionnels de $\mathcal{WA}$ est :

$$\mathcal{C}(\mathcal{WA}) = \left\{ \begin{array}{l} \{(A(a),0,9),(B(a),0{,}3)\}, \\ \{(D(b),0{,}6),(C(b),0{,}1)\}, \\ \{(C(a),0{,}5),(D(a),0{,}4)\}, \\ \{(C(a),0{,}5),(B(a),0{,}3)\} \end{array} \right\}$$

□

Nous nous intéressons à calculer le degré de priorité le plus élevé où l'inconsistance est rencontrée dans la ABox, appelé degré d'inconsistance. Formellement :

Définition 2 *Soit $\mathcal{WK} = \langle \mathcal{T}, \mathcal{WA} \rangle$ une KB pondérée. Considérons un poids $\beta \in]0,1]$.*

- *Soit $\mathcal{A}^{\geq\beta} = \{f \mid (f,\alpha) \in \mathcal{WA}, \alpha \geq \beta\}$ la β-coupe de la base assestionnelle pondérée $\mathcal{WA}$.*
- *Soit $\mathcal{A}^{>\beta} = \{f \mid (f,\alpha) \in \mathcal{WA}, \alpha > \beta\}$ la β-coupe stricte de $\mathcal{WA}$.*

Le degré d'inconsistance de $\mathcal{WA}$, noté $Inc(\mathcal{WA})$, est :

$$Inc(\mathcal{WA}) = \left\{ \begin{array}{ll} 0 & \text{ssi } \langle \mathcal{T}, \mathcal{WA}^* \rangle \text{ est consistant} \\ \beta & \text{ssi } \langle \mathcal{T}, \mathcal{A}^{\geq\beta} \rangle \text{ est inconsistant} \\ & \text{et } \langle \mathcal{T}, \mathcal{A}^{>\beta} \rangle \text{ est consistant} \end{array} \right.$$

Exemple 3 *Il est aisé de vérifier que pour $\beta = 0{,}4$:*

- *$\mathcal{A}^{>\beta} = \{A(a), A(b), B(c), E(a), D(b), C(a)\}$ est consistant avec $\mathcal{T}$, alors que*
- *$\mathcal{A}^{\geq\beta} = \mathcal{A}^{>\beta} \cup \{D(a)\}$ est inconsistant avec $\mathcal{T}$.*

Donc $Inc(\mathcal{WA}) = 0{,}4$. □

Le degré d'inconsistance est un moyen de restorer la consistance d'une ABox inconsistante. En effet, seules les assertions ayant un degré de certitude strictement supérieur au degré d'inconsistance sont incluses dans la réparation possibiliste, ce qui garantit que les résultats soient sains. Par ailleurs, cette méthode a l'avantage d'être efficace. En effet, pour une ABox pondérée $\mathcal{WA}$, $Inc(\mathcal{WA})$ peut être calculé de manière traitable en utilisant $log_2(n)$ (où n est le nombre de poids différents dans $\mathcal{WA}$) vérifications de consistance d'une ABox classique (sans les poids).

La réparation possibiliste [1], appelée π-réparation, est définie formellement comme suit :

Définition 3 *Soit $\mathcal{WK} = \langle \mathcal{T}, \mathcal{WA} \rangle$ une KB pondérée et $Inc(\mathcal{WA})$ le degré d'inconsistance.*
La π-réparation de $\mathcal{WA}$, notée $\pi(\mathcal{WA})$, est :

$$\pi(\mathcal{WA}) = \{f \mid (f,\alpha) \in \mathcal{WA}, \alpha > Inc(\mathcal{WA})\}.$$

La π-réparation $\pi(\mathcal{WA})$ est composée des assertions de $\mathcal{WA}$ ayant un degré de priorité strictement supérieur à $Inc(\mathcal{WA})$. Donc par la Définition 2, $\pi(\mathcal{WA})$ est consistant avec $\mathcal{T}$. Notons que les degrés de priorité sont omis dans $\pi(\mathcal{WA})$. D'autre part, lorsque $\mathcal{WK}$ est consistant (i.e., $Inc(\mathcal{WA}) = 0$), alors $\pi(\mathcal{WA})$ revient à $\mathcal{WA}^*$ (i.e., la ABox sans les degrés de priorité).

1. Dans la littérature, une réparation est souvent définie comme un sous-ensemble d'assertions maximal consistant. Ici, nous utilisons le terme réparation pour désigner un sous-ensemble d'assertions consistant.

Exemple 4 *La π-réparation de $\mathcal{WA}$ est :*
$\pi(\mathcal{WA}) = \{A(a), A(b), B(c), C(a), D(b), E(a)\}$. □

Jusqu'à présent, nous avons considéré des ABox pondérées telles que les poids attachés aux assertions peuvent être utilisés pour induire un pré-ordre total sur la ABox. Dans la prochaine section, nous étendons les résultats au cas où les poids sont partiellement ordonnés.

4 Bases de Connaissances Partiellement Pré-Ordonnées

Dans cette section, nous supposons toujours que les axiomes sont complètement fiables. Cependant, les degrés de priorité associés aux assertions sont partiellement ordonnés, i.e., les niveaux de fiabilité associés aux assertions peuvent être incomparables les uns avec les autres. Ceci se produit en général lorsque les informations sont obtenues de sources multiples. Donc, il n'est parfois pas possible de décider sur la préférence entre deux assertions f_i et f_j, car selon une source, l'assertion f_i devrait être préférée à f_j, alors que selon une autre source, cela devrait être le contraire. Nous rappelons que les assertions sont partiellement pré-ordonnées.

A présent, nous introduisons la notion d'échelle d'incertitude partiellement ordonnée $\mathbb{L} = (U, \rhd)$, définie sur un ensemble non-vide d'éléments $U = \{u_1, \ldots, u_n\}$, appelé ensemble partiellement ordonné, et un ordre partiel strict $\rhd$ (relation irréflexive et transitive).

Intuitivement, les éléments de U représentent des degrés de priorité appliqués aux assertions. Nous supposons que U contient un élément spécial noté $\mathbb{1}$, où $\mathbb{1}$ représente la certitude complète, tel que pour tout $u_i \in U \setminus \{\mathbb{1}\}$, $\mathbb{1} \rhd u_i$. De plus, si $u_i \not\rhd u_j$ et $u_j \not\rhd u_i$, nous disons que u_i et u_j sont incomparables et le notons par $u_i \bowtie u_j$.

Une KB DL-Lite partiellement pré-ordonnée est un triplet $\mathcal{K}_\rhd = \langle \mathcal{T}, \mathcal{A}_\rhd, \mathbb{L} \rangle$, avec $\mathcal{A}_\rhd = \{(f_i, u_i) \mid f_i$ est une assertion DL-Lite, $u_i \in U\}$ (où un poids unique u_i est affecté à chaque assertion f_i) et $\mathbb{L} = (U, \rhd)$.

Etant données deux assertions $(f_i, u_i), (f_j, u_j) \in \mathcal{A}_\rhd$, nous abuserons parfois la notation et écrirons $f_i \rhd f_j$ pour signifier $u_i \rhd u_j$, et écrirons $f_i \bowtie f_j$ pour signifier $u_i \bowtie u_j$.

4.1 Bases compatibles

Une façon naturelle de représenter une ABox partiellement pré-ordonnée est de considérer l'ensemble de toutes les ABox compatibles, à savoir celles qui préservent l'ordre de préférence strict entre les assertions, dans l'esprit des résultats établis en logique propositionelle [7].

Définition 4 *Soit $\mathbb{L} = (U, \rhd)$ une échelle d'incertitude. Soit $\mathcal{K}_\rhd = \langle \mathcal{T}, \mathcal{A}_\rhd, \mathbb{L} \rangle$ une KB DL-Lite partiellement pré-ordonnée.*
Soit $\mathcal{WK} = \langle \mathcal{T}, \mathcal{WA} \rangle$ une KB pondérée, obtenue de $\mathcal{K}_\rhd$ en remplaçant chaque élément u par un nombre réel unique dans l'intervalle $]0,1]$, où :

$$\mathcal{WA} = \{(f,\alpha) \mid (f,u) \in \mathcal{A}_\rhd, \alpha \in]0,1]\}.$$

La ABox pondérée $\mathcal{WA}$ est dite compatible avec $\mathcal{A}_\rhd$ si :

$$\forall (f_i, \alpha_i), (f_j, \alpha_j) \in \mathcal{WA},\ \textit{si}\ f_i \rhd f_j\ \textit{alors}\ \alpha_i > \alpha_j.$$

Notons que les bases compatibles ne sont pas uniques. En fait, il en existe une infinité. En réalité, les valeurs des poids importent peu, seul l'ordre entre les assertions est important, tel que démontré plus loin.

Exemple 5 *Soit $\mathbb{L} = (U, \rhd)$ une échelle d'incertitude définie sur l'ensemble $U = \{u_1, \ldots, u_4\}$, tel que : $u_4 \rhd u_3 \rhd u_1$, $u_4 \rhd u_2 \rhd u_1$ et $u_2 \bowtie u_3$. Soit $\mathcal{K}_\rhd = \langle \mathcal{T}, \mathcal{A}_\rhd, \mathbb{L} \rangle$ une KB partiellement pré-ordonnée. Soit $\mathcal{T} = \{A \sqsubseteq \neg B, B \sqsubseteq \neg C, C \sqsubseteq \neg D\}$.*

$$\mathcal{A}_\rhd = \left\{ \begin{array}{l} (A(a), u_4), (A(b), u_4), (B(c), u_4), \\ (C(a), u_3), (D(b), u_3), (E(a), u_3), \\ (C(b), u_2), \\ (B(a), u_1), (D(a), u_1), (D(c), u_1) \end{array} \right\}$$

Soit $\{0{,}2, 0{,}4, 0{,}6, 0{,}8\}$ un ensemble de poids. Les bases $\mathcal{WA}_1, \mathcal{WA}_2$ et $\mathcal{WA}_3$ sont compatibles avec $\mathcal{A}_\rhd$:

$$\mathcal{WA}_1 = \left\{ \begin{array}{l} (A(a), 0{,}8), (A(b), 0{,}8), (B(c), 0{,}8), \\ (C(a), 0{,}6), (D(b), 0{,}6), (E(a), 0{,}6), \\ (C(b), 0{,}4), \\ (B(a), 0{,}2), (D(a), 0{,}2), (D(c), 0{,}2) \end{array} \right\}$$

$$\mathcal{WA}_2 = \left\{ \begin{array}{l} (A(a), 0{,}8), (A(b), 0{,}8), (B(c), 0{,}8), \\ (C(b), 0{,}6), \\ (C(a), 0{,}4), (D(b), 0{,}4), (E(a), 0{,}4), \\ (B(a), 0{,}2), (D(a), 0{,}2), (D(c), 0{,}2) \end{array} \right\}$$

$$\mathcal{WA}_3 = \left\{ \begin{array}{l} (A(a), 0{,}8), (A(b), 0{,}8), (B(c), 0{,}8), \\ (C(a), 0{,}6), (D(b), 0{,}6), (E(a), 0{,}6), (C(b), 0{,}6), \\ (B(a), 0{,}4), (D(a), 0{,}4), (D(c), 0{,}4) \end{array} \right\}$$

Pour $\mathcal{WA}_3$, tout sous-ensemble de trois poids qui préservent l'ordre entre les assertions convient. □

4.2 Calcul de la réparation partiellement pré-ordonnée

Nous cherchons à calculer une réparation unique pour une ABox partiellement pré-ordonnée. Cependant, la famille de ABox compatibles est infinie, ce qui signifie que le choix d'une ABox compatible parmi d'autres serait arbitraire. Une meilleure approche pour calculer la réparation partiellement pré-ordonnée consiste à :

(i) définir les ABox compatibles (Définition 4),

(ii) calculer la π-réparation associée à chaque ABox compatible (Définition 3), et finalement

(iii) prendre l'intersection de toutes les π-réparations.

Cela garantit que les résultats soient sains, puisque toutes les ABox compatibles seraient prises en compte.

Définition 5 *Soit $\mathbb{L} = (U, \rhd)$ une échelle d'incertitude. Soit $\mathcal{K}_\rhd = \langle \mathcal{T}, \mathcal{A}_\rhd, \mathbb{L} \rangle$ une KB DL-Lite partiellement pré-ordonnée. Soit $\mathcal{F}(\mathcal{A}_\rhd) = \{\pi(\mathcal{WA}) \mid \mathcal{WA}$ est compatible avec $\mathcal{A}_\rhd\}$ l'ensemble des π-réparations associées à toutes les bases compatibles de $\mathcal{A}_\rhd$ (données par la Définition 3).*

La réparation partiellement pré-ordonnée de $\mathcal{A}_\rhd$, notée $\pi(\mathcal{A}_\rhd)$, est donnée par :

$$\pi(\mathcal{A}_\rhd) = \bigcap \{\pi(\mathcal{WA}) \mid \pi(\mathcal{WA}) \in \mathcal{F}(\mathcal{A}_\rhd)\}.$$

En d'autres termes, $\pi(\mathcal{A}_\rhd) = \{f \mid (f, u) \in \mathcal{A}_\rhd, \forall \mathcal{WA}$ compatible avec $\mathcal{A}_\rhd, f \in \pi(\mathcal{WA})\}$.

Les poids sont omis dans la réparation partiellement pré-ordonnée $\pi(\mathcal{A}_\rhd)$, de la même façon que pour la π-réparation $\pi(\mathcal{WA})$.
L'ensemble $\mathcal{F}(\mathcal{A}_\rhd)$ est infini puisqu'il y a un nombre infini de ABox pondérées qui sont compatibles avec la ABox partiellement pré-ordonnée $\mathcal{A}_\rhd$. Cependant, il n'est pas nécessaire de considérer toutes les bases compatibles de $\mathcal{A}_\rhd$ pour calculer la réparation partiellement pré-ordonnée $\pi(\mathcal{A}_\rhd)$. En effet, il suffit de considérer uniquement les bases compatibles (et leurs réparations associées) qui définissent un ordre différent entre les assertions. Cela est capturé par le lemme suivant.

Lemme 1 *Soit $\mathcal{WA}_1$ une ABox pondérée. Soit $S = \{\alpha \mid (f, \alpha) \in \mathcal{WA}_1\}$ l'ensemble de poids attachés aux assertions de $\mathcal{WA}_1$.*
Soit une fonction d'affectation $\omega : S \longrightarrow]0, 1]$ telle que $\forall \alpha_1, \alpha_2 \in S, \alpha_1 \geq \alpha_2$ ssi $\omega(\alpha_1) \geq \omega(\alpha_2)$.
Soit $\mathcal{WA}_2 = \{(f, \omega(\alpha)) \mid (f, \alpha) \in \mathcal{WA}_1\}$ une ABox pondérée obtenue en appliquant la fonction d'affectation ω aux poids attachés aux assertions de $\mathcal{WA}_1$. Alors :

$$\pi(\mathcal{WA}_1) = \pi(\mathcal{WA}_2).$$

Dans le Lemme 1, même si la ABox $\mathcal{WA}_2$ est différente de la ABox $\mathcal{WA}_1$, elle en préserve l'ordre sur les assertions. Dans ce cas, les deux bases pondérées génèrent les mêmes réparations.

Preuve : Montrons que $Inc(\mathcal{WA}_1) = \beta$ ssi $Inc(\mathcal{WA}_2) = \omega(\beta)$. Notons d'abord que si $\mathcal{C}_{12} = \{(f_1, \alpha_1), (f_2, \alpha_2)\}$ et $\mathcal{C}_{34} = \{(f_3, \alpha_3), (f_4, \alpha_4)\}$ sont deux conflits de $\mathcal{WA}_1$, alors $\mathcal{C}'_{12} = \{(f_1, \omega(\alpha_1)), (f_2, \omega(\alpha_2))\}$ et $\mathcal{C}'_{34} = \{(f_3, \omega(\alpha_3)), (f_4, \omega(\alpha_4))\}$ sont aussi deux conflits de $\mathcal{WA}_2$. Donc, par définition de la fonction $\omega(\cdot)$, si nous avons $min\{\alpha : (f, \alpha) \in \mathcal{C}_{12}\} = \alpha_1$ (resp. α_2), alors nous avons aussi $min\{\omega(\alpha) : (f, \omega(\alpha)) \in \mathcal{C}'_{12}\} = \omega(\alpha_1)$ (resp. $\omega(\alpha_2)$). De la même façon, si $min\{\alpha : (f, \alpha) \in \mathcal{C}_{12}\} > min\{\alpha : (f, \alpha) \in \mathcal{C}_{34}\}$, alors $min\{\omega(\alpha) : (f, \omega(\alpha)) \in \mathcal{C}'_{12}\} > min\{\omega(\alpha) : (f, \omega(\alpha)) \in \mathcal{C}'_{34}\}$. Donc si $Inc(\mathcal{WA}_1) = \beta$, alors $Inc(\mathcal{WA}_2) = \omega(\beta)$.
Supposons que $Inc(\mathcal{WA}_1) = \beta$. Soit $(f, \alpha) \in \mathcal{WA}_1$ t.q. $\alpha > \beta$. Alors $f \in \pi(\mathcal{WA}_1)$. Par définition de $\omega(\cdot)$, nous obtenons $\omega(\alpha) > \omega(\beta) = Inc(\mathcal{WA}_2)$. Cela signifie que $f \in \pi(\mathcal{WA}_2)$. De la même façon, soit $(f, \alpha) \in \mathcal{WA}_1$ t.q. $\alpha \leq \beta$. Alors $f \notin \pi(\mathcal{WA}_1)$. Par définition de $\omega(\cdot)$, nous obtenons $\omega(\alpha) \leq \omega(\beta) = Inc(\mathcal{WA}_2)$. Cela signifie que $f \notin \pi(\mathcal{WA}_2)$.
Nous en concluons que $\pi(\mathcal{WA}_1) = \pi(\mathcal{WA}_2)$. ■

Nous illustrons ces notions sur notre exemple.

Exemple 6 *Grâce au Lemme 1, afin de calculer la réparation $\pi(\mathcal{A}_\rhd)$, il suffit de considérer uniquement les trois bases $\mathcal{WA}_1$, $\mathcal{WA}_2$ et $\mathcal{WA}_3$ comme bases compatibles de $\mathcal{A}_\rhd$. Leurs π-réparations associées sont données par :*

— $\pi(\mathcal{WA}_1) = \{A(a), A(b), B(c), C(a), D(b), E(a)\}$.

— $\pi(\mathcal{WA}_2) = \{A(a), A(b), B(c), C(b)\}$.

— $\pi(\mathcal{WA}_3) = \{A(a), A(b), B(c)\}$.

La réparation partiellement pré-ordonnée est :
$\pi(\mathcal{A}_\rhd) = \bigcap_{i=1...3} \pi(\mathcal{WA}_i) = \{A(a), A(b), B(c)\}$. □

La prochaine section montre comment calculer la réparation $\pi(\mathcal{A}_\rhd)$ sans énumérer toutes les bases compatibles.

4.3 Caractérisation de la réparation partiellement pré-ordonnée

Afin d'éviter de calculer toutes les bases compatibles avec une ABox partiellement pré-ordonnée $\mathcal{A}_\rhd$, nous proposons une caractérisation de la Définition 5 en introduisant la notion d'assertion π-acceptée. En substance, une assertion est π-acceptée si elle est strictement préférée à au moins une assertion de chaque conflit de $\mathcal{A}_\rhd$.

Définition 6 *Soit $\mathbb{L} = (U, \rhd)$ une échelle d'incertitude. Soit $\mathcal{K}_\rhd = \langle \mathcal{T}, \mathcal{A}_\rhd, \mathbb{L} \rangle$ une KB DL-Lite partiellement pré-ordonnée. Soit $\mathcal{C}(\mathcal{A}_\rhd)$ l'ensemble des conflits de $\mathcal{A}_\rhd$.*
Une assertion $(f, u) \in \mathcal{A}_\rhd$ est π-acceptée ssi :

$$\forall \mathcal{C} \in \mathcal{C}(\mathcal{A}_\rhd), \exists (g, u_j) \in \mathcal{C}, g \neq f, \text{ t.q. } f \rhd g \text{ (i.e., } u \rhd u_j).$$

L'ensemble de conflits assertionnels $\mathcal{C}(\mathcal{A}_\rhd)$ est obtenu par la Définition 1 où la KB pondérée $\mathcal{WK}$ et la ABox pondérée $\mathcal{WA}$ sont remplacées par la KB partiellement pré-ordonnée $\mathcal{K}_\rhd$ et la ABox partiellement pré-ordonnée $\mathcal{A}_\rhd$.

Exemple 7 *L'ensemble de conflits assertionnels de $\mathcal{A}_\rhd$ est :*

$$\mathcal{C}(\mathcal{A}_\rhd) = \left\{ \begin{array}{l} \{(A(a), u_4), (B(a), u_1)\}, \\ \{(C(a), u_3), (D(a), u_1)\}, \\ \{(D(b), u_3), (C(b), u_2)\}, \\ \{(C(a), u_3), (B(a), u_1)\} \end{array} \right\}$$

Il est aisé de vérifier que les assertions $(A(a), u_4)$, $(A(b), u_4)$ et $(B(c), u_4)$ sont strictement préférées à au moins une assertion de chaque conflit, puisque u_4 est strictement préféré à tous les autres poids. Ainsi, ces assertions sont toutes π-acceptées. □

Un résultat important de cet article est que l'ensemble des assertions π-acceptées correspond exactement à la réparation d'une ABox partiellement pré-ordonnée $\mathcal{A}_\rhd$ (où les poids sont omis).

Proposition 1 *Une assertion $(f, u) \in \mathcal{A}_\rhd$ est π-acceptée ssi $f \in \pi(\mathcal{A}_\rhd)$.*

Preuve :

(i) Supposons que $(f, u) \in \mathcal{A}_\rhd$ est π-acceptée mais $f \notin \pi(\mathcal{A}_\rhd)$. Donc il existe une base compatible $\mathcal{WA}$ de $\mathcal{A}_\rhd$ et un poids $\alpha_i \in]0, 1]$ t.q. $(f, \alpha_i) \in \mathcal{WA}$ et $f \notin \pi(\mathcal{WA})$.
Soit $Inc(\mathcal{WA}) = \beta$. Par la Définition 2, cela signifie que $\mathcal{A}^{\geq\beta}$ est inconsistante mais $\mathcal{A}^{>\beta}$ est consistante.
Soit un conflit $\{(g, \alpha_j), (h, \alpha_k)\} \in \mathcal{C}(\mathcal{WA})$ où les assertions $g, h \in \mathcal{A}^{\geq\beta}$ (un tel conflit existe puisque $\mathcal{A}^{\geq\beta}$ est inconsistante). Donc, nécessairement $\alpha_j \geq \beta$ et $\alpha_k \geq \beta$ (puisque $\mathcal{A}^{\geq\beta}$ est inconsistante).
Par la Définition 3, $f \notin \pi(\mathcal{WA})$ signifie que $\alpha_i \leq \beta$. Donc $\alpha_j \geq \alpha_i$ et $\alpha_k \geq \alpha_i$. Mais cela contredit le fait que (f, u) est π-acceptée, ce qui garantit que $f \rhd g$ ou $f \rhd h$, i.e., $\alpha_i > \alpha_j$ ou $\alpha_i > \alpha_k$.

(ii) Supposons que l'assertion (f, u) n'est pas π-acceptée mais $f \in \pi(\mathcal{A}_\rhd)$. L'assertion (f, u) n'est pas π-acceptée signifie qu'il y a un conflit $\{(g, u_j), (h, u_k)\} \in \mathcal{C}(\mathcal{A}_\rhd)$ tel que $f \not\rhd g$ et $f \not\rhd h$, i.e., $u \not\rhd u_j$ et $u \not\rhd u_k$. Trois cas sont à considérer :

(a) $g \rhd f$ et $h \rhd f$ ont lieu, i.e., $u_j \rhd u$ et $u_k \rhd u$. Donc, dans toutes les bases compatibles de $\mathcal{A}_\rhd$, les assertions g et h sont préférées à f. Soit $\mathcal{WA}$ une base compatible qui contient $(f, \alpha_i), (g, \alpha_j)$ et (h, α_k), avec $\alpha_i, \alpha_j, \alpha_k \in]0, 1]$. Donc $\alpha_j > \alpha_i$ et $\alpha_k > \alpha_i$.
Les assertions g et h sont en conflit signifie que $Inc(\mathcal{WA}) \geq min(\alpha_j, \alpha_k)$. Ainsi, $Inc(\mathcal{WA}) \geq \alpha_i$, donc $f \notin \pi(\mathcal{WA})$. Mais cela contredit le fait que $f \in \pi(\mathcal{A}_\rhd)$.

(b) $f \bowtie g$ et $h \rhd f$ ont lieu, i.e., $u \bowtie u_j$ et $u_k \rhd u$. Dans ce cas, il suffit d'avoir une base compatible $\mathcal{WA}$ qui contient $(f, \alpha_i), (g, \alpha_j)$ et (h, α_k), avec $\alpha_i, \alpha_j, \alpha_k \in]0, 1]$, $\alpha_j > \alpha_i$ et $\alpha_k > \alpha_i$. Une telle base compatible existe toujours. Donc, $f \notin \pi(\mathcal{WA})$. Mais cela contredit le fait que $f \in \pi(\mathcal{A}_\rhd)$.
Le cas où $g \rhd f$ mais $f \bowtie h$ est également valide par symmétrie.

(c) $f \bowtie g$ et $f \bowtie h$ ont lieu, i.e., $u \bowtie u_j$ et $u \bowtie u_k$. Il est alors suffisant d'avoir une base compatible $\mathcal{WA}$ qui contient $(f, \alpha_i), (g, \alpha_j)$ et (h, α_k) où $\alpha_j > \alpha_i$ et $\alpha_k > \alpha_i$. Cela revient au cas (a) ci-dessus.

■

Exemple 8 *Les Exemples 6 et 7 montrent que les assertions π-acceptées (sans poids) sont exactement celles de $\pi(\mathcal{A}_\rhd)$, à savoir : $\{A(a), A(b), B(c)\}$.*

□

4.4 Propriétés de la réparation partiellement pré-ordonnée

La caractérisation donnée dans la Définition 6 nous permet d'établir les résultats suivants.

Proposition 2

1. *La base $\pi(\mathcal{A}_\rhd)$ est consistante avec la TBox.*
2. *Le calcul de $\pi(\mathcal{A}_\rhd)$ se fait en temps polynomial en taille de la ABox.*

Preuve :

1. La consistance de $\pi(\mathcal{A}_\rhd)$ est directe. Puisque la π-réparation $\pi(\mathcal{WA})$ de chaque base compatible $\mathcal{WA}$ de $\mathcal{A}_\rhd$ est consistante, l'intersection de toutes les π-réparations est nécessairement consistante.
2. Pour ce qui est de la complexité, nous rappelons que le calcul de l'ensemble de conflits $\mathcal{C}(\mathcal{A}_\rhd)$ se fait en temps polynomial en taille de $\mathcal{A}_\rhd$ en DL-Lite. Donc, calculer $\pi(\mathcal{A}_\rhd)$ se fait aussi en temps polynomial. En effet, vérifier si une assertion $(f, u) \in \mathcal{A}_\rhd$ est π-acceptée revient à parcourir tous les conflits dans $\mathcal{C}(\mathcal{A}_\rhd)$. Cela se fait en temps linéaire en taille de $\mathcal{C}(\mathcal{A}_\rhd)$ (la taille elle-même est bornée par $\mathcal{O}(|\mathcal{A}_\rhd|^2)$).

■

En plus de ces deux résultats, par construction de $\pi(\mathcal{A}_\rhd)$, il est aisé de constater que lorsque l'ordre partiel $\rhd$ est un ordre total noté $\succ$, $\pi(\mathcal{A}_\rhd)$ correspond à la π-réparation $\pi(\mathcal{A}_\succ)$.

Nous concluons que raisonner (i.e., répondre à des requêtes) à partir d'une KB inconsistante partiellement pré-ordonnée revient à remplacer la ABox originale $\mathcal{A}_\rhd$ par sa réparation $\pi(\mathcal{A}_\rhd)$. En effet, nous avons établi la consistance de la réparation avec la TBox mais aussi la traitabilité de son calcul. De plus, nous avons montré que lorsque la relation de préférence est un ordre total, notre méthode revient à calculer une réparation possibiliste standard.

5 Conclusion

Dans cet article, nous avons proposé une extension de DL-Lite possibiliste au cas de bases de connaissances partiellement pré-ordonnées afin de gérer l'inconsistance. L'idée principale consiste à interpréter une ABox partiellement pré-ordonnée comme une famille de ABox pondérées compatibles, puis de calculer la réparation possibiliste de chaque base compatible, et enfin de considérer l'intersection de toutes les réparations possibilistes. Cela produit une réparation unique pour la ABox partiellement pré-ordonnée. Nous avons proposé une caractérisation en introduisant la notion d'assertions π-acceptées et avons montré que la réparation partiellement pré-ordonnée revient à calculer l'ensemble des assertions π-acceptées. En particulier, nous avons montré que ce calcul peut se faire en temps polynomial en DL-Lite.

Dans des travaux futurs, nous comptons explorer des méthodes pour augmenter la productivité de la réparation partielle, par exemple en considérant la fermeture des réparations possibilistes associées aux ABox compatibles. Un aspect important consiste à déterminer si le calcul de la réparation fermée possibiliste partielle peut être fait en temps polynomial en DL-Lite. Nous espérons montrer que c'est en effet le cas en réduisant le problème à celui de répondre à une requête de *instance checking*. Plus généralement, nous comptons investiguer si des méthodes de calcul de réparations qui sont polynomiales dans les cas non-ordonné et prioritaire seraient également polynomiales en présence d'un ordre partiel. Une autre piste à étudier consiste à munir les axiomes de la TBox de degrés de possibilités.

Dans le cadre de notre projet de recherche appelé AniAge, nous comptons appliquer nos résultats au problème de *query answering* à partir d'ontologies de danses de l'Asie du Sud-est. Des experts en danses traditionnelles annotent sémantiquement des vidéos de danse, selon l'ontologie (la TBox) pour représenter les aspects culturels qui sont exprimés par certains mouvements de danse, des postures, des tenues et des accessoires. Les experts peuvent affecter des degrés de confiance à leurs annotations pour refléter différents degrés de fiabilité de l'information. Cela revient à définir une relation de priorité, à savoir un préordre total, sur les assertions. Cependant, différents experts peuvent ne pas partager la même signification des échelles de confiance. Cela peut s'exprimer en appliquant un préordre partiel à la ABox. Des conflits peuvent émerger lorsqu'une même vidéo est annotée différemment par plusieurs experts. Cela souligne l'importance de gérer l'inconsistance efficacement afin de pouvoir répondre à des requêtes.

Remerciements

Ce travail a été financé par le projet européen H2020-MSCA-RISE : AniAge (High Dimensional Heterogeneous Data based Animation Techniques for Southeast Asian Intangible Cultural Heritage). Ce travail a également bénéficié du soutien du projet AAP A2U QUID (QUeryIng heterogeneous Data).

Références

[1] Franz Baader, Andreas Ecke, Gabriele Kern-Isberner, and Marco Wilhelm. The complexity of the consistency problem in the probabilistic description logic $\mathcal{ALC}^{\mathsf{me}}$. In *12th International Symposium on Frontiers of Combining Systems (FroCoS), London, UK*, pages 167–184, 2019.

[2] Jean-François Baget, Salem Benferhat, Zied Bouraoui, Madalina Croitoru, Marie-Laure Mugnier, Odile Papini, Swan Rocher, and Karim Tabia. A general modifier-based framework for inconsistency-tolerant query answering. In *Principles of Knowledge Representation and Reasoning (KR), Cape Town, South Africa*, pages 513–516, 2016.

[3] Sihem Belabbes, Salem Benferhat, and Jan Chomicki. Elect : An inconsistency handling approach for partially preordered lightweight ontologies. In *Logic Programming and Nonmonotonic Reasoning (LPNMR), Philadelphia, USA*, pages 210–223, 2019.

[4] S. Benferhat, Z. Bouraoui, and K. Tabia. How to select one preferred assertional-based repair from inconsistent and prioritized DL-Lite knowledge bases ? In *International Joint Conference on Artificial Intelligence (IJCAI), Buenos Aires, Argentina*, pages 1450–1456, 2015.

[5] S. Benferhat, D. Dubois, and H. Prade. How to infer from inconsistent beliefs without revising ? In *Inter-*

national Joint Conference on Artificial Intelligence, pages 1449–1457. Morgan Kaufmann, 1995.

[6] Salem Benferhat and Zied Bouraoui. Min-based possibilistic DL-Lite. *Journal of Logic and Computation*, 27(1) :261–297, 2017.

[7] Salem Benferhat, Sylvain Lagrue, and Odile Papini. Reasoning with partially ordered information in a possibilistic logic framework. *Fuzzy Sets and Systems*, 144(1) :25–41, 2004.

[8] Meghyn Bienvenu and Camille Bourgaux. Inconsistency-tolerant querying of description logic knowledge bases. In *Reasoning Web : Logical Foundation of Knowledge Graph Construction and Query Answering*, volume 9885, pages 156–202. LNCS. Springer, 2016.

[9] Fernando Bobillo and Umberto Straccia. Reasoning within fuzzy OWL 2 EL revisited. *Fuzzy Sets and Systems*, 351 :1–40, 2018.

[10] Stefan Borgwardt, İsmail İlkan Ceylan, and Thomas Lukasiewicz. Recent advances in querying probabilistic knowledge bases. In *27th International Joint Conference on Artificial Intelligence, (IJCAI), Stockholm, Sweden*, pages 5420–5426, 2018.

[11] Stefan Borgwardt and Rafael Peñaloza. Fuzzy description logics - a survey. In *Scalable Uncertainty Management (SUM)*, pages 31–45, 2017.

[12] D. Calvanese, E. Kharlamov, W. Nutt, and D. Zheleznyakov. Evolution of DL-Lite knowledge bases. In *International Semantic Web Conference (1)*, pages 112–128, 2010.

[13] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics : The DL-Lite family. *Journal of Automated Reasoning*, 39(3) :385–429, 2007.

[14] Arthur P. Dempster. Upper and lower probabilities induced by a multivalued mapping. *The Annals of Mathematical Statistics*, 38 :325–339, 1967.

[15] Didier Dubois, Jérôme Mengin, and Henri Prade. Possibilistic uncertainty and fuzzy features in description logic. a preliminary discussion. *Fuzzy Logic and the Semantic Web. Volume 1 of Capturing Intelligence*, pages 101–113, 2006.

[16] Didier Dubois and Henri Prade. Possibility theory and its applications : Where do we stand ? In *Springer Handbook of Computational Intelligence*, pages 31–60. 2015.

[17] Didier Dubois, Henri Prade, and Steven Schockaert. Generalized possibilistic logic : Foundations and applications to qualitative reasoning about uncertainty. *Artificial Intelligence*, 252 :139–174, 2017.

[18] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning About Knowledge*. MIT Press, 2003.

[19] Marcelo Finger, Lluis Godo, Henri Prade, and Guilin Qi. Advances in weighted logics for artificial intelligence. *International Journal of Approximate Reasoning*, 88 :385–386, 2017.

[20] Bernhard Hollunder. An alternative proof method for possibilistic logic and its application to terminological logics. *International Journal of Approximate Reasoning*, 12(2) :85–109, 1995.

[21] Roman Kontchakov, Carsten Lutz, David Toman, Frank Wolter, and Michael Zakharyaschev. The combined approach to query answering in DL-Lite. In *12th International Conference on Principles of Knowledge Representation and Reasoning (KR), Toronto, Canada*, pages 247–257, 2010.

[22] D. Lembo, M. Lenzerini, R. Rosati, M. Ruzzi, and D. Fabio Savo. Inconsistency-tolerant semantics for description logics. In *International Conference on Web Reasoning and Rule Systems*, volume 6333 of *LNCS*, pages 103–117, 2010.

[23] Carsten Lutz and Lutz Schröder. Probabilistic description logics for subjective uncertainty. In *12th International Conference on Principles of Knowledge Representation and Reasoning (KR), Toronto, Canada*, 2010.

[24] Jeff Z. Pan, Giorgos B. Stamou, Giorgos Stoilos, and Edward Thomas. Expressive querying over fuzzy DL-Lite ontologies. In *20th DL workshop, Bressanone, Italy*, 2007.

[25] G. Qi, Q. Ji, Jeff Z. Pan, and J. Du. Extending description logics with uncertainty reasoning in possibilistic logic. *International Journal of Intelligent Systems*, 26(4) :353–381, 2011.

[26] Glenn Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[27] Wolfgang Spohn. *The Laws of Belief - Ranking Theory and Its Philosophical Applications*. Oxford University Press, 2014.

[28] Umberto Straccia. Towards top-k query answering in description logics : The case of DL-Lite. In *10th European Conference on Logics in Artificial Intelligence (JELIA), Liverpool, UK*, pages 439–451, 2006.

[29] Umberto Straccia. *Foundations of Fuzzy Logic and Semantic Web Languages*. Chapman & Hall/CRC, 2013.

[30] Fayçal Touazi, Claudette Cayrol, and Didier Dubois. Possibilistic reasoning with partially ordered beliefs. *Journal of Applied Logic*, 13(4) :770–798, 2015.

[31] Despoina Trivela, Giorgos Stoilos, and Vasilis Vassalos. Query rewriting for DL ontologies under the ICAR semantics. In *Rules and Reasoning - Third International Joint Conference, RuleML+RR, Bolzano, Italy*, pages 144–158, 2019.

[32] Lofti A. Zadeh. Fuzzy sets as a basis for a theory of probability. *Fuzzy Sets and Systems*, 1 :3–28, 1978.

Classification des Séries Temporelles Incertaines Par Transformation Shapelet

Michael Franklin MBOUOPDA[1], Engelbert MEPHU NGUIFO[1]

[1] University Clermont Auvergne, CNRS, ENSMSE, LIMOS, F-63000 CLERMONT-FERRAND, FRANCE

{michael.mbouopda, engelbert.mephu_nguifo}@uca.fr

Abstract

Time series classification is a task that aims at classifying chronological data. It is used in a diverse range of domains such as meteorology, medicine and physics. In the last decade, many algorithms have been built to perform this task with very appreciable accuracy. However, the uncertainty in data is not explicitly taken into account by these methods. Using uncertainty propagation techniques, we propose a new uncertain dissimilarity measure based on euclidean distance. We also show how to classify uncertain time series using the proposed dissimilarity measure and shapelet transform, one of the best time series classification methods. An experimental assessment of our contribution is done on the well known UCR dataset repository.

Keywords

Time series, Classification, Uncertainty, Shapelet.

Résumé

La classification des séries temporelles est une tâche qui consiste à classifier les données chronologiques. Elle est utilisée dans divers domaines tels que la météorologie, la médecine et la physique. Plusieurs techniques performantes ont été proposées durant les dix dernières années pour accomplir cette tâche. Cependant, elles ne prennent pas explicitement en compte l'incertitude dans les données. En utilisant la propagation de l'incertitude, nous proposons une nouvelle mesure de dissimilarité incertaine basée sur la distance euclidienne. Nous montrons également comment faire la classification de séries temporelles incertaines en couplant cette mesure avec la méthode de transformation shapelet, l'une des méthodes les plus performantes pour cette tâche. Une évaluation expérimentale de notre contribution est faite sur le dépôt de données temporelles UCR.

Mots-clés

Série temporelle, Classification, Incertitude, Shapelet.

1 Introduction

La dernière décennie fut caractérisée par la disponibilité de données dans un large domaine d'application tels que la météorologie, l'astronomie et le suivi d'objets. Généralement, ces données sont représentées sous forme de séries temporelles [5], c'est-à-dire des données séquentielles ordonnées suivant le temps. Par ailleurs, plusieurs méthodes de classification automatique de ces types de données ont été développées durant la même décennie [3, 8, 11]. Pour autant que nous sachions cependant, toutes ces méthodes supposent que les données sont précises et fiables. Ainsi elles ne prennent pas en compte l'incertitude présente dans les mesures. Pourtant, toute mesure est sujette à l'incertitude qui pourrait provenir de l'environnement, de la précision de l'outil de mesure, des contraintes de confidentialité et bien d'autres facteurs. De plus, même si l'incertitude peut être réduite, elle ne peut être supprimée [18]. Dans certaines applications, l'incertitude ne peut être négligée et doit être traité avec rigueur [17]. Par exemple, [4] a montré l'importance de la prise en compte de l'incertitude dans l'interprétation des images médicales. De même, la nécessité de bien gérer l'incertitude dans les données spatiales a été mise en évidence par [7].

Les méthodes basées sur les shapelets sont parmi les meilleures approches qui ont été développées pour la classification des séries temporelles [3]. Elles sont spécialement appréciées pour leur caractère interprétable, leur robustesse et leur vitesse d'inférence [19]. Ces approches procèdent en trois étapes :

1. étape d'extraction des shapelets, qui peut être vue comme une extraction de caractéristiques,
2. étape de transformation, qui consiste à calculer les vecteurs caractéristiques de chaque séries temporelle dans le jeu de données,
3. étape d'apprentissage, qui consiste à entraîner un modèle de classification supervisée sur le jeu de données obtenu après transformation.

Dans ce papier, nous montrons comment cette approche peut être adaptée et appliquée dans le contexte de la classification des séries temporelles incertaines. Pour y parvenir, nous proposons premièrement une mesure de dissimilarité incertaine inspirée de la distance euclidienne. Ensuite, nous intégrons cette mesure dans l'approche de transformation shapelet afin de faire la classification des séries temporelles incertaines.

La suite de cet article est organisée comme suit : les travaux connexes sont présentés dans la section 2. Dans la section 3, nous présentons une nouvelle mesure de dissi-

milarité incertaine et dans la section 4, nous construisons le modèle de classification des séries temporelles incertaines. La section 5 présente les expérimentations que nous avons effectuées et les résultats obtenus. Finalement, la section 6 conclut ce papier.

2 Travaux connexes

L'analyse des séries temporelles incertaines est un problème bien connu et il existe dans la littérature des travaux sur le sujet. Ces travaux ont conduit au développement de mesures de similarité dites probabilistes. Étant données deux séries temporelles incertaines, une mesure de similarité incertaine calcule la probabilité que la distance entre ces deux séries soit inférieure à un seuil défini par l'utilisateur [5]. Ces mesures probabilistes ont été couplées avec le modèle de classification supervisée 1-PPV (pour 1-Plus Proche Voisin) afin d'effectuer la classification des séries temporelles incertaines. Par exemple, [17] a proposé la mesure de similarité probabiliste DUST et l'a couplée avec un 1-PPV. Les mesures de similarité probabilistes ne sont pas toujours applicables en pratique. En effet, elles sont basées sur des suppositions qui ne sont pas toujours satisfaites. C'est le cas de la mesure de similarité probabiliste PROUD [20] qui requiert que l'incertitude soit la même à chaque observation de la série temporelle. MUNICH [1], une autre mesure de similarité incertaine représente l'incertitude comme étant la non unicité des observations à chaque instant de la série. Ainsi à chaque instant, on n'a pas une unique observation, mais plutôt un ensemble d'observations possibles de cet instant. DUST a été proposé comme solution aux limitations de MUNICH et PROUD, mais fait aussi une supposition qui n'est pas toujours observable en pratique : DUST requiert que l'incertitude des observations du même instant dans toutes les séries temporelles du jeu de données soit la même. Plus récemment, la mesure de dissimilarité FOTS [9] a été proposée ; elle est robuste à l'incertitude, mais ladite incertitude n'est pas explicitement prise en compte dans le calcul de FOTS. Toutes ces mesures de similarité/dissimilarité pour les données incertaines partagent ensemble une même caractéristique : celle de représenter la similarité/dissimilarité entre deux données incertaines par une valeur certaine. Il est impossible de comparer des données contenant de l'incertitude et espérer que le résultat de la comparaison soit sans incertitude. Pour toutes ces raisons, nous proposons à la section 3, UED, une mesure de dissimilarité qui ne fait aucune supposition sur la distribution de l'incertitude et qui donne le résultat de la comparaison accompagné d'un intervalle de confiance.

Introduite par [19] en tant que arbre de décision shapelet, la classification des séries temporelles par shapelet a été généralisée par [10] sous l'appellation transformation shapelet. La généralisation permet d'utiliser les shapelets avec tout modèle de classification supervisée, et plus le modèle de classification utilisé est performant plus la classification des séries est meilleure[10]. Pour autant que nous sachions, le meilleur modèle de classification des séries temporelles tel que reporté dans la littérature est HIVE-COTE [13]. Il s'agit d'un méta-modèle composé de plusieurs modules dont l'un est basé sur les shapelets. Pour avoir un modèle comme HIVE-COTE pour les séries temporelles incertaines, il faut prendre en compte l'incertitude dans chacun des modules. Dans ce papier, nous présentons à la section 4 un moyen de prendre en compte l'incertitude dans le module basé sur les shapelets.

3 UED : Mesure de dissimilarité incertaine

L'incertitude est différente de l'erreur car l'erreur peut être évitée en faisant plus attention alors qu'on ne peut pas échapper à l'incertitude [18]. Cependant il existe des moyens pour réduire l'incertitude. Peu importe la méthode de mesure, il y a toujours une incertitude et les mesures incertaines ne peuvent pas être comparées avec 100% de confiance. Il existe plusieurs représentation de l'incertitude et dans ce papier, une mesure incertaine x est représentée par deux composantes qui sont l'estimation optimiste $\hat{x}$ et l'incertitude δx qui est l'écart maximal possible par rapport à l'estimation optimiste.

$$x = \hat{x} \pm \delta x, \delta x \geq 0 \tag{1}$$

Cette formule signifie que x est un élément de l'interval $[\hat{x} - \delta x, \hat{x} + \delta x]$ et qu'il est fort probable que sa valeur soit $\hat{x}$.

La distance euclidienne (ED) est très utilisée dans la littérature pour mesurer la dissimilarité entre deux séries temporelles. Elle est particulièrement utilisée dans les méthodes à base de shapelet [3, 10, 19]. Étant données deux séries temporelles $T_1 = \{t_{11}, t_{12}, ..., t_{1n}\}$ et $T_2 = \{t_{21}, t_{22}, ..., t_{2n}\}$, la distance euclidienne entre elles est définie comme suit :

$$ED(T_1, T_2) = \sum_{i=1}^{n} (t_{1i} - t_{2i})^2 \tag{2}$$

La racine carrée étant une fonction positive et croissante, il n'est pas obligatoire de l'appliquer dans le calcul de la dissimilarité. En effet, si on ordonne des séries temporelles selon leur similarité les unes par rapport aux autres, l'application ou pas de la racine carrée n'influence pas l'ordre. Lorsque chaque composante t_{ij} est une mesure incertaine, T_1 et T_2 sont appelées des séries temporelles incertaines et la similarité entre elles ne pas être calculée avec 100% de fiabilité. Nous utilisons les techniques de propagation de l'incertitude [18] afin de mesurer l'incertitude qu'il y a dans la dissimilarité entre les deux séries temporelles.

Soient $x = \hat{x} \pm \delta x$ et $y = \hat{y} \pm \delta y$ deux mesures incertaines, nous avons les propriétés suivantes :

- $z = x + y = \hat{z} \pm \delta z$, où $\hat{z} = \hat{x} + \hat{y}$ et $\delta z = \delta x + \delta y$
- $z = x - y = \hat{z} \pm \delta z$, où $\hat{z} = \hat{x} - \hat{y}$ et $\delta z = \delta x + \delta y$
- $z = x^n = \hat{z} \pm \delta z$, où $\hat{z} = (\hat{x})^n$ et $\delta z = |n\frac{\delta x}{x}(\hat{x})^n|$

En utilisant ces propriétés de propagation de l'incertitude, nous pouvons calculer l'incertitude qu'il y a sur la distance euclidienne entre deux séries temporelles incertaines T_1 et T_2 en propageant l'incertitude. Nous obtenons une mesure

de dissimilarité incertaine que nous appellons UED et elle est définie comme suit :

$$\begin{aligned}\mathrm{UED}(T_1, T_2) &= \sum_{i=1}^{n} (\hat{t_{1i}} - \hat{t_{2i}})^2 \pm 2 \sum_{i=1}^{n} |\hat{t_{1i}} - \hat{t_{2i}}| \times (\delta t_{1i} + \delta t_{2i}) \\ &= ED(\hat{T_1}, \hat{T_2}) \pm 2 \sum_{i=1}^{n} |\hat{t_{1i}} - \hat{t_{2i}}| \times (\delta t_{1i} + \delta t_{2i})\end{aligned} \tag{3}$$

où $\hat{T_i}$ en obtenue en ignorant l'incertitude dans T_i, c'est à dire en supposant que toutes les incertitudes sont à 0.
La sortie de UED correspond à la dissimilarité incertaine entre deux séries temporelles incertaines qu'on place en entrée. Afin d'utiliser UED pour la classification des séries temporelles incertaines, plus précisement avec les méthodes à shapelet, il est nécessaire de définir une relation d'ordre pour les mesures incertaines. Nous proposons deux façons de comparer deux mesures incertaines : la première est la plus simple et est basée sur la confiance et la seconde est un ordre stochastique.

Ordre simple des mesures incertaines

Cet ordonnancement est basé sur deux propiétés basiques. Soient x et y deux mesures incertaines, la première propriété est celle de l'égalité et stipule que les mesures sont équivalentes si et seulement si leurs estimations optimistes et leurs incertitudes sont égales :

$$x = y \iff \hat{x} = \hat{y} \wedge \delta x = \delta y \tag{4}$$

La propriété d'infériorité est la seconde et stipule que x est inférieur y si et seulement si l'estimation optimiste de x est inférieure à celle de y. Lorsque leurs estimations optimistes sont égales, la plus petite est celle qui la plus petite incertitude.

$$x < y \iff (\hat{x} < \hat{y}) \vee ((\hat{x} = \hat{y}) \wedge (\delta x < \delta y)) \tag{5}$$

Contrairement à la propriété d'égalité qui est intuitive, la propriété d'infériorité est moins simple. Malheureusement, nous n'avons pas de justification mathématique de cette propriété. Cependant deux aspects ont guidé sa conception : premièrement nous faisons d'une certaine manière confiance à l'estimation optimiste donnée par l'expert du domaine. Deuxièmement, il est préferable de travailler avec les valeurs dont l'incertitude est réduite au maximum.
Il est à noter que ces deux propriétés ne permettent pas toujours d'avoir le bon ordre ; en effet, si $x = 2 \pm 0.5$ et $y = 2 \pm 0.1$ alors l'application de la propriété d'infériorité nous dit que $y < x$. Maintenant, s'il y avait un oracle capable de calculer la valeur exacte d'une mesure incertaine, il pourrait dire que $x = 1.8$ et $y = 2$. Et ainsi, l'ordre défini serait incorrect. Ce phénomène peut aussi s'observer avec la propriété d'égalité.

Ordre stochastique des mesures incertaines

Une mesure incertaine peut être considérée comme une variable aléatoire dont la moyenne est l'estimation optimiste et l'incertitude est l'écart-type. Partant de cette considération, l'ordre stochastique entre deux mesures incertaines peut être défini. Une variable aléatoire X est *stochastiquement plus petite ou égale* (noté $\leq_{st}$) à une autre variable aléatoire Y si et seulement si $P(X > t) \leq P(Y > t) \ \forall t \in \mathbb{R}$ [14]. Étant donnée que la valeur exacte d'une mesure incertaine x est dans l'intervalle $[\hat{x} - \delta x, \hat{x} + \delta x]$, le domaine de t peut être réduit à l'intervalle $\mathbb{I} = [\min(X, Y); \max(X, Y)]$; où $\min(X, Y)$ et $\max(X, Y)$ sont repectivement les valeurs minimale et maximale possibles de l'union des valeurs de X et Y. L'ordre stochastique peut être écrite et développée comme suit :

$$\begin{aligned}X \leq_{st} Y &\iff P(X > t) \leq P(Y > t) \ \forall t \in \mathbb{I} \\ &\iff 1 - P(X > t) > 1 - P(Y > t) \ \forall t \in \mathbb{I} \\ &\iff P(X \leq t) > P(Y \leq t) \ \forall t \in \mathbb{I} \\ &\iff CDF_X(t) > CDF_Y(t) \ \forall t \in \mathbb{I}\end{aligned} \tag{6}$$

$CDF_X(t)$ est la fonction de répartition de la variable aléatoire X évaluée à t. Puisque le nombre d'éléments de $\mathbb{I}$ est infini, nous discrétisons en divisant l'intervalle en k valeurs différentes :

$$\min(X, Y) + i \times \frac{\max(X, Y) - \min(X, Y)}{k} \tag{7}$$

$0 \leq i \leq k$ et k est un nombre entier à fixer.
Contrairement à l'ordre simple qui est total, l'ordre stochastique n'est qu'un ordre partiel. Ainsi, la relation *stochastiquement plus petite ou égale* n'est pas définie pour toute paire de mesures incertaines. L'ordre stochastique entre deux mesures incertaines n'est donc pas toujours défini. Il s'agit là clairement d'une limitation, cependant nous n'avons pas trouvé un ordre stochastique qui soit total dans la littérature.
Maintenant que nous pouvons ordonner des mesures incertaines, voyons comment UED est utilisée pour la classification de séries temporelles incertaines.

4 Classification shapelet incertaine

Dans cette partie, nous décrivons comment effectuer la classification de séries temporelles incertaines avec l'approche par shapelet. Nous nous basons sur l'algorithme de classification par transformation shapelets [10]. Tout d'abord, il est nécessaire de définir quelques concepts fondamentaux.
Une *série temporelle incertaine* T est une séquence de m (sa taille) valeurs incertaines ordonnées suivant une dimension temporelle.

$$T = \hat{T} \pm \delta T = \{t_1 \pm \delta t_1, t_2 \pm \delta t_2, ..., t_m \pm \delta t_m\} \tag{8}$$

Une *sous séquence incertaine* S d'une série temporelle incertaine T est une séquence de l (sa taille) valeurs consecutives dans T.

$$S = \hat{S} \pm \delta S = \{t_{i+1} \pm \delta t_{i+1}, ..., t_{i+l} \pm \delta t_{i+l}\} \tag{9}$$

, où $1 \leq i \leq m - l, \ 1 \leq l \leq m$ et m la longueur de T

La *dissimilarité entre deux sous séquences incertaines* S et R est donnée par UED

$$d = \mathrm{UED}(S, R) = \mathrm{UED}(R, S). \tag{10}$$

Et la dissimilarité entre une série temporelle incertaine T et une sous séquence quelconque S de longueur inférieure ou égale à la longueur de T est définie comme suit :

$$\mathrm{UED}(T, S) = \min\{\mathrm{UED}(S, R) \mid \forall R \subset T, |S| = |R|\} \tag{11}$$

Un *séparateur incertain* sp pour un jeu de données D de séries temporelles incertaines est une sous séquence incertaine qui divise D en deux sous ensembles D_1 et D_2 tels que :

$$\begin{aligned} D_1 &= \{T \mid \mathrm{UED}(T, sp) \leq \epsilon\,, \forall T \in D\} \\ D_2 &= \{T \mid \mathrm{UED}(T, sp) > \epsilon\,, \forall T \in D\} \end{aligned} \tag{12}$$

La qualité d'un séparateur est mesurée en utilisant le gain d'information (IG). Étant données les définitions précédentes, nous pouvons définir ce qu'est un *shapelet incertain*. Pour un jeu de données D, un shapelet incertain S est un séparateur incertain qui maximise le gain d'information

$$S = \underset{sp}{\mathrm{argmax}}(IG(D, sp)) \tag{13}$$

L'algorithme de tranformation shapelet est décrite en détail dans [10]. Nous donnons ici un résumé de cet algorithme tout en mettant l'accent sur les changements lorsqu'on est dans le contexte des séries temporelles incertaines.
Étant donné un jeu de données D de séries temporelles incertaines, la première étape consiste à sélectionner parmi toutes les sous séquences de D les k shapelets de gain d'information maximal. Cette étape est effectuée par l'algorithme 1 qui prend en entrée le jeu de données, le nombre de shapelets incertains à extraire, les longueurs minimale et maximale d'un shapelet incertain. La longueur minimale est au moins égale à 3 et la longueur maximale est celle de la plus longue série dans le jeu de données. Cependant, une connaissance du domaine d'application peut aider à mieux fixer ces paramètres. L'algorithme utilise trois sous procédures :

- $GenCand(T, MIN, MAX)$ qui génère toutes les sous séquences incertaines contenues dans la série temporelle T. Ici, seules les sous séquences de longueur comprise entre MIN et MAX sont générées.
- $EvaluerCand(cands, D)$ qui évalue la qualité de séparation de chacune des sous séquences générées. Pour chaque sous séquence dans $cands$, le gain d'information obtenu en l'utilisant comme séparateur pour D est calculé.
- $ExtraireTop(C, Q, k)$ prend la liste des shapelets candidats C, leur gain d'information ou qualité Q et retourne les k shapelets incertains dont les qualités sont les plus élevées.

En résumé, l'algorithme 1 génère toutes les sous séquences de longueur au moins égale à MIN et au plus égale à MAX à partir du jeu de données, mesure la qualité de chacune de ces sous séquences en tant que séparateur en utilisant le gain d'information et finalement renvoie les k sous séquences ayant les meilleures qualités.

Algorithme 1 : Sélection des top-k Shapelets Incertains

Données : D, k, MIN, MAX
Résultat : k meilleurs shapelets incertains
début
  $C \leftarrow \emptyset; Q \leftarrow \emptyset;$
  pour $i \leftarrow 1, n$ **faire**
    $cands \leftarrow GenCand(T_i, MIN, MAX);$
    $qualities \leftarrow EvaluerCand(cands, D);$
    $C \leftarrow C \cup cands;$
    $Q \leftarrow Q \cup qualities;$
  fin
  $S \leftarrow ExtraireTop(C, Q, k);$
  retourner S
fin

La prochaine étape après la sélection des k shapelets incertains est la transformation shapelet incertain. Il s'agit de la transformation shapelet telle que décrite par [10], mais à la différence que l'incertitude est propagée tout au long du processus de transformation. Cette étape est faite par l'algorithme 2 qui a pour entrée le jeu de données D, l'ensemble des k meilleurs shapelets S ainsi que sa taille k. Pour chaque série temporelle incertaine dans le jeu de donnée, son vecteur caractéristique de longeur k est calculé en utilisant UED. Le i-ème élément de ce vecteur est la distance euclidienne incertaine entre la série temporelle incertaine et le i-ème shapelet incertain dans S. L'ensemble des vecteurs caractéristiques forme le jeu de données transformé et est renvoyé par la procédure.

Algorithme 2 : Transformation Shapelet Incertain

Données : D, S, k
Résultat : Le jeu de données transformées
début
  pour $i \leftarrow 1, n$ **faire**
    $temp \leftarrow \emptyset;$
    pour $j \leftarrow 1, k$ **faire**
      $temp_j \leftarrow UED(T_i, S_j)$
    fin
    $D_i \leftarrow temp_j$
  fin
  retourner D
fin

La troisième et dernière étape est la classification proprement dite. Un algorithme de classification supervisée est entraîné sur le jeu de données transformé, de sorte qu'étant donné le vecteur caractéristique d'une nouvelle série temporelle, sa classe puisse être prédite. Vu que l'incertitude a été propagée, elle doit être prise en compte dans le processus d'apprentissage. L'algorithme d'apprentissage a donc comme entrées les meilleurs estimations (les caractéristiques) et l'incertitude sur ces estimations (méta-caractéristiques). Si au lieu de UED, les algorithmes 1 et 2 utilisent l'une des mesures de dissimilarité/similarité de la littérature (DUST, MUNICH, PROUD ou FOTS), le modèle de classification ne pourrait être avisé de l'incertitude dans les données car aucune de ces métriques ne donne l'incertitude qu'il y a sur son résultat, ce qui est problématique sachant que nos données ont de l'incertitude.

La figure 1 donne un aperçu global du processus de classification de séries temporelles incertaines. Pendant la phase d'apprentissage, les k meilleurs shapelets sont sélectionnés puis un modèle de classification supervisée (illustré ici par un arbre de décision pour sa simplicité) est entraîné sur le jeu de données transformées. À chaque noeud de l'arbre de décision une fois entraîné est associé un shapelet incertain et la branche suivie par une série temporelle dans l'arbre est fonction de la similarité (calculée avec UED) entre elle et les shapelets associés à chacun des noeuds de ladite branche. Durant la phase d'inférence, les shapelets incertains extraits lors de la phase d'apprentissage sont utilisés pour transformer les données de test, et l'arbre de décision précédemment entraîné est utilisé pour prédire les différentes classes. Appelons ce processus de classification par transformation shapelet incertain *UST*.

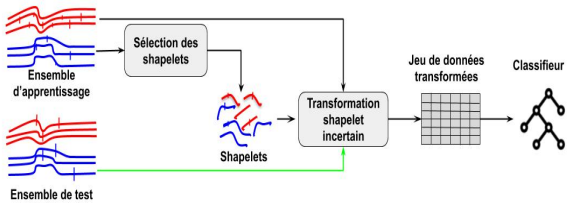


FIGURE 1 – Processus de classification des séries temporelles incertaines

5 Expérimentations

Dans cette section, nous évaluons expérimentalement notre approche de classification de séries temporelles incertaines et la comparons avec ce qui existe dans la littérature. Tout comme il est généralement le cas dans la littérature, le critère de comparaison pour les différents modèles est le taux d'exactitude (ou de généralisation) qui mesure le taux de bonnes prédictions sur l'ensemble de test [3, 5, 9, 10]. Étant donné que les modèles produisent en sortie la distribution de probabilité des instances en entrée par rapport aux différentes classes, nous utilisons la classe la plus probable comme la classe prédite, et c'est elle qui est prise en compte dans l'évaluation des modèles. Nous avons comparé les 4 modèles suivants :

— **UST_FLAT** : il s'agit de l'algorithme décrit dans la section 4. Les vecteurs caractéristiques sont représentés comme des vecteurs plats dont la première moitié contient les estimations optimistes, et la deuxième moitié contient les incertitudes sur les estimations. Ce modèle utilise l'ordre simple des mesures incertaines.

— **UST_FLAT_ST** : ce modèle est pareil que UST_FLAT, mais à la seule différence qu'il utilise l'ordre stochastique pour trier les mesures incertaines. Dans ce modèle, une mesure incertaine $x = \hat{x} \pm \delta x$ est considérée comme une variable aléatoire distribuée suivant une loi normale de moyenne $\hat{x}$ et d'écart-type δx. La fonction de répartition d'une telle variable aléatoire est :

$$CDF_X(t) = \frac{1}{2}(1 + erf(\frac{t - \hat{x}}{\delta x \sqrt{2}}))$$

$erf(\cdot)$ est la fonction d'erreur de Gauss et nous avons utilisé l'implémentation de Apache Commons Math [1]. Afin de discrétiser $\mathbb{I}$ (en utilisant Eq. 7), nous avons fixé la valeur de k à 100. Plus k est grand, mieux l'intervalle $\mathbb{I}$ est approché, mais une valeur de k trop grande ralentit le processus de classification. Nous avons essayé plusieurs valeurs de k, et les meilleurs résultats pour nos données sont obtenus avec $k = 100$. Nous avons aussi utilisé une version relaxée de l'ordre stochastique : étant données deux variables aléatoires X et Y, nous considérons $X \leq Y$ si le nombre de valeurs t dans $\mathbb{I}$ pour lesquelles $CDF_X(t) > CDF_Y(t)$ est plus grand que le nombre de valeurs t' dans $\mathbb{I}$ pour lesquelles $CDF_X(t') \leq CDF_Y(t')$

— **DUST_UNIFORM** : Il s'agit ici de l'algorithme UST dans lequel UED a été remplacée par la version uniforme de DUST. Chaque série temporelle est supposée suivre une distribution uniforme. Ainsi il n'y a pas de propagation de l'incertitude et le modèle de classification n'est pas au courant de l'incertitude dans les données. Pour comparer deux mesures incertaines, DUST requiert qu'elles aient la même incertitude. Nous avons utilisé l'incertitude de la valeur la plus incertaine comme incertitude dans le calcul de DUST. Ceci permet d'inclure les deux incertitudes, bien que cela rend encore plus incertaine la valeur qui a moins d'incertitude.

— **DUST_NORMAL** : tout comme DUST_UNIFORM, mais suppose que chaque série temporelle est distribuée suivant une loi normale.

Bien que des modèles d'apprentissage supervisé tels que les machines à vecteur de support, les forêts aléatoires et les réseaux de neurones peuvent être utilisés pour augmenter le taux d'exactitude (TE), nous avons choisi d'utiliser un arbre de décision (implémentation J48) comme modèle de classification dans chacun de ces quatre modèles. Nous souhaitons que le TE soit corrélé à la mesure de dissimilarité utilisée et non au modèle de classification. Ceci permet de mettre en évidence l'importance de la prise en compte de l'incertitude.

5.1 Datasets

Nous avons expérimenté sur 29 datasets provenant de UCR [6]. La première colonne du tableau 1 est la liste de nos jeux de données. Bien que UCR contient des séries temporelles univariées et multivariées, nous nous limitons aux datasets univariés. Cependant l'approche par transformation shapelet est aussi applicable sur des datasets multivariés, et aussi sur des datasets avec des séries temporelles de longueur

1. https://commons.apache.org/proper/commons-math/

variable. Chaque dataset du dépôt UCR est déjà divisé en ensemble de test et d'apprentissage.
Les datasets qui sont sur UCR ne contiennent par d'incertitude. Nous avons manuellement ajouté l'incertitude dans nos datasets. Pour chaque dataset, l'incertitude ajoutée suit une distribution normale de moyenne 0 et de déviation standard $c \times \sigma$, où σ est l'écart-type du jeu de données et c est un paramètre qui nous permet de contrôler la grandeur de l'incertitude ajoutée. Nous avons utilisé deux valeurs de c qui sont 0.1 et 0.2. L'ajout de l'incertitude dans le dataset CBF est illustré par la figure 2 pour une instance. La ligne orange est la série temporelle originale ne contenant pas d'incertitude. La ligne bleue est la série temporelle obtenue après ajout de l'incertitude. Pendant la phase d'apprentissage, la série temporelle originale n'est pas utilisée ; seule la série temporelle incertaine est utilisée. Chaque valeur dans un jeu de données a sa propre incertitude qui est l'écart-type de la loi normale qui a généré cette incertitude.

5.2 Code source

Nous avons utilisé un code existant comme base pour notre implémentation. Il s'agit d'un projet open source contenant l'implémentation d'une panoplie de méthodes de classification de séries temporelles parmi lesquelles la transformation shapelet. Ce projet est développé par [3] et est hébergé sur la plateforme Github [2]. Il est écrit en langage Java. Nous y avons ajouté notre implémentation de UED et UST et avons distribué le code via Github [3]. Le script Python utilisé pour ajouter l'incertitude est également disponible sur sur le même dépôt [4].

5.3 Résultats et discussion

Le taux d'exactitude (TE) de chacun de nos 4 modèles a été mesuré et est donné dans le tableau 1. Pour chaque jeu de données dans le tableau, nous avons le TE obtenu par chaque modèle pour chacun des deux niveaux d'incertitude.
Focalisons nous premièrement sur les trois premiers modèles du tableau 1, il s'agit de UST_FLAT, DUST_UNIFORM et DUST_NORMAL. Nous résumons leur performance en utilisant les boites à moustaches (Boxplot) comme on peut le voir sur la figure 3. Le triangle vert représente la moyenne des TE sur l'ensemble des jeux de données. Les méthodes basées sur DUST (DUST_UNIFORM and DUST_NORMAL) ne sont pas significativement différentes l'une de l'autre. Cependant, DUST_UNIFORM est légèrement meilleur que DUST_NORMAL. UST_FLAT est meilleur que les modèles utilisant DUST, surtout lorsque l'incertitude est élevée (figure 3b). Le troisième quantile des methodes basées sur DUST est seulement à 0.6, tandis qu'il est à environ 0.8 pour la méthode UST. Ce qui signifie que UST obtient 80% de TE sur 25% des jeux de données. UST est meilleur sur plus de datasets que les méthodes utilisant DUST. En particulier, pour $c = 0.2$, le meilleur TE pour les modèles DUST est de 55% et 54% pour le dataset DodgerLoopGame et BME respectivement ; alors que UST donne 77% de TE pour DodgerLoopGame, et 82% pour BME.

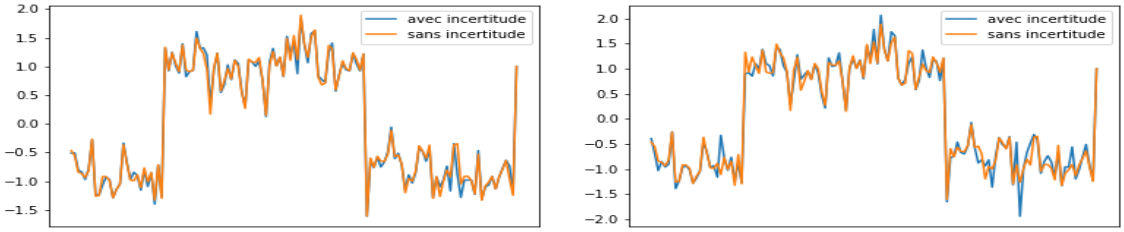
Examinons ensuite l'impact de la relation d'ordre utilisée par UST en comparant UST_FLAT et UST_FLAT_ST. Étant donné que cet ordre n'est pas total, UST_FLAT_ST ne s'est pas exécuté jusqu'au bout sur les 23 datasets ; Il y a eu des valeurs non comparables sur ces jeux de données soit lors de la phase d'apprentissage, soit lors de la phase de test. Les résultats obtenus sur les 6 jeux de données restant sont résumés dans la figure 4. L'ordre stochastique a amélioré les résultats pour $c = 0.1$ (figure 4a) ; En particulier, le TE est passé de 61% à 84% (une augmentation de 23%) pour le dataset Chinatown, et de 14% à 35% (une augmentation de 21%, mais pas assez pour passer les 50% de TE) sur le dataset SonyAIBORobotSurface1. Lorsque $c = 0.2$ (figure 4b), nous n'observons aucune amélioration, par contre il y a une baisse du TE sur le dataset BME qui passe de 82% à 69% (une baisse de 13%) lorsqu'on utilise l'ordre stochastique.

Avec UED, nous obtenons des résultats similaires, et même meilleurs qu'avec les mesures incertaines de la littérature. Cependant, notre approche a des limites. En effet, en utilisant une représentation applatie, le modèle de classification n'est pas réellement avisé de l'incertitude qui a été propagée. Bien que l'incertude soit une méta-caractéristique, elle est considérée par le modèle de classification supervisée comme une caractéristique à part entière. Nous pensons qu'une meilleure façon de prendre en compte l'incertitude propagée mènerait à de meilleures classifications. En particulier, un arbre de décision flou [16] pourrait être un bon candidat. Il permet à une instance de traverser chaque branche de l'arbre avec une probabilité. À cause de l'incertitude, il est intuitif qu'une instance puisse passer par plusieurs branches de l'arbre avec des probabilités non nulles. Ainsi, nous comptons explorer la possibilité de modéliser le problème de classification des séries temporelles incertaines en utilisant la logique floue [2, 12, 15].

6 Conclusion

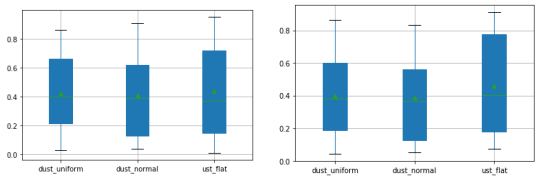
Le but de ce papier était de faire la classification des séries temporelles incertaines en utilisant l'approche par transformation shapelet. Pour le faire, nous avons utilisé les techniques de propagation de l'incertitude afin de dériver une mesure de dissimilarité incertaine appelée UED. Ensuite, nous avons adapté la méthode de classification par transformation shapelet au contexte des séries temporelles incertaines en utilisant UED et avons finalement proposé l'algorithme de classification par transformation shapelet incertain (UST). Nous avons effectué des expérimentations sur des jeux de données de la littérature et les résultats montrent l'efficacité de notre approche. En perspective, nous avons l'intention d'évaluer notre approche sur un vrai jeu de données contenant l'incertitude. Nous visons par

2. https://github.com/uea-machine-learning/tsml
3. https://github.com/frankl1/Uncertain-Shapelet-Transform
4. https://github.com/frankl1/Uncertain-Shapelet-Transform/blob/master/add-noise.ipynb



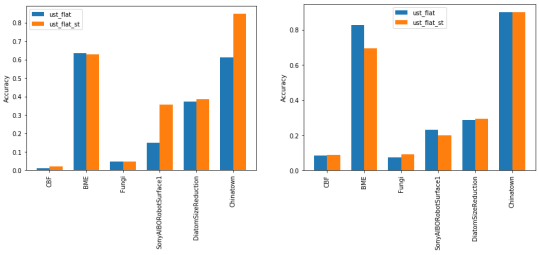
(a) $c = 0.1$ (b) $c = 0.2$

FIGURE 2 – Illustration de l’incertitude pour une instance du dataset CBF



(a) $c = 0.1$ (b) $c = 0.2$

FIGURE 3 – DUST vs UST



(a) $c = 0.1$ (b) $c = 0.2$

FIGURE 4 – UST_FLAT vs UST_FLAT_ST

exemple le dataset du challenge Plasticc [5]. Nous explorerons également la possibilité d’utiliser un modèle de classification flou afin de prendre mieux en compte l’incertitude après la propagation.

Remerciements

Ce travail a été soutenu par le Ministère français de l’Enseignement Supérieur, de la Recherche et de l’Innovation (MESRI), par le LabEx IMobS3 et le projet CNRS PEPS TransiXplore. Nous remercions également les relecteurs annonymes pour leurs remarques constructives.

Références

[1] Johannes Aßfalg, Hans-Peter Kriegel, Peer Kröger, and Matthias Renz. Probabilistic similarity search for uncertain time series. In *International Conference on Scientific and Statistical Database Management*, pages 435–443. Springer, 2009.

[2] Jamal Atif, Isabelle Bloch, and Céline Hudelot. Some relationships between fuzzy sets, mathematical morphology, rough sets, f-transforms, and formal concept analysis. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 24(Supplement-2) :1–32, 2016.

[3] Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh. The great time series classification bake off : a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31(3) :606–660, 2017.

[4] Isabelle Bloch. Model-based image interpretation under uncertainty and fuzziness. In Francesco Masulli, Gabriella Pasi, and Ronald Yager, editors, *Fuzzy Logic and Applications*, pages 171–183, Cham, 2013. Springer International Publishing.

[5] Michele Dallachiesa, Besmira Nushi, Katsiaryna Mirylenka, and Themis Palpanas. Uncertain time-series similarity : Return to the basics. *Proceedings of the VLDB Endowment*, 5(11) :1662–1673, 2012.

[6] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The ucr time series archive, 2018.

[7] Florence Dupin de Saint-Cyr and Henri Prade. Logical handling of uncertain, ontology-based, spatial information. *Fuzzy Sets Syst.*, 159(12) :1515–1534, 2008.

[8] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification : a review. *Data Mining and Knowledge Discovery*, 33(4) :917–963, 2019.

[9] Vanel Steve Siyou Fotso, Engelbert Mephu Nguifo, and Philippe Vaslin. Frobenius correlation based u-shapelets discovery for time series clustering. *Pattern Recognition*, 103 :107301, 2020.

[10] Jon Hills, Jason Lines, Edgaras Baranauskas, James Mapp, and Anthony Bagnall. Classification of time series by shapelet transformation. *Data Mining and Knowledge Discovery*, 28(4) :851–881, 2014.

5. https://www.kaggle.com/c/PLAsTiCC-2018

TABLE 1 – Taux d'exactitude de chaque modèle sur chaque dataset et pour chaque niveau d'incertitude

Datasets	**DUST_UNIFORM**		**DUST_NORMAL**		**UST_FLAT**		**UST_FLAT_ST**	
	c = 0.1	**c = 0.2**	**c = 0.1**	**c = 0.2**	**c = 0.1**	**c = 0.2**	**c = 0.1**	**c = 0.2**
ArrowHead	0.58	0.57	0.58	0.56	0.51	0.61	-	-
BME	0.62	0.54	0.62	0.54	0.63	0.82	0.62	0.69
CBF	0.06	0.11	0.06	0.11	0.01	0.08	0.02	0.08
Chinatown	0.84	0.76	0.89	0.78	0.61	0.90	0.84	0.90
DiatomSizeReduction	0.26	0.27	0.26	0.27	0.37	0.28	0.38	0.29
DistalPhalanxTW	0.10	0.11	0.10	0.12	0.04	0.10	-	-
DodgerLoopGame	0.61	0.55	0.52	0.53	0.71	0.77	-	-
DodgerLoopWeekend	0.75	0.60	0.90	0.83	0.72	0.63	-	-
ECGFiveDays	0.12	0.18	0.12	0.10	0.17	0.22	-	-
ECG200	0.23	0.20	0.23	0.24	0.19	0.18	-	-
Fungi	0.03	0.04	0.03	0.05	0.04	0.07	0.04	0.09
GunPoint	0.12	0.08	0.12	0.08	0.08	0.12	-	-
GunPointOldVersusYoung	0.86	086	0.52	0.52	0.94	0.91	-	-
InsectEPGSmallTrain	0.41	0.30	0.37	0.45	0.31	0.37	-	-
ItalyPowerDemand	0.21	0.12	0.21	0.12	0.08	0.10	-	-
MedicalImages	0.38	0.38	0.40	0.35	0.42	0.40	-	-
MelbournePedestrian	0.40	0.28	0.12	0.10	0.70	0.64	-	-
MiddlePhalanxOutlineAgeGroup	0.40	0.40	0.42	0.40	0.34	0.45	-	-
MiddlePhalanxOutlineCorrect	0.41	0.38	0.41	0.43	0.34	0.36	-	-
MiddlePhalanxTW	0.38	0.40	0.38	0.37	0.40	0.41	-	-
MoteStrain	0.83	0.69	0.83	0.62	0.80	0.79	-	-
Plane	0.29	0.31	0.29	0.28	0.28	0.26	-	-
ProximalPhalanxOutlineAgeGroup	0.12	0.12	0.12	0.13	0.12	0.16	-	-
ProximalPhalanxTW	0.07	0.08	0.06	0.07	0.08	0.08	-	-
SmoothSubspace	0.72	0.71	0.73	0.70	0.84	0.73	-	-
SonyAIBORobotSurface1	0.21	0.22	0.21	0.23	0.14	0.23	0.35	0.19
SyntheticControl	0.76	0.78	0.78	0.81	0.90	0.90	-	-
TwoLeadECG	0.75	0.71	0.78	0.70	0.86	0.79	-	-
UMD	0.65	0.72	0.65	0.72	0.90	0.85	-	-

[11] Vassilis G Kaburlasos, Eleni Vrochidou, Fotios Panagiotopoulos, Charalampos Aitsidis, and Alexander Jaki. Time series classification in cyber-physical system applications by intervals' numbers techniques. In *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6. IEEE, 2019.

[12] Le Xu, Mo-Yuen Chow, and L. S. Taylor. Data mining based fuzzy classification algorithm for imbalanced data. In *2006 IEEE International Conference on Fuzzy Systems*, pages 825–830, July 2006.

[13] Jason Lines, Sarah Taylor, and Anthony Bagnall. Time series classification with hive-cote : The hierarchical vote collective of transformation-based ensembles. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 12(5) :52, 2018.

[14] Albert W. Marshall, Ingram Olkin, and Barry C. Arnold. *Stochastic Ordering*, chapter 17, pages 693–756. Springer New York, New York, NY, 2010.

[15] P. Martín-Muñoz and F. J. Moreno-Velo. Fuzzycn2 : An algorithm for extracting fuzzy classification rule lists. In *International Conference on Fuzzy Systems*, pages 1–7, July 2010.

[16] Cristina Olaru and Louis Wehenkel. A complete fuzzy decision tree technique. *Fuzzy Sets and Systems*, 138(2) :221 – 254, 2003.

[17] Smruti R Sarangi and Karin Murthy. Dust : a generalized notion of similarity between uncertain time series. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 383–392. ACM, 2010.

[18] John R. Taylor. *An Introduction to Error Analysis : The Study of Uncertainties in Physical Measurements*. University Science Books, 2 sub edition, 1996.

[19] Lexiang Ye and Eamonn Keogh. Time series shapelets : a new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 947–956. ACM, 2009.

[20] Mi-Yen Yeh, Kun-Lung Wu, Philip S Yu, and Ming-Syan Chen. Proud : a probabilistic approach to processing similarity queries over uncertain data streams. In *Proceedings of the 12th International Conference on Extending Database Technology : Advances in Database Technology*, pages 684–695. ACM, 2009.

MDP augmentés pour la résolution de jeux de sécurité stochastiques

Romain Châtel[1], Abdel-Illah Mouaddib[1]

[1]Université de Caen Normandy, GREYC

[1]{romain.chatel, mouaddib.abdel-illah}@unicaen.fr

Résumé

Nous proposons une nouvelle approche théorique utilisant les MDP augmentés pour la résolution d'un Jeu de Sécurité Stochastique suivie de son évaluation expérimentale. La plupart des travaux mentionnés dans la littérature utilisent des techniques de Programmation Linéaire recherchant des équilibres de Stackelberg forts au sein de l'espace des stratégies de l'attaque et de la défense. Bien qu'efficaces, ces techniques sont coûteuses en temps de calcul et ont tendance à s'adapter difficilement à de très grosses instances. En fixant l'ensemble des stratégies possibles pour la défense, notre approche est en mesure d'utiliser le formalisme bien connu des MDP augmentés pour calculer une politique optimale d'un attaquant faisant face à un défenseur en patrouille. Deux modèles sont proposés pour adresser l'observabilité totale puis partielle sur le comportement et l'état de la défense. Les résultats expérimentaux obtenus lors de la simulation des modèles à observabilité totale valident notre approche mais soulignent également la nécessité de la raffiner pour prendre en charge de plus grosses instances ainsi que l'observabilité partielle.

Mots-clés

Théorie des jeux, Jeu à somme générale, MDP augmentés, Jeu de sécurité sur graphe, Jeux de stackelberg.

Abstract

We propose a novel theoretical approach for solving a Stochastic Security Game using augmented MDPs and an experimental evaluation. Most of the previous works mentioned in the literature focus on Linear Programming techniques seeking Strong Stackelberg Equilibria through the defender and attacker's strategy spaces. Although effective, these techniques are computationally expensive and tend to not scale well to very large problems. By fixing the set of the possible defense strategies, our approach is able to use the well-known augmented MDP formalism to compute an optimal policy for an attacker facing a defender patrolling. Two models are proposed to deal with total and partial observability of the defender behaviour and state. Experimental results on fully observable cases validate our approach and show good performances in comparison with optimistic and pessimistic approaches. However, these results also highlight the need of refinements to improve scalability and handle the partial observability cases.

Keywords

Game theory, Nonzero sum games, Augmented MDPs, Network Security Games, Stackelberg Games.

1 Introduction

Les jeux de sécurité fournissent un cadre formel ayant récemment émergé dans le domaine des systèmes multi-agents. Ils permettent de modéliser les interactions entre deux types de joueurs : défenseurs et attaquants lorsque la défense tente d'empêcher ses opposants d'attaquer une ou plusieurs cibles. Ils sont principalement appliqués dans l'organisation du déploiement de forces de sécurité (défenseurs) aussi bien civiles que militaires [8, 5, 14, 6]. Considérant que les attaquants ont habituellement tout le loisir de préparer leurs attaques, et donc de connaître la stratégie adoptée par la défense, la plus grande partie des travaux s'est concentrée sur le développement de Programmes Linéaires recherchant un équilibre de Stackelberg fort [11, 4] dans l'espace joint des stratégies de l'attaquant et du défenseur. Parmi les jeux de sécurité, les plus difficiles sont ceux sur graphe car les deux équipes (défense et attaque) sont mobiles sur un graphe rendant la recherche d'une stratégie pour chacune des deux parties fortement combinatoire [3]. Ils décrivent des situations où la défense tente d'empêcher la partie adverse d'atteindre une ou plusieurs cibles.

En fixant l'ensemble des stratégies possibles pour la défense, et en permettant des transitions entre ces stratégies au cours du temps, nous utilisons le formalisme des MDP augmentés pour résoudre un problème de patrouille sur graphe du point de vue d'un attaquant faisant face à une stratégie de défense néanmoins complexe.

Cet article est organisé comme suit : la section 2 définit les jeux et équilibres de Stackelberg, la section 3 décrit le formalisme des (PO)MDP, puis la section 4 définit le problème, la section 5 présente les formalisations du problème restreint à l'observabilité totale, la section 6 étend les formalisations de la section 5 à l'observabilité partielle sur la stratégie de la défense, la section 7 étend les formalisations de la section 6 à l'observabilité partielle sur l'état de la défense, la section 8 présente le dispositif expérimental et la section 9 expose les résultats obtenus. Finalement, la section 10 conclut cet article et évoque les travaux futurs.

2 Jeux et équilibres de Stackelberg

Les jeux de Stackelberg forment une classe de jeux à deux joueurs dans laquelle un joueur est désigné comme étant le meneur et le second le suiveur. Le meneur choisit une stratégie, celle ci est communiquée au suiveur qui à son tour choisit sa stratégie en intégrant cette information lors de sa décision (stratégie best-response). Cette situation donne lieu à une notion d'équilibre analogue de l'équilibre de Nash appelée équilibre de Stackelberg. Elle peut être subdivisée en deux sous-équilibres : l'équilibre de Stackelberg fort lorsqu'un équilibre est choisi parmi plusieurs en tranchant en faveur du meneur et l'équilibre de Stackelberg faible lorsque l'on tranche en faveur du suiveur. L'existence de l'équilibre de Stackelberg fort est garantie dans tous les jeux de Stackelberg tandis que celle du faible ne l'est pas [1]. Plus formellement, un équilibre de Stackelberg fort est définit comme suit :

Définition 1. *Soit deux joueurs, M le meneur et S le suiveur, jouant respectivement les statégies m et $g_s(m)$ et recevant une utilité $U_M(m, g_s(m))$ et $U_S(m, g_s(m))lp$. Le couple $\langle m, g_s(m)\rangle$ forme un équilibre de Stackelberg fort si et seulement si :*

1. *Le meneur joue une stratégie best-response :*

$$U_M(m, g_s(m)) \geq U_M(m', g_s(m')),\ \forall m'.$$

2. *Le suiveur joue une stratégie best-response :*

$$U_S(m, g_s(m)) \geq U_S(m, g'_s(m)),\ \forall m, g'_s.$$

3. *En cas d'indifférence, le suiveur tranche en faveur du meneur :*

$$U_M(m, g_s(m)) \geq U_M(m, s),\ \forall m, s,$$

où s appartient à l'ensemble des stratégies best-responses du suiveur.

3 Le formalisme des (PO)MDP

Les (PO)MDP sont des modèles permettant à un agent de « décider comment agir dans un environnement accessible stochastique et faisant appel à un modèle de transition connu » (Russel and Norvig [7], p .500). Un MDP est composé des éléments suivants :

- Un ensemble fini d'états, $\mathcal{S} = \{s_1, s_2, \dots s_{|\mathcal{S}|}\}$, représentant l'ensemble des états atteignables par l'agent ;
- Un ensemble fini d'actions, $\mathcal{A} = \{a_1, a_2, \dots a_{|\mathcal{A}|}\}$, représentant toutes les actions réalisables à chaque instant par l'agent ;
- Une fonction de transition $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ exprimant la probabilité, pour l'agent, de transiter de l'état s_i à l'état s_k lorsqu'il réalise l'action a_j : $\mathcal{T}_{ijk} = P(s_k|s_i, a_j)$ et donnant ainsi la dynamique du système ;
- Une fonction de récompense immédiate $\mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ motivant l'agent à atteindre son but.

La résolution des MDP consiste à chercher des politiques $\pi : \mathcal{S} \mapsto \mathcal{A}$ permettant à l'agent de savoir quelle action accomplir pour chacun des état du système. Elles doivent, pour être optimales, satisfaire l'opérateur d'optimalité de Bellman, ie maximiser la récompense cumulée espérée par l'agent pour chaque état :

$$V_i^{\pi *} = V^{\pi *}(s_i) = \max_{a_j \in \mathcal{A}} [\mathcal{R}_{ij} + \gamma \sum_{s_k \in \mathcal{S}} \mathcal{T}_{ijk} V_k^{\pi *}]$$

Le facteur d'atténuation $\gamma \in [0, 1]$ permet ici un compromis entre la récompense obtenue immédiatement par l'agent et celle qu'il espère obtenir dans le futur. Lorsque $\gamma < 1$ et que la fonction de récompense est bornée, l'existence d'une politique optimale est garantie [2].
Cette première définition est dite à observabilité totale : l'état du système est, à chaque instant, directement observable par l'agent. Les POMDP permettent d'ajouter l'observabilité partielle sur l'état courant en complétant le formalisme précédent avec :

- Un ensemble fini d'observations, $\mathcal{Z} = \{z_1, z_2, \dots z_{|\mathcal{Z}|}\}$, donnant ce que peut observer l'agent à chaque instant sur son état caché ;
- Une fonction d'observation $\mathcal{TO} : \mathcal{S} \times \mathcal{A} \times \mathcal{Z} \mapsto [0, 1]$, donnant $\mathcal{TO}_{ijk} = P(z_k|s_i, a_j)$ et reliant les observations de l'agent à ses états sous-jacents.

Dans ce formalisme, l'agent reçoit après chaque action une observation sur le nouvel état dans lequel il se trouve. La succession des paires $(a, z) \in \mathcal{A} \times \mathcal{Z}$ forme l'historique du système, $h^t = \langle z^0, a^0, \dots, a^{t-2}, z^{t-1}, a^{t-1}, z^t\rangle$ représente l'historique à l'instant t vu par l'agent. Il est commode de compresser l'espace des historiques $\mathcal{H}$ en un espace de croyances sur l'état de l'agent $\mathcal{B} : [0, 1]^{|\mathcal{S}|}$ tel que $b^t = (b_1, b_2, \dots, b_k, \dots, b_{|\mathcal{S}|}) \in \mathcal{B}$ représente l'état de croyance de l'agent à l'instant t sur son état caché avec $b_k = P(s_k^t|h^t)$. Les politiques $\pi : \mathcal{B} \mapsto \mathcal{A}$ permettent alors à l'agent de prendre à chaque instant une décision.
Cette transformation conduit à la définition d'un nouveau formalisme, les *Belief MDP* construits à partir des éléments suivants :

- $\mathcal{B}$, l'ensemble des états de croyance de l'agent ;
- $\mathcal{A}$, l'ensemble fini des actions possibles ;
- $\mathcal{T}^b : \mathcal{B} \times \mathcal{A} \times \mathcal{B} \mapsto [0, 1]$, la fonction de transition des états de croyance ;
- $\mathcal{R}^b : \mathcal{B} \times \mathcal{A} \mapsto \mathbb{R}$, la fonction de récompense immédiate associée aux états de croyance ;
- $\zeta^b : \mathcal{B} \times \mathcal{A} \times \mathcal{Z} \mapsto \mathcal{B}$, la fonction mettant à jour l'état de croyance à chaque observation reçue.

3.1 Les (PO)MDP augmentés

Les (PO)MDP peuvent s'interpréter comme des jeux stochastiques à un joueur, où l'évolution et éventuellement la perception de l'environnement sont incertaines. Les (PO)MDP augmentés permettent d'étendre ce cadre à des situations plus riches en complétant les espaces d'états, d'actions, d'observations et les modèles de transitions et de récompenses avec des informations ne décrivant pas directement le joueur et son environnement. Il est ainsi possible, par exemple, de décrire des jeux stochastiques à plu-

sieurs joueurs en considérant les espaces d'états, d'actions et d'observations joints de tous les agents.

4 Définition du problème

Notre travail consiste à résoudre le jeu de sécurité stochastique $\langle \{I, P\}, \mathcal{S} = \mathcal{S}^I \times \mathcal{S}^P \times \Pi^P, \mathcal{A} = \mathcal{A}^I \times \mathcal{A}^P, \mathcal{R}, \mathcal{T} \rangle$. Par exemple, sur une grille de taille $N \times N$, un drone patrouilleur P surveille. Un second drone intrus I tente de rejoindre une case sur la grille (la cible τ) sans se faire repérer par le premier (voir Figure 1). L'attaquant dispose des informations suivantes sur le défenseur :

1. Le défenseur est doté d'un champ de perception $\mathcal{F}^P : \mathcal{S}^P \times \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$. Il donne l'intensité avec laquelle le patrouilleur perçoit chaque case (x, y) de la grille en fonction de son état interne $p \in \mathcal{S}^P$.
2. Le défenseur dispose d'un ensemble $\mathcal{A}^P$ d'actions possibles.
3. Le défenseur utilise à chaque instant une politique de patrouille k choisie parmi un ensemble Π^P de politiques possibles.
4. Le défenseur utilise un modèle de transition $\mathcal{T}^P$.
5. Le défenseur est susceptible de transiter à tout instant d'une politique de patrouille à une autre selon un modèle de transition inter-politique θ^P.

$\mathcal{T}^P$ et θ^P sont deux composantes du modèle de transition du jeu $\mathcal{T} = \{P(s'|s, a^I, a^P) \,|\, \forall (s', s, a^I, a^P) \in [\mathcal{S}^I \times \mathcal{S}^P \times \Pi^P]^2 \times \mathcal{A}^I \times \mathcal{A}^P\}$ où $\mathcal{A}^I$ est l'ensemble fini des actions possibles pour l'intrus et $\mathcal{S}^I$ l'ensemble fini de ses états internes.

Le but du jeu est de trouver une stratégie pour l'attaquant permettant d'atteindre rapidement la cible tout en minimisant le risque de se faire repérer par le patrouilleur. Ce but détermine le modèle de récompense du jeu $\mathcal{R} = \mathcal{R}^I \times \mathcal{R}^P$.

En supposant pour le défenseur des politiques markoviennes stationnaires et en utilisant le lemme 1 nous pouvons limiter notre étude à l'utilisation de MDP augmentés tout en garantissant l'optimalité des solutions trouvées.

Lemme 1 (Vorobeychik and Singh [12], 2012)**.** *Pour tout jeu de Stackelberg stochastique à somme générale et à facteur d'atténuation, si le meneur suit une politique markovienne stationnaire alors il existe une politique déterministe markovienne et stationnaire qui est une politique best-response pour le suiveur.*

Dans le but d'aborder ce problème pas à pas, celui-ci a été subdivisé en sous-problèmes de difficulté croissante. Dans un premier temps, nous supposons l'observabilité totale sur l'état et la politique de patrouille courante de la défense. Puis nous étendons ce modèle à l'observabilité partielle sur cette politique. Enfin, nous introduisons un second niveau d'incertitude : l'observabilité partielle sur l'état du patrouilleur.

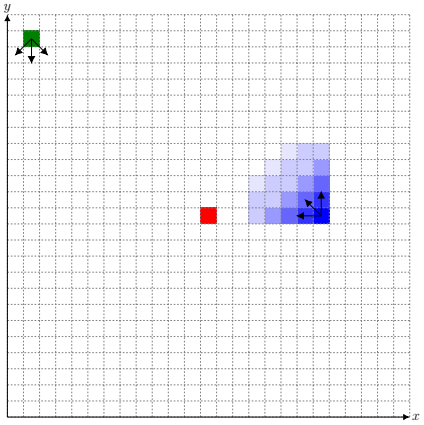


FIGURE 1 – Le jeu : l'intrus (vert) est situé dans le coin supérieur gauche, le patrouilleur (bleu) est situé à droite accompagné de son champ de perception (dégradé de bleu) et la cible (rouge) est située au centre de la grille.

5 Modèles pour l'observabilité totale

5.1 Formalisation du risque

Nous commençons par formaliser le risque pris par l'intrus vis à vis de la présence du patrouilleur comme une fonction $D^P : \mathcal{S}^I \times \mathcal{S}^P \mapsto [0, 1]$. Elle représente alors la probabilité que l'intrus soit détecté par le patrouilleur connaissant l'état $(i, p) \in \mathcal{S}^I \times \mathcal{S}^P$, des deux drones :

$$D^P_{ip} = P(\text{DETECTED} = T|i, p) = \frac{\mathcal{F}^P_{pi_\mathcal{X} i_\mathcal{Y}}}{\sum_{i' \in \mathcal{S}^I} \mathcal{F}^P_{pi'_\mathcal{X} i'_\mathcal{Y}}},$$

où $i_\mathcal{X}$ et $i_\mathcal{Y}$ désignent respectivement les valeurs sur $\mathcal{X}$ et $\mathcal{Y}$ des états i de l'intrus.

5.2 Formalisation de la récompense

Afin d'inciter l'agent attaquant à atteindre la cible τ, la fonction de récompense $\mathcal{R}^I$ lui délivre la récompense suivante :

$$\mathcal{R}^I_i = \begin{cases} 1 \text{ si } (i_\mathcal{X}, i_\mathcal{Y}) == \tau, \\ 0 \text{ sinon.} \end{cases}$$

5.3 Modèle de transition agrégé du patrouilleur

Les politiques du patrouilleur ainsi que son modèle de transitions sont agrégés afin de construire un modèle de transition agrégé T^P pour le patrouilleur, tel que :

$$T^P_{p'kp} = \mathcal{T}^P_{p'\pi^P_k(p)p} = P(p'|\pi^P_k(p), p).$$

5.4 Modèle de transition inter-politique du patrouilleur

Le modèle de transition inter-politique du patrouilleur capture la dynamique de transition entre les différents comportements possibles du patrouilleur. Afin de prendre en compte des évènements mettant en jeu l'état de l'intrus par rapport à celui du patrouilleur, il est défini comme une

fonction $\Pi^P \times \Pi^P \times S^P \times S^I \mapsto [0,1]$ et représente la probabilité $P(k'|k,i,p)$, pour le patrouilleur, de transiter d'une politique k à une politique k' lorsque les drones sont dans les états respectifs i et p.

5.5 Fonction d'utilité de l'intrus

La fonction d'utilité de l'intrus permet d'exprimer le compromis entre les objectifs parfois contradictoires d'atteindre la cible rapidement et celui de ne pas être repéré par le drone de défense. Elle est donnée par :

$$U^I_{ip} = \beta \mathcal{R}^I_i + (1-\beta)(1-D^P_{ip}),$$

le paramètre β permettant de moduler la prise de risque de l'intrus.

5.6 MDP résultant

Le problème peut alors être résolu en définissant le MDP $\langle \mathcal{S}, \mathcal{A}^I, \mathcal{T}^I, T^P, \theta^P, U^I \rangle$ où $\mathcal{S} : \mathcal{S}^I \times \mathcal{S}^P \times \Pi^P$, et $\mathcal{T}^I$ est le modèle de transition de l'intrus. En utilisant le critère γ-pondéré, requis par le lemme 1 et permettant à l'intrus de privilégier les plus courts chemins vers la cible, l'opérateur d'optimalité de Bellman s'écrit alors :

$$V^*_{ipk} = U^I_{ip} + \gamma \max_{a \in \mathcal{A}^I} \sum_{i'p'k'} \mathcal{T}^I_{i'ai} T^P_{p'kp} \theta^P_{k'kip} V^*_{i'p'k'}. \quad (1)$$

5.7 Formulation pseudo maximin

Traditionnellement, dans les jeux à deux joueurs et à somme nulle, les notions d'équilibre de Nash et d'équilibre de Stackelberg fort se confondent, et de tels jeux peuvent être résolus en appliquant le théorème du *minimax* / *maximin* de Von Neuman. Parce que la stratégie du défenseur est une variable d'entrée du problème, ce jeu n'est pas, dans le cas général, à somme nulle, nous ne pouvons donc pas utiliser cette équivalence. Nous proposons cependant une seconde version de ce modèle à observabilité totale sous une forme pouvant rappeler le principe du *maximin*. Cette formulation consiste pour l'attaquant à considérer que le but de la défense est de le mettre en danger. Pour cela nous attribuons à la défense une fonction de valeur traduisant cette mise en danger minimisée par l'attaquant :

$$V^{P*}_{pik} = D^P_{ip} + \gamma \min_{a \in \mathcal{A}^I} \sum_{i'p'k'} \mathcal{T}^I_{i'ai} T^P_{p'kp} \theta^P_{k'kip} V^{P*}_{p'i'k'}. \quad (2)$$

Cette fonction de valeur est ensuite soustraite de l'opérateur de Bellman de l'attaquant :

$$V^*_{ipk} = \beta \Big[\mathcal{R}^I_i + \gamma \max_{a \in \mathcal{A}^I} \sum_{i'p'k'} \mathcal{T}^I_{i'ai} T^P_{p'kp} \theta^P_{k'kip} V^*_{i'p'k'} \Big] - (1-\beta) \frac{V^{P*}_{pik}}{\max_{s \in \mathcal{S}} V^{P*}_s} \quad (3)$$

Chacun des protagonistes pénalisant ainsi fortement son adversaire, nous espérons obtenir des politiques plus prudentes qu'avec la formulation précédente.

6 Extension à l'observabilité partielle sur la politique du patrouilleur

La succession au cours du temps des couples (i,p) d'états de la défense et de l'attaquant permettant d'inférer la succession des politiques k utilisées par la défense, ces couples d'états sont assimilables à des observations sur la politique qu'elle utilise. Nous intégrons alors l'incertitude sur la politique utilisée à chaque instant par le drone de surveillance en transformant le MDP précédent directement en *Belief MDP* à observabilité mixte $\langle \mathcal{S}, \mathcal{A}^{\mathcal{I}}, \mathcal{T}^I, T^P, U^I, \theta^P, \zeta_\pi \rangle$ avec :

- $\mathcal{S} : \mathcal{S}^I \times \mathcal{S}^P \times \mathcal{B}^P_\pi$ où $\mathcal{B}^P_\pi$ est l'ensemble des états de croyance de l'attaquant sur la politique utilisée par la défense ;
- $\zeta_\pi : \mathcal{S} \mapsto \mathcal{B}^P_\pi$, est la fonction mettant à jour l'état de croyance à chaque observation reçue telle que $b'_\pi = \zeta_\pi(i,p,b_\pi) = (b'_{k_1}, b'_{k_2}, \dots, b'_{k|\Pi^P|})$, et

$$b'_k = P(k'|i,p,b_\pi) = \sum_k b_\pi(k) \theta_{k'kip}.$$

L'opérateur d'optimalité de Bellman s'écrit alors :

$$V^*_{ipb_\pi} = U^I_{ip} + \gamma \max_{a \in \mathcal{A}^I} \sum_{i'p'} \mathcal{T}^I_{iai'} \Big[\sum_k T^P_{pkp'} b_\pi(k) \Big] V^*_{i'p'b'_\pi}. \quad (4)$$

La formulation pseudo maximin s'étend sur le même principe à l'observabilité partielle sur la politique du patrouilleur.

7 Extension à l'observabilité partielle sur l'état du patrouilleur

Nous intégrons à présent l'observabilité partielle sur l'état de la défense en utilisant un oracle (un satellite par exemple) fournissant des observations sur cet état. Cela donne le *Belief MOMDP* $\langle S, \mathcal{A}^I, \mathcal{Z}, \mathcal{T}^I, U^I, TO, \theta^P, \zeta_s, \zeta_\pi \rangle$ suivant :

- $\mathcal{S} : \mathcal{S}^I \times \mathcal{B}^P_s \times \mathcal{B}^P_\pi$, avec $\mathcal{B}^P_s$ l'ensemble des états de croyance b_s sur l'état du patrouilleur ;
- $U^I_{ib_s} = \beta \mathcal{R}^I_i + (1-\beta)\Big[1 - \sum_p b_s(p) D^P_{ip}\Big]$;
- $TO : \mathcal{Z} \times \mathcal{S}^P \times \Pi^P \mapsto [0,1]$, est la fonction d'observation de l'oracle ;
- $\zeta_s(b_s, b_\pi, z) = b'_s = (b'_{p_1}, b'_{p_2}, \dots, b'_{p|\mathcal{S}^P|})$, avec

$$b'_p = P(p'|b_s, b_\pi, z) = \frac{\sum_{k,p} TO_{zp'k} b_\pi(k) T^P_{pkp'} b_s(p)}{\sum_{p,p''k} \Big[TO_{zp''k} T^P_{pkp''} b_\pi(k) \Big] b_s(p)}$$

  la fonction de mise à jour de $b_s \in \mathcal{B}^P_s$;
- $\zeta_\pi(i, b_s, b_\pi) = b'_\pi = (b'_{k_1}, b'_{k_2}, \dots, b'_{k|\Pi^P|})$, et

$$b'_k = P(k'|i, b_s, b_\pi) = \sum_k b_\pi(k) \sum_p b_s(p) \theta_{k'kip}$$

  est la fonction de mise à jour de $b_\pi \in \mathcal{B}^P_\pi$.

L'opérateur de Bellman s'écrit finalement :

$$V^*_{ib_s b_\pi} = U^I_{ib_s} + \gamma \max_{a \in \mathcal{A}^I} \sum_{i'} \mathcal{T}^I_{iai'} \sum_{zp'k} TO_{zp'k} \sum_{p} T^P_{pkp'} b_\pi(k) b_s(p) V^*_{i' b'_s b'_\pi}. \quad (5)$$

Là encore cette extension peut être dérivée selon le même principe à partir de la formulation pseudo maximin.

Afin d'évaluer nos formulations à observabilité totale, nous prenons le parti d'utiliser un des algorithmes de résolution exacte les plus décrits dans la littérature : *Value Iteration*. Nous devons toutefois faire remarquer que la non-convexité des fonctions de valeur décrites précédemment influencera à la baisse la qualité des solutions trouvées.

8 Expérimentations

8.1 Dispositif

Seules les formulations à observabilité totale ayant été évaluées, nous nous contenterons d'expliciter ici les éléments composant une simulation simple à observabilité totale du problème initial. Les formalismes exposés en section 5 sont suffisamment abstraits pour autoriser une modélisation plus complexe du monde et du comportement de la défense dans le cadre d'une implémentation du problème plus fidèle à la réalité. En effet, seul l'espace d'états de l'intrus est contraint à contenir sa position sur la grille. La définition de l'espace d'états du défenseur, de ses stratégies de défense et des différents modèles de transitions du jeu peuvent donc être librement ajustés sous réserve de satisfaire le lemme 1.

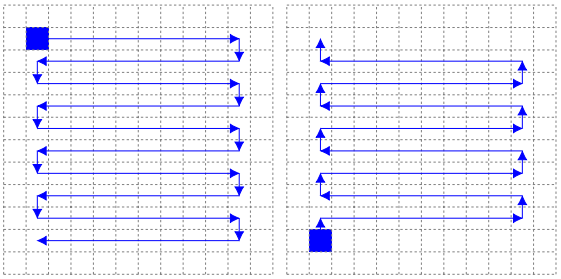
Pour les besoins de la simulation, nous dotons l'intrus d'un espace d'états $\mathcal{S}^I : \mathcal{X} \times \mathcal{Y} \times \mathcal{O}$ où $\mathcal{O} = \{1, \ldots, 8\}$ dénote l'ensemble fini des 8 orientations cardinales. Les politiques du patrouilleur sont calculées sur l'espace d'états $\mathcal{S}^P : \mathcal{X} \times \mathcal{Y} \times \mathcal{O} \times \mathcal{L} \times \mathcal{M}$, avec $\mathcal{L} = \mathcal{Y}$ et $\mathcal{M} = \{0, 1\}$. Deux politiques sont attribuées à la défense. La première consiste à patrouiller sur la grille en suivant un parcours en $Boustrophedon$ comme le montre la Figure 2. La variable d'état $l \in \mathcal{L}$ désigne le numéro de la ligne que doit compléter le défenseur tandis que la variable $m \in \mathcal{M}$ renseigne sur le caractère ascendant ou descendant du parcours. Le second comportement $Cover$ consiste à protéger la cible. Sa politique est calculée en utilisant une fonction de récompense $\mathcal{R}^P$ similaire à $\mathcal{R}^I$. Le patrouilleur est également doté d'un champ de perception construit à partir de la fonction F_{xy}, résultat d'un mélange de fonctions logistiques :

$$\begin{aligned}
f(t, a, r) &= \frac{1}{1 + ae^{-rt}}, \\
g(t, a, r, L) &= \max(f(t, a, r) - f(L, a, r), 0), \\
h(t, a, r, L) &= \frac{g(t, a, r, L)}{g(0, a, r, L)}, \\
F_x(x, a, r, L) &= h(|x|, a, r, L), \\
F_y(y, a, r, L) &= \begin{cases} 0 & \text{if } y \leq 0, \\ h(y, a, r, L) & \text{otherwise} \end{cases}, \\
F_{xy}(x, a_x, r_x, L_x, & \\
\quad y, a_y, r_y, L_y) &= \max(F_x(x, a_x, r_x, L_x) + \\
& \qquad F_y(y, a_y, r_y, L_y) - 1, 0).
\end{aligned}$$

Cette fonction de perception permet d'engendrer différents champs de perception en faisant varier ses paramètres $a_x, r_x, L_x, a_y, r_y, L_y$. L_x et L_y contrôlent la profondeur de perception vers l'avant pour L_y et sur les flancs pour L_x. Les paramètres a et r ont quant à eux une action conjointe sur la rapidité de décroissance de la perception en fonction de l'éloignement au point $(0, 0)$. Comme le montre la Figure 3, F_{xy} représente l'intensité avec laquelle le patrouilleur perçoit chaque case de la gille lorsqu'il se situe en $(0, 0)$ et est orienté au nord. Le champ de perception $\mathcal{F}^P_{\mathcal{XY}}$ est donc obtenu de F_{xy} à partir d'une translation de vecteur $(p_\mathcal{X}, p_\mathcal{Y})$ puis d'une rotation d'angle $\omega = p_\mathcal{O} - \frac{\pi}{2}$. Enfin, nous fixons le modèle inter-politique du patrouilleur de manière à ce qu'il dépende de la probabilité de percevoir l'intrus : lorsque celle-ci est supérieure à 0.3 le patrouilleur défend la cible ($Cover$), sinon la transition vers $Cover$ (resp. $Boustrophedon$) suit une loi de Bernoulli de paramètre D^P_{ip}.

	$k' = Cover$	$k' = Boustrophedon$
$D^P_{ip} > 0.3$	1	0
$D^P_{ip} \leq 0.3$	D^P_{ip}	$1 - D^P_{ip}$

TABLE 1 – Modèle de transition inter-politique du patrouilleur $P(k'|k, i, p)$



(a) Parcours descendant (b) Parcours ascendant

FIGURE 2 – Parcours en boustrophedon du patrouilleur

Par simplicité, les deux drones partagent le même espace d'action : $\mathcal{A}^I = \mathcal{A}^P = \mathcal{A}$. Pour chaque position et orientation sur la grille, ils peuvent avancer vers les trois cases leur faisant face (voir Figure 4) :

$$\mathcal{A} = \{\text{GO-LEFT, GO-STRAIGHT, GO-RIGHT}\}.$$

Pour finir, la stochasticité de l'environnement représente, par exemple, l'action du vent sur les drones : nous fixons donc pour chaque action une probabilité de succès égale à 0.9 ainsi qu'une probabilité de dévier le drone à gauche ou à droite de 0.05.

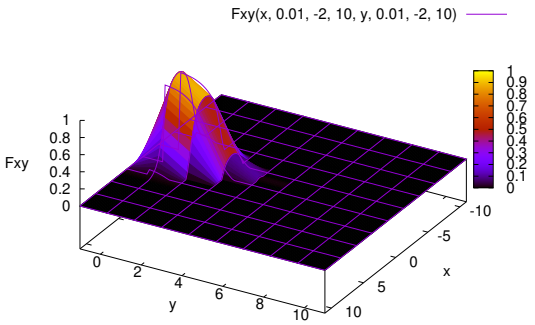


FIGURE 3 – La fonction de perception utilisée : mélange de fonctions logistiques

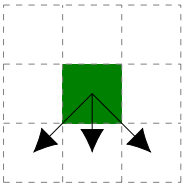
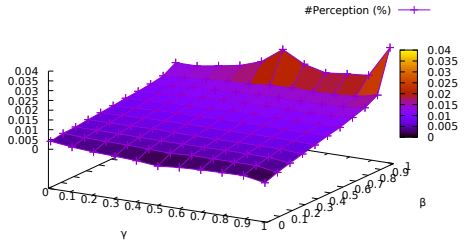
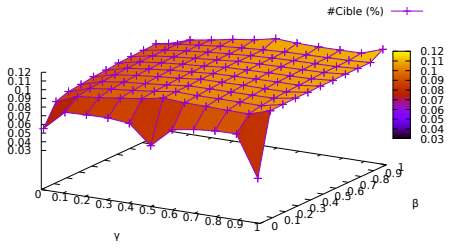


FIGURE 4 – Les trois actions possibles (orientation au sud).



(a) #Perception (%)



(b) #Cible (%)

FIGURE 5 – Nombre de perceptions (a) et d'arrivées à la cible (b) en fonctions des hyper-paramètres γ et β.

Nous basons notre évaluation sur le scénario suivant : la cible est située au centre de la grille $(\frac{N}{2}, \frac{N}{2})$, l'intrus est initialement en position $(1, N)$ orienté à l'est et le patrouilleur en (N, N) orienté à l'ouest. Il démarre sa patrouille par le comportement $Boustrophedon$ avec un parcours descendant en réalisant la ligne numéro N. Nous ne définissons pas d'état but : la simulation se déroule jusqu'à atteindre 30000 pas de temps.
Notre implémentation utilise la bibliothèque C++ MADP Toobox [9]. Nous évaluons notre preuve de concept avec un des algorithmes de résolution exacte les plus connus : *Value Iteration*. Les calculs sont effectués sur un serveur équipé de 4 processeurs AMD Opteron 6282SE (2.6 GHz) totalisant 64 coeurs et 512 GB de RAM. Il faut cependant noter que MADP n'utilise qu'un seul coeur lors de l'exécution de *Value Iteration*.

8.2 Protocole d'évaluation

Nous commençons dans un premier temps par rechercher des valeurs optimales pour les hyper-paramètres γ et β représentant respectivement le facteur d'atténuation du MDP de l'intrus et son coefficient de prise de risque. Pour cela, nous étudions les courbes montrant l'évolution du nombre moyen de perceptions et d'atteintes de la cible par pas de temps en fonction de γ et β (Figure 5). Elles sont calculées après 50 simulations du scénario pendant 30000 pas de temps sur une grille de taille $N = 8$.
Ces courbes mettent en lumière deux zones d'intérêt, centrées autour des points $(\gamma = 0.5, \beta = 0)$ et $(\gamma = 1, \beta = 0)$, au regard du critère de sélection suivant :

$$\pi' \geq \pi \Rightarrow \begin{cases} C(\pi') \geq C(\pi), \text{ si } P(\pi') = P(\pi), \\ P(\pi') < P(\pi), \text{ sinon,} \end{cases}$$

avec $C = \#Cible$ et $P = \#Perception$. Ce critère de sélection reflète l'objectif énoncé en section 4 : tandis que γ et le modèle de récompense favorisent la recherche de plus courts chemins vers la cible, il s'attache à minimiser en premier lieu le risque pris par l'intrus. Après recherche par Gridsearch dans ces zones d'intérêt, nous retenons les valeurs $\gamma = 0.99$ et $\beta = 1e^{-4}$.
Finalement, nous étudions l'évolution du temps de calcul, du nombre moyen de passages sur la cible, du nombre moyen de perceptions et de l'intensité moyenne de perception en fonction de la taille de la grille. Nous comparons notamment la formulation (1) « Normal » et la formulation (2) (3) « Maximin » avec deux formulations (1) « Beta = 0 » (resp. « Beta = 1 ») où $\beta = 0$ (resp. $\beta = 1$) ainsi qu'à une politique où l'intrus choisit une action selon une loi uniforme « Random ».

9 Résultats

Sans surprise, le temps de calcul par *Value Iteration* croit à un rythme exponentiel avec la taille de la grille (Figure 6). En effet, la taille de l'espace d'états est polynomiale en la taille de la grille :

$$|\mathcal{S}| = |\mathcal{S}^I| \times |\mathcal{S}^P| \times |\Pi^P| = 256N^5.$$

L'explosion combinatoire est cependant limitée par le nombre réduit de transitions possibles sur la grille ($\mathbb{O}(|\mathcal{S}| \times |\mathcal{A}|)$). Cela rend possible la résolution de grilles de taille modeste ($\approx 1.2e^4$ s pour $N = 10$) avec $|\mathcal{S}| = 2.56e^7$ états). Ce résultat est identique pour les formulations « Normal » et « Maximin ». Il s'explique par l'usage du para-

mètre $\beta = 1e^{-4}$ qui accélère grandement la convergence de (3) ajoutant seulement 3 itérations au calcul de « Maximin ». Il est à noter que l'échelle de la figure ne permet pas de visualiser les écart-types négligeables (de l'ordre de 0.1 s) devant les valeurs moyennes mesurées.

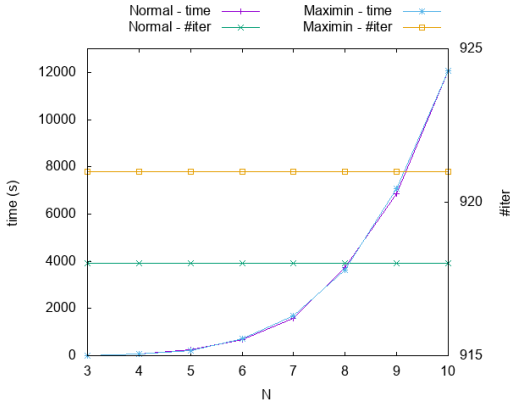


FIGURE 6 – Temps de calcul et nombre d'itérations

La Figure 7 montre quant à elle des taux de passage sur la cible intéressants. Le scénario « Normal » approche le plus favorable (« Beta = 1 ») lorsque la taille de grille augmente (8.31% vs 10.5% pour $N = 10$. Le scénario « Maximin » est conforme à nos attentes : il est plus performant que « Beta = 0 » (prise de risque minimale) et « Random » mais moins que « Normal », sa formulation le rendant plus paranoïaque. Ici encore, les écart-types de l'ordre de 0.1% pour les plus élevés sont difficilement perceptibles sur les différentes courbes.

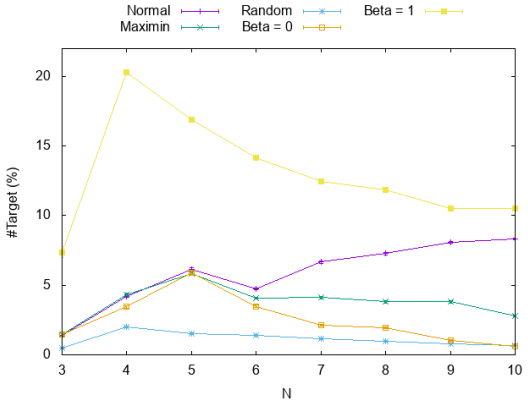


FIGURE 7 – Nombre de passages sur la cible (%)

Enfin, les Figures 8 et 9 montrent des taux de perceptions inférieurs à 0.1% dès que la grille est suffisamment grande ($N = 8$). Cela traduit le fait que l'intrus obtient suffisamment de place pour évoluer autour du patrouilleur. Ici encore le scénario « Normal » s'approche du plus favorable (« Beta = 0 ») avec un taux de perception inférieur à 0.001% pour $N = 10$. Bien que le scénario « Maximin » suit globalement la même tendance que « Normal », nous notons que l'intrus s'y fait plus souvent percevoir contrairement à notre hypothèse. Nous interprétons ce résultat par l'importance prise par le risque à long terme dans la formulation (2) (3). En favorisant les déplacements minimisant le risque à long terme moyen, le risque à court terme se trouve négligé et conduit à une dégradation des performances par rapport à la formulation (1). Une analyse plus fine reste cependant nécessaire afin d'infirmer ou de confirmer cette interprétation.

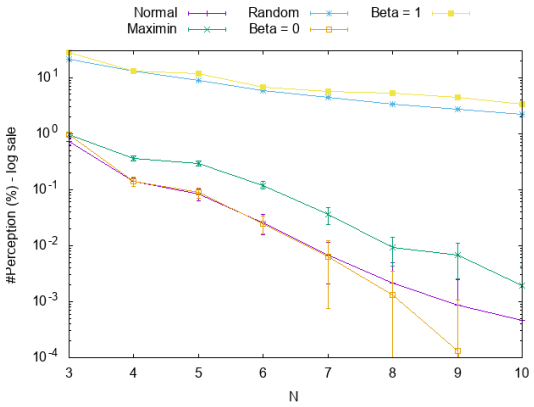


FIGURE 8 – Nombre de perceptions (%)

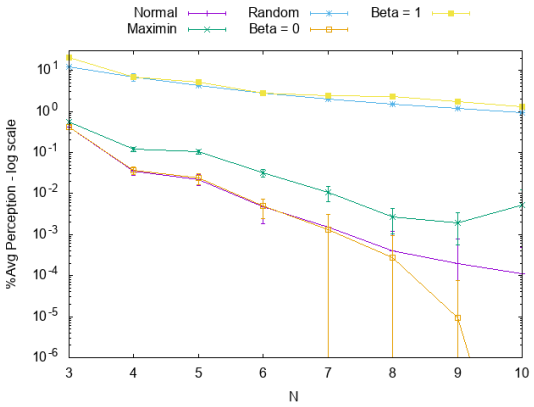


FIGURE 9 – Intensité moyenne de perception

10 Conclusion

Nous avons proposé une nouvelle approche pour la résolution d'un jeu de patrouille par des MDP augmentés à partir de trois modélisations intégrant au fur et à mesure différents niveaux d'observabilité partielle. Les résultats des expérimentations à observabilité totale valident notre approche. Bien que les formulations actuelles ne permettent pas de résoudre le problème sur de larges grilles ($N > 10$) comme nous l'ambitionnions en Figure 1, elles permettent de traiter des tailles de grille inaccessibles par les techniques classiques de résolution de jeux stochastiques. Les nombreux travaux de la littérature sur les MDP nous conduisent à envisager d'utiliser des techniques de compression de l'espace d'états [10] ainsi que de décomposition hiérarchique [13] afin de profiter des différentes sy-

métries présentes et ainsi résoudre notre problème sur des grilles de tailles plus conséquentes ($N \geq 50$). Nous prévoyons également d'expérimenter des méthodes de résolution autres que *Value Iteration*, notamment des algorithmes de résolution approchée. Bien que certains d'entre eux ne nécessitent pas d'hypothèses sur la convexité des fonctions de valeur utilisées, une adaptation de nos fonctions d'utilité semble nécessaire afin d'obtenir des politiques représentant des optima globaux. Enfin, une comparaison de nos formulations avec leurs homologues dans le formalisme des jeux stochastiques permettra de positionner nos travaux par rapport à ces approches.

Références

[1] Tamer Basar and Geert Jan Olsder. *Dynamic noncooperative game theory*, volume 23. Siam, 1999.

[2] Ronald A Howard. *Dynamic programming and markov processes.* John Wiley, 1960.

[3] Sara Marie MC Carthy. *Hierarchical planning in security games ; A game theoretic approach to strategic, tactical and operational decision making.* PhD thesis, University of Southern California, 2018.

[4] Praveen Paruchuri, Jonathan P Pearce, Janusz Marecki, Milind Tambe, Fernando Ordonez, and Sarit Kraus. Efficient algorithms to solve bayesian stackelberg games for security applications. In *AAAI*, pages 1559–1562, 2008.

[5] James Pita, Manish Jain, Janusz Marecki, Fernando Ordóñez, Christopher Portway, Milind Tambe, Craig Western, Praveen Paruchuri, and Sarit Kraus. Deployed armor protection : the application of a game theoretic model for security at the los angeles international airport. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems : industrial track*, pages 125–132. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[6] Yundi Qian, William B Haskell, Albert Xin Jiang, and Milind Tambe. Online planning for optimal protector strategies in resource conservation games. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 733–740. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[7] Stuart J Russell and Peter Norvig. *Artificial intelligence : a modern approach*. Prentice-Hall, 1995.

[8] Eric Shieh, Bo An, Rong Yang, Milind Tambe, Craig Baldwin, Joseph DiRenzo, Ben Maule, and Garrett Meyer. Protect : A deployed game theoretic system to protect the ports of the united states. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 13–20. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[9] Matthijs TJ Spaan, Frans A Oliehoek, et al. The multiagent decision process toolbox : software for decision-theoretic planning in multiagent systems. In *Proc. of the AAMAS Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains (MSDM)*, pages 107–121, 2008.

[10] Jonathan Taylor, Doina Precup, and Prakash Panagaden. Bounding performance loss in approximate mdp homomorphisms. In *Advances in Neural Information Processing Systems*, pages 1649–1656, 2009.

[11] Heinrich Von Stackelberg. *Marktform und gleichgewicht*. J. springer, 1934.

[12] Yevgeniy Vorobeychik and Satinder Singh. Computing stackelberg equilibria in discounted stochastic games. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[13] Sha Yi, Changjoo Nam, and Katia Sycara. Indoor pursuit-evasion with hybrid hierarchical partially observable markov decision processes for multi-robot systems. In *Distributed Autonomous Robotic Systems*, pages 251–264. Springer, 2019.

[14] Zhengyu Yin, Albert Xin Jiang, Matthew P Johnson, Christopher Kiekintveld, Kevin Leyton-Brown, Tuomas Sandholm, Milind Tambe, and John P Sullivan. Trusts : Scheduling randomized patrols for fare inspection in transit systems. In *Twenty-Fourth IAAI Conference*, 2012.

Apprentissage Automatique pour l'Optimisation Combinatoire : Étude du problème du voyageur de commerce

A. Yaddaden[1], S. Harispe[1], M. Vasquez[1], M. Buljubašić[1]

[1] EuroMov Digital Health in Motion, Univ Montpellier, IMT Mines Alès, Alès, France

prenom.nom@mines-ales.fr

Résumé

Les récentes avancées en apprentissage profond et par renforcement ont ravivé l'intérêt d'étudier la résolution de problèmes d'Optimisation Combinatoire sous l'angle de l'apprentissage automatique : ensemble d'approches qui permettront ici de s'affranchir de la limitante définition manuelle et explicite d'heuristiques spécifiques au problème traité, en visant à apprendre, i.e. découvrir, des procédures de résolution implicites à partir d'analyses d'instances du problème. Cet article propose une revue concise des approches récentes d'apprentissage automatique appliquées aux problèmes du voyageur de commerce (TSP).

Mots-clés

Optimisation combinatoire, problème du voyageur de commerce, TSP, apprentissage automatique, apprentissage profond, apprentissage par renforcement.

Abstract

Recent advances in deep and reinforcement learning have rekindled the interest of studying the resolution of Combinatorial Optimization problems using machine learning : a set of approaches allowing to overcome the limiting manual definition of explicit heuristics specific to the problem being addressed, aiming to learn, i.e to discover, implicit resolution procedures from instances analysis of the problem. This article offers a concise review of recent machine learning approaches applied to the traveling salesman problem (TSP).

Keywords

Combinatorial optimization, traveling salesman problem, TSP, machine learning, deep learning, reinforcement learning.

1 Introduction

Les développements récents en Intelligence Artificielle, et plus particulièrement en apprentissage automatique, sont à l'origine de systèmes très compétitifs pour la résolution de tâches difficiles dans des domaines aussi variés que le traitement d'images, le traitement du langage naturel, la résolution de jeux complexes, ou encore plus récemment, la résolution de problèmes impliquant du raisonnement symbolique [13, 2, 27, 14]. Une des caractéristiques particulièrement intéressante de ces systèmes est qu'ils sont capables, sans programmation explicite d'une méthode de résolution, de réaliser efficacement des tâches complexes ; ils permettent alors d'aborder des tâches difficiles ou impossibles à traiter efficacement de manière explicite (e.g. traduction automatique, classification d'images), mais aussi de surpasser dans certains cas des algorithmes spécialisés, fruits de savoir-faire issus de dizaines voire centaines d'années de recherche, e.g. résolution d'équations différentielles complexes [14].

Dans ce contexte, les récents développements en apprentissage automatique apparaissent particulièrement intéressants pour l'étude de problèmes d'optimisation combinatoire difficiles à résoudre – traditionnellement abordés par la définition d'heuristiques complexes et très spécifiques du problème traité. Des revues générales sur les méthodes d'apprentissage machine appliquées aux problèmes d'optimisation combinatoire sont proposées dans la littérature [5, 8]. Ce papier propose une revue dédiée aux développements récents en apprentissage automatique pour la résolution des problèmes du voyageur de commerce. Ces problèmes revêtent un intérêt particulier pour la communauté de la Recherche Opérationnelle. Ils sont en effet étroitement liés à des problèmes d'optimisation combinatoire NP-Difficiles (e.g. VRP, problèmes de tournées de véhicules) très étudiés au vu de leurs nombreuses applications pratiques dans le transport, la télécommunication, et dans la formulation de problèmes variés (e.g. en protéomique).

Le problème du voyageur du commerce – *Traveling Salesman Problem* ou TSP – vise à distinguer le trajet le plus court passant par un ensemble de villes. Pléthore de variantes ont été définies pour tenir compte de caractéristiques spécifiques d'importance, e.g. nombre de voyageurs (mTSP), fenêtres de temps (TSPTW), capacité du voyageur (CVRP [1]). Le TSP est très étudié dans la littérature, et de nombreuses méthodes exactes et approchées ont été proposées pour ses différentes variantes. Les méthodes exactes sont généralement très coûteuses en temps de calcul, e.g. *branch-and-bound* et ses variantes, et seulement applicables en pratique à des instances de petite taille (peu de villes). Plusieurs approches heuristiques visent à

1. CVRP : *Capacited Vehicle Routing Problem*, généralise le TSP en construisant des tournées avec des véhicules (voyageurs) ayant une capacité précise qui doivent desservir plusieurs clients (villes).

trouver des solutions de bonnes qualités en un temps de calcul raisonnable : les méthodes de recherche locale, le recuit simulé, des algorithmes génétiques, etc. Bien que ces méthodes soient efficaces, elles nécessitent un effort de conception important pour tenir compte de toutes les contraintes du problème. Généralement, une légère modification des contraintes du problème entraîne des adaptations majeures des heuristiques développées. Nous proposons par la suite une discussion de la littérature visant à s'affranchir de la définition manuelle de ces heuristiques par l'utilisation de techniques d'apprentissage automatique. Notre étude portera sur les méthodes contemporaines à base d'apprentissage profond proposées pour la résolution du TSP.
Le papier est structuré comme suit : la section 2 présente les différentes notions d'apprentissage automatique importantes dans la littérature étudiée ; la section 3 discutera plus en détail les approches utilisées, et les performances obtenues dans le cas du TSP. Nous proposerons par la suite, en section 4, une discussion sur les différentes approches ainsi que plusieurs perspectives de recherche dans le domaine.

2 Modélisation du TSP et pré-requis

Cette section présente succinctement i) la modélisation du TSP classiquement adoptée dans la littérature traitée, et ii) les concepts importants de l'apprentissage automatique qui seront par la suite utilisés pour caractériser les approches que nous étudierons. Nous proposons des pointeurs vers la littérature technique propre à chacune des notions d'apprentissage automatique mentionnée.

2.1 Modélisation du TSP

Une solution du TSP peut être exprimée comme un ordre de traitement de l'ensemble des nœuds d'un graphe, ou plus généralement une séquence de nœuds (chemin). Les nœuds du graphe correspondent aux villes à visiter et la séquence traduit donc l'ordre de visite. Formellement, pour un problème de TSP académique considérant un graphe complet de n villes $X = \{x_1, ..., x_n\}$, tq. $x_i \in \mathbb{R}^2 \quad \forall i \in \{1, ..., n\}$, avec une distance euclidienne entre les villes, nous cherchons une permutation de $\{1, 2, \ldots, n\}$, $\phi^* = \{\phi^*(1), \ldots, \phi^*(n)\}$ correspondant à un tour de longueur minimale ; avec
$L(\phi) = ||x_{\phi(n)} - x_{\phi(1)}||_2 + \sum_{i=1}^{n-1} ||x_{\phi(i)} - x_{\phi(i+1)}||_2$ la longueur du tour ϕ. La problématique d'apprentissage automatique correspond donc à identifier une fonction qui permettra tant que possible de distinguer le tour de coût minimal pour une instance donnée du problème caractérisée par la liste des villes à visiter et leurs distances deux à deux. Notons que dans la littérature, les variantes du TSP traitées sont de nature académique, i.e. caractérisées par les simples coordonnées des localisations, en considérant un graphe complet et une norme (e.g. L^1, L^2) pour calculer les distances entres nœuds ; des informations supplémentaires, e.g. nombre de voyageurs, fenêtres de temps, capacités, demandes sont définies en fonction de la variante (mTSP, TSPTW, CVRP...).
Deux types d'approches impliquant de l'apprentissage automatique peuvent être distingués dans la littérature en fonction du paradigme considéré pour estimer la fonction précitée ; celui dit *supervisé* ou celui dit *par renforcement*.
Les approches supervisées reposent sur l'analyse d'une base d'instances résolues, généralement obtenues par l'utilisation de méthodes exactes ou heuristiques – on cherche dans ce cas à prédire la solution attendue, i.e. la séquence de localisations connue. La littérature s'intéresse à différentes approches capables de prédire des séquences ; elle se concentre dernièrement très largement sur des approches à base de réseaux de neurones dits profonds (e.g., modèles de type *Seq2Seq* détaillés par la suite).
Les approches par renforcement reposent sur l'analyse de collections d'instances non résolues ; on cherche dans ce cas de manière itérative à définir une approche qui permet d'optimiser une fonction objectif qui traduit la qualité des solutions produites. Les approches par renforcement abordent le TSP sous la forme d'un problème de prise de décision séquentielle dans lequel on vise à trouver une fonction (appelée *politique*) à même de sélectionner la prochaine ville à visiter à chaque étape, et cela bien entendu de manière à minimiser le coût global du tour. Différentes approches peuvent être considérées pour optimiser la politique - les plus populaires seront mentionnées par la suite. Elles utilisent elles-aussi très largement des estimateurs de fonction pour mesurer certaines quantités d'importance pour optimiser une politique ; l'état de l'art plébiscite une fois de plus très largement les approches à base d'apprentissage profond pour effectuer ces approximations.

2.2 Concepts d'Apprentissage Automatique

2.2.1 Pré-requis sur l'apprentissage profond

Les modèles *séquence à séquence* (*Seq2Seq*) [22] permettent de traiter les problèmes qui peuvent être modélisés comme des problèmes consistant à prédire une séquence à partir d'une séquence donnée ; ils sont par exemple utilisés en traduction automatique, mais aussi, pour modéliser le traitement de problèmes d'optimisation combinatoire tels que le TSP. Parmi les modèles *Seq2Seq*, les approches de type *encoder-decoder* sont les plus fréquentes ; elles considèrent la prédiction en deux phases : i) encodage de l'information portée par la séquence fournie en entrée, ii) décodage de l'information encodée pour produire la nouvelle séquence. Une approche impliquant deux réseaux récurrents (RNN) de type LSTM [2] peut être utilisée. Ces deux réseaux vont respectivement i) encoder la séquence d'entrée de manière itérative, symbole par symbole, en une représentation de taille fixe, pour par la suite ii) décoder la représentation de la séquence encodée de manière itérative en une nouvelle séquence de symboles.
Des *mécanismes d'attention* peuvent être utilisés de manière à enrichir les étapes d'encodage et de décodage [2]. Il est par exemple possible, lors du décodage, de tirer parti de l'information traitée lors de l'encodage itératif de la séquence encodée – le modèle porte alors en quelque sorte attention à certaines informations locales utiles pour le dé-

2. LSTM : *Long Short Term Memory*, type de réseaux de neurones récurrents qui peuvent apprendre des dépendances à long terme.

codage.

Pointer Network (Ptr-Net) [28] : architecture spécifique d'un modèle *Seq2Seq* [2], qui a l'avantage de pouvoir traiter un nombre variable d'entrées et de produire un nombre variable de sorties. Il s'agit de deux LSTM : l'un pour l'encodage, l'autre pour le décodage avec un mécanisme d'attention permettant de pointer sur les symboles encodés. Ce mécanisme d'attention prend en entrée les vecteurs cachés engendrés par l'encodeur à partir des données en entrée ainsi que le vecteur caché engendré par le décodeur pour produire un vecteur d'attention qui servira à donner les probabilités de pointer sur les entrées lors de la génération de la solution.

Transformer [25] : approche permettant de faire du *Seq2Seq* sans récurrence, en se basant exclusivement sur un modèle d'attention (des approches *multi-head* permettent de démultiplier le mécanisme d'attention).

Graph Neural Network (GNN) [31] : modèle permettant d'encoder l'information portée par un graphe ; ces modèles sont utilisés pour encoder l'information caractérisant l'instance à traiter, e.g. modéliser les dépendances entre les nœuds grâce aux plongements de graphes.

2.2.2 Pré-requis sur l'apprentissage par renforcement

Dans le contexte de l'étude du TSP, les approches à base d'apprentissage par renforcement visent à distinguer une politique (fonction) qui permettra de traiter itérativement une instance du problème, i.e. en construisant la séquence des nœuds de la tournée de manière itérative, nœud après nœud, et cela de manière à minimiser la longueur du tour complet résultant de l'application itérative de la politique. Le problème est très généralement modélisé sous la forme d'un processus de décision markovien (MDP) – une introduction aux MDP et à l'apprentissage par renforcement est fournie par Sutton et Barto [23]. L'utilisation de ce modèle considère la plupart du temps un état comme l'ensemble des informations caractéristiques du traitement de l'instance du problème à un temps donné (villes visitées, capacités...), une action applicable dans un état comme une des villes qui peut être visitée, et la notion de récompense, relative au choix d'une action dans un état, comme l'inverse de la distance ajoutée au tour lorsqu'une ville est ajoutée au tour partiel – les transitions entre états au regard des actions choisies et les récompenses sont déterministes. Une procédure basée sur l'analyse de cette politique permettra par la suite de construire une solution pour une instance donnée (e.g., algorithme glouton ou de type *beam search*).

Différentes approches proposées dans la littérature permettent de chercher une politique de qualité (méthodes de résolution de MDP avec espace d'action discret). Deux approches populaires sont citées ci-après : i) *Q-learning* et ses variantes (e.g., *Double*, *Deep*, *Dueling*) aborde indirectement la recherche de la politique en affinant itérativement une estimation de la qualité associée à une action à effectuer dans un état donné (à l'image des techniques à base de programmation dynamique) ; ii) les approches de type *policy gradient* (e.g REINFORCE [29] et Actor Critic [18]) visent directement à optimiser les paramètres d'une politique paramétrique de manière à minimiser le coût des solutions produites. En pratique ces deux approches reposent la plupart du temps sur l'utilisation d'estimateurs de fonction de type réseaux de neurones profonds (cf. *Deep Reinforcement Learning*).

3 Méthodes de l'état de l'art

Nous synthétisons dans cette section la littérature qui traite de la résolution du TSP à l'aide de méthodes à base d'apprentissage automatique. La présentation proposée est structurée de manière à souligner leurs similitudes et différences. Nous distinguons notamment le paradigme considéré (à base d'apprentissage supervisé ou par renforcement), le type des plongements considérés (plongements de graphe *vs.* plongements des nœuds), et le type d'approche utilisé (construction *vs.* amélioration de la solution). Nous préciserons aussi la taille des instances d'entraînement et de test considérées. A noter que les métriques les plus pertinentes pour l'évaluation d'une méthode sont l'écart à l'optimum des solutions données à l'inférence ainsi que le temps de résolution et la capacité de généralisation à de plus grandes instances que celles utilisées pour l'entraînement.

3.1 Méthodes à base d'apprentissage supervisé

Ptr-Net est le premier modèle à base d'apprentissage profond qui propose de traiter de bout en bout la résolution du TSP [28]. Ce modèle de type *Seq2Seq* est entraîné de manière supervisée avec des solutions obtenues par des heuristiques (algorithme Glouton, algorithme de Christofides suivi de 2-opt). L'entraînement est effectué en maximisant la vraisemblance des tours attendus (labels) présents dans une base d'instances résolues. Notez que la paramétrisation du réseau de neurones qui sera utilisé pour inférer les tours en pratique ne repose pas sur une évaluation explicite de la longueur des tours de la base labélisée. Le modèle *Seq2Seq* paramétrisé permet d'évaluer la distribution de probabilité des villes pour chaque étape du tour. L'inférence est finalement faite avec une procédure de *beam search* basée sur l'analyse des distributions de probabilités. L'approche est compétitive par rapport aux heuristiques précitées sur des instances de 5, 10, 20 et 50 villes. Elle a aussi l'avantage de ne pas être contrainte au traitement d'instances de même taille que celles considérées lors de l'entraînement (test sur des instances de taille 50 pour un modèle entraîné avec des instances de taille 5 à 20).

Une amélioration de Ptr-Net, appelée *Objective-based learning*, modifie la procédure d'optimisation paramétrique du modèle afin de prendre en compte la longueur des tours produits à partir du modèle paramétré [17]. Ainsi, les paramètres du modèle sont mis à jour seulement si la solution prédite est de moins bonne qualité que la solution attendue. Les tests sur des problèmes à 20 villes ont montré une amélioration par rapport au modèle Ptr-Net standard.

Une architecture a aussi été étudiée dans le cas spécifique du mTSP [10]. Celle-ci se base sur le modèle PointNet [21]

qui permet d'assurer une invariance à l'ordre des éléments qui composent l'entrée du modèle - les modèles *Seq2Seq* sont par nature définis pour tenir compte de la structure de la séquence d'entrée. L'entrée du modèle considère non plus une séquence mais 3 groupes d'éléments : le dépôt (singleton), les villes et les voyageurs. L'invariance à la permutation sur les éléments est gérée par un mécanisme de *pooling* classiquement utilisé dans les réseaux convolutifs. Pour chaque groupe, des vecteurs contextes sont calculés avec l'opération de *pooling* de type *Leave-One-Out*. Le groupe des villes est augmenté de l'information sur la distance entre chacune d'elles avant le calcul des vecteurs contextes. Les éléments de chaque groupe et les vecteurs contextes sont passés en entrée d'une fonction de projection affine. Comme dans *Transformer*, la couche invariante à la permutation est suivie de couches entièrement connectées avec, entre chaque couche, une opération de normalisation. L'entraînement se fait de manière supervisée, les labels étant ici obtenus par la résolution de la formulation en programme en nombre entier du problème. A l'inférence, la méthode *beam search* est utilisée pour produire des tours valides. Les instances d'entraînement et de test sont de taille 5, 10 et 20 villes avec un nombre de clients allant de 1 et 5. Les résultats montrent une nette dominance par rapport aux heuristiques classiques, OR-tools et d'autres méthodes à base d'apprentissage profond [28, 4, 12]. Les auteurs attestent aussi de la généralisation de leur modèle en termes de taille des instances et de distribution de génération des instances. La méthode s'avère aussi dans ce cas très souvent compétitive avec OR-tools.

3.2 Méthodes à base d'apprentissage par renforcement

Le choix de l'apprentissage par renforcement est motivé par le fait que toute approche supervisée nécessite la constitution coûteuse en temps de calcul d'une base d'instances résolues ; ce qui est limitant dans le sens où la résolution d'instances complexes ne peut être qu'approchée et traitée avec des heuristiques. Cela limitera de fait la découverte de solutions plus compétitives puisque le modèle visera alors à *"imiter"* une heuristique la plupart du temps, sans pouvoir *in fine* trouver de meilleures solutions. [3] Pour rappel, le TSP est exprimé comme un problème d'apprentissage par renforcement en considérant un état comme étant la solution partielle donnée itérativement par la sélection de la prochaine ville à visiter ; une action consiste alors à choisir cette ville en minimisant la longueur du tour construit. L'idée de Ptr-Net [28] a été adaptée à un contexte d'apprentissage par renforcement [4]. La politique est donnée par un mécanisme de *Pointer Network* similaire à Ptr-Net, augmenté d'un mécanisme de masque afin d'exclure les villes desservies. L'algorithme d'entraînement est de type Actor Critic. Le réseau de neurones *actor* joue le rôle d'une politique paramétrique π_θ, tandis que le réseau de neurones *critic* est utilisé pour estimer la longueur du tour. Cette dernière estimation permettra de mettre à jour les paramètres de π_θ en servant de *baseline* dans le but de réduire la variance lors de l'entraînement. Différentes stratégies d'inférence sont proposées. La stratégie d'échantillonage (*sampling*) consiste à calculer plusieurs solutions candidates avec π_θ pour choisir celle minimisant la distance. La stratégie *active search* vise à mettre à jour π_θ en minimisant la moyenne des longueurs des tours pour chaque instance évaluée. Les résultats montrent que la politique apprise avec ce type d'approche donne des résultats compétitifs avec des métaheuristiques raffinées telles que le recuit simulé.

Des approches proposent aussi de se passer des cellules récurrentes (LSTM) utilisées dans Ptr-Net. Par exemple, Deudon et al. [7] utilisent un encodeur similaire à celui de Transformer pour obtenir un ensemble de vecteurs d'actions (un vecteur par ville) représentant l'interaction d'une ville avec les autres. Le décodeur est une somme des transformations linéaires des trois derniers vecteurs d'actions choisis dans la solution partielle du TSP suivie d'une fonction d'activation ReLu. D'après leurs tests, les performances d'un décodeur exploitant des cellules récurrentes sont égalées avec ce type d'approche. Le vecteur généré par le décodeur ainsi que les vecteurs d'actions sont traités par un mécanisme d'attention (cf. [4]) pour générer les probabilités de sélection de la prochaine ville. L'entraînement est traité par une approche de type REINFORCE. Les solutions générées durant la phase de test sont cependant améliorées avec une procédure de recherche locale de type 2-opt. Les instances d'entraînement et de test sont de taille 20, 50 et 100 villes. Les tests suggèrent une meilleure performance que Ptr-Net [28] et Pointer Network [4] quand la solution trouvée est améliorée avec 2-opt mais moins bonne sans. A noter que les résultats des instances à 100 villes sont donnés avec un modèle entraîné avec des instances à 50 villes pour tester la généralisation sur des instances de plus grande taille. Le modèle proposé sans 2-opt peine cependant à généraliser par rapport au modèle de type Pointer Network [4].

De manière alternative aux approches précitées, des travaux proposent de considérer l'utilisation d'une matrice de transition stockant la probabilité de passer d'une ville à l'autre [16] - et non pas de prédire la probabilité de chaque ville à chaque étape. L'action à réaliser dans un état vise à sélectionner à partir de cette matrice la prochaine ville à visiter. Une approche de type Actor Critic [16] considère par exemple que le réseau de neurones *actor* modifie la matrice de transition en générant un vecteur de mise à jour. Le réseau de neurones *critic* estime le coût d'une solution de l'instance. Contrairement aux autres architectures, les entrées du réseau *actor* sont les coordonnées des villes sans plongement préalable dans un espace latent de dimension supérieure. Ce choix est justifié par le fait que cela permet d'obtenir une architecture du réseau de neurones relativement plus petite que celle utilisant des plongements. Dans la phase d'entraînement, une procédure d'échantillonnage tire plusieurs tours pour une instance donnée et garde le tour de longueur minimale pour faire les mises à jour des

3. Notez qu'une adaptation précitée vise à s'affranchir de cette limitation en évaluant la longueur des tours générés [17]. La constitution de la base annotée reste cependant un problème important.

réseaux de neurones *actor* et *critic*. Les tests sur des instances à 10, 20 et 50 villes montrent que la méthode est compétitive avec le modèle Pointer Network [4] ; elle présente néanmoins le désavantage de ne pas être invariante à la taille de l'instance puisque la matrice de transition et le réseau *actor* (i.e. la politique) dépendent de celle-ci.

3.3 Méthodes à base de GNN

Des méthodes à base de *Graph Neural Network* (GNN) proposent d'approcher les problèmes d'optimisation combinatoire en exploitant explicitement leur formulation sous la forme de graphe ; dans le TSP, les nœuds représentent les villes, les arcs les distances les reliant, et une solution (tour) est un chemin. Ces méthodes visent à encoder des informations du graphe.

La structure de graphe est par exemple exploitée pour calculer pour chaque ville son plongement grâce à l'approche *struct2vec* [6] en considérant le sous-graphe connectant chaque nœud avec ses k plus proches voisins [11]. La sélection de la prochaine ville à visiter se fait avec un algorithme de type *Deep* Q-learning [19] avec une politique ϵ-greedy, i.e. en considérant une ville aléatoirement parmi les non-visitées avec une probabilité ϵ et la ville qui maximise la fonction d'évaluation Q avec une probabilité $1 - \epsilon$. La fonction Q a pour rôle d'estimer le coût de l'insertion d'une ville (représentée par son plongement) dans le tour partiel (représenté lui aussi par un plongement de graphe). Cette approche a permis de traiter des instances de plus grandes tailles (200-300) que les approches précédentes (50-100). L'algorithme donne de bons résultats pour des instances à 1000-1200 nœuds alors qu'il a été entraîné sur des instances de 50-100 nœuds. Notons tout de même que l'évaluation proposée dans cette étude considère une restriction de temps d'exécution qui semble pénalisante pour le solveur Concorde [1] utilisé dans la comparaison.

Une architecture encodeur-décodeur à base de *Graph Attention Network* [26] considérant le graphe complet pour le TSP propose un modèle entièrement fondé sur le mécanisme d'attention pour le calcul des probabilités de sélection des villes [12]. L'encodeur est similaire à celui de l'architecture Transformer [25] et donne en sortie les plongements des nœuds villes et le plongements du graphe comme la moyenne des plongements des nœuds. Le décodeur construit la solution de manière séquentielle. Il prend en entrée les plongements des nœuds et un nœud contexte calculé à partir de la concaténation du plongement du graphe, du dernier nœud ajouté au tour, et du premier nœud dans le tour. Il est composé d'un *multi-head attention* (MHA) suivi d'un *single-head attention* (SHA) [25] qui donne les probabilités d'ajout d'un noeud dans le tour partiel. La politique de résolution du problème est apprise grâce à un algorithme REINFORCE. La *baseline* utilisée est une copie de la meilleure politique apprise employée de manière gloutonne. Tout comme le réseau cible dans le *Deep* Q-learning, cette politique est figée et ne change que si l'amélioration entre la politique apprise actuellement et la politique de la *baseline* est significative. Le modèle est entraîné sur des instances de 20, 50 et 100 villes. A l'inférence, les stratégies gloutonne et d'échantillonnage sont considérées. Les meilleurs résultats sont obtenus avec échantillonnage. L'évaluation proposée montre que cette approche surpasse des heuristiques classiques du TSP, ainsi que d'autres méthodes à base d'apprentissage profond dans le contexte testé ; cela en assurant un delta à l'optimal au pire des cas inférieur à 5%. De plus, la méthode rivalise avec les solveurs Gurobi et Concorde.

Des travaux proposent une méthode supervisée à base de *graph convolutional network* et de *beam search* [9, 20]. Les labels des instances d'entraînement et de test proviennent d'une résolution à l'optimal avec le solveur Concorde. Les instances considérées sont de taille 20, 50 et 100 villes permettant une résolution à l'optimal. Contrairement aux autres approches qui ne considèrent que les coordonnées géographiques d'une ville, les auteurs augmentent celles-ci de la matrice des distances entres les villes et de la matrice d'adjacence qui associe pour chaque ville ses k-plus proches voisines. Deux sortes de plongements sont alors calculés : ceux des coordonnées et ceux des distances concaténés aux plongements de la matrice d'adjacence. Ces plongements sont par la suite traités par des couches de convolution. Enfin, la probabilité qu'un arc figure dans une solution est obtenue avec un perceptron multi-couches qui donnera en sortie une matrice d'adjacence des arcs de la solution. Plusieurs configurations ont été testées pour la génération de la solution finale : la génération gloutonne, *beam search* sur les probabilités d'occurrence des arcs et *beam search* sur la longueur des arcs. Cette dernière semble donner les meilleurs résultats avec un delta à l'optimal de 0.01%, 0.01% et 1.39% pour les instances de 20, 50 et 100 villes respectivement, ce qui est aussi meilleur que les résultats d'autres méthodes basées sur l'apprentissage profond [28, 4, 7, 12]. Le modèle semble cependant ne pas bien généraliser à des instances plus grandes que celles de l'entraînement. Les auteurs expliquent cela par le fait que les caractéristiques apprises par celui-ci semblent dépendre de la taille du graphe.

Le concept de *Graph Pointer Networks* (GPN) [15], basé sur Pointer Network [4], propose de coupler plusieurs idées précitées. L'encodeur est ici composé de deux phases : une première dans laquelle les coordonnées sont plongées dans un espace à $d-$dimensions, suivie d'une seconde encodant le graphe basée sur un GNN. L'entrée du GNN est un vecteur contexte défini comme étant un vecteur de pointeurs d'une ville vers les autres. Les plongements des entrées sont traités par un LSTM pour produire un vecteur caché qui sera utilisé dans la phase de décodage et aussi ré-injecté dans le LSTM pour le calcul du vecteur caché de la prochaine ville. Le vecteur en sortie du GNN et le vecteur caché en sortie du LSTM sont utilisés dans un décodeur similaire au modèle Pointer Network [4] pour obtenir la probabilité de sélection de la prochaine ville à inclure dans la tournée. Le modèle est entraîné avec l'algorithme REINFORCE. L'entraînement et le test sont réalisés sur des instances allant de 20 et 50 villes. Les tests montrent que la méthode donne de meilleurs résultats que 2-opt et l'algorithme de Christofides et qu'elle est compéti-

tive avec plusieurs modèles de l'état de l'art [4, 11, 12]. De plus, la généralisation à des instances de 250 à 1000 villes a été testée avec un modèle entraîné sur des instances de 50 villes. Ce modèle semble être plus compétitif que les autres modèles à base d'apprentissage et donner des solutions d'assez bonne qualité si l'on considère le compromis entre temps d'exécution et delta à l'optimal. Les auteurs expliquent ces résultats par l'introduction du vecteur contexte et du GNN et par un entraînement sur 10 *epochs* seulement (*early stopping*) qui évite le sur-apprentissage sur les petites instances.

Un algorithme d'apprentissage par renforcement hiérarchique à deux niveaux a aussi été étudié pour le TSP avec fenêtre de temps (TSPTW) [15]. L'apprentissage par renforcement hiérarchique permet de définir t niveaux avec t politiques à apprendre et donc t fonctions de récompense. Chaque niveau correspond à une tâche différente à apprendre. Dans le premier niveau, une pénalité est associée au cas où l'arrivée dans une ville se fait après le temps de départ de celle-ci. Dans le second, la fonction de récompense est le coût total du tour défini par une combinaison linéaire du temps total de trajet et de la fonction de pénalité du premier niveau. GPN est adapté au cas hiérarchique (HGPN) et la distribution finale pour l'inclusion d'une ville dans le tour est donnée par un softmax d'une combinaison linéaire entre les vecteurs des mécanismes d'attentions des deux niveaux précités. La comparaison avec des instances de 20 villes entre HGPN, OR-Tools et un algorithme de colonie de fourmis (ACO) sur le pourcentage de faisabilité des solutions obtenues montrent qu'HGPN donne, dans presque tous les cas, des solutions faisables alors qu'OR-Tools et ACO ont un pourcentage de faisabilité plus faible. Notons que le delta à l'optimal d'HGPN est de 3.45% ce qui est un excellent score.

3.4 Les méthodes d'amélioration

Une autre classe de méthodes de résolution porte sur l'amélioration d'une solution faisable. Une approche consiste à réaliser ce traitement à l'aide d'un modèle à base d'apprentissage par renforcement [30]. L'heuristique se base sur la sélection d'une paire de nœuds pour pouvoir effectuer une opération de recherche locale. Deux politiques de recherche locale à base de paire de nœuds ont été implémentées, 2-opt et la permutation de nœuds (*swap*). La politique de sélection d'une paire est apprise grâce à un réseau de neurones basé sur l'architecture Transformer. Le calcul des plongements des entrées comprend également l'opération d'encodage de position contrairement aux autres travaux présentés précédemment et le MHA est remplacé par un SHA. Une opération de *max-pooling* permet ensuite d'obtenir le plongement de graphe. Les plongements des entrées et le plongements de graphe sont passés dans un module qui somme leurs transformations linéaires pour inclure l'information globale du graphe dans chaque nœud. La compatibilité entre les nœuds est ensuite évaluée grâce à un produit matriciel similaire au *dot-product* dans un SHA suivi d'une fonction tanh. Enfin, la fonction softmax est appliquée à la matrice résultante pour donner les probabilités de choisir une paire de nœuds pour l'opération de recherche locale. L'entraînement se fait avec l'algorithme *n-step* Actor Critic [18]. L'entraînement et le test ont été conduits sur des instances à 20, 50 et 100 clients. Les tests entre les deux politiques apprises montrent une meilleure amélioration de la solution pour la politique implémentant 2-opt. L'algorithme donne de bonnes solutions même lorsque la solution initiale est de mauvaise qualité. Les solutions obtenues ont un delta à l'optimal de 1.42% pour des instances à 100 clients.

4 Discussion et perspectives

L'état de l'art propose de nombreuses approches à base d'apprentissage automatique permettant de traiter diverses variantes du TSP, avec des premiers résultats intéressants sur des problèmes de taille réduite. Cela souligne l'intérêt d'étudier ce cadre théorique pour aborder ce type de problème d'optimisation combinatoire.

On remarque une nette dominance du paradigme de l'apprentissage par renforcement. Celui-ci permet de s'affranchir du besoin de constituer une base de données d'instances résolues. Cette dernière est difficile à construire compte tenu de la difficulté d'accès aux labels optimaux, principalement quand la taille du problème augmente. Notons aussi que les méthodes *policy gradient* (principalement REINFORCE) dominent les méthodes dites *value based* (e.g. *Deep* Q-learning). Bien que les méthodes *policy gradient* nécessitent plus d'exemples d'entraînement, elles ont l'avantage d'apprendre la politique de résolution directement et d'épouser le cadre probabiliste considéré dans les modèles *Seq2seq* (la probabilité d'une permutation est définie comme un produit de probabilités conditionnelles).

Une autre préférence porte sur le schéma global de résolution ; tous les modèles considèrent une phase d'encodage des données en entrée suivie d'un décodage qui produira la solution.

Bien que les premiers modèles étaient composés de RNN, le mécanisme d'attention et principalement les modèles s'inspirant de Transformer sont aujourd'hui préférés. Cela souligne la dépendance des performances des modèles à la qualité des plongements des (informations des) instances. Une première piste d'amélioration des approches porterait sur une meilleure représentation des données en entrée avec des plongements en adéquation avec les spécificités du problème traité. Une attention particulière pourrait être accordée aux approches tirant profit de la représentation sous forme de graphe. Des études visant à s'affranchir de certaines considérations aujourd'hui communes dans ces travaux, e.g. graphes complets, prise en compte de coordonnées géographiques sans matrice de distance (norme L^1 ou L^2) seraient par ailleurs souhaitables.

Nous remarquons globalement que les résultats des méthodes à base d'apprentissage automatique sont encore aujourd'hui inférieurs à ceux obtenus par des méthodes de Recherche Opérationnelle (RO). Les méthodes de RO permettent de traiter des instances plus grandes, avec parfois des garanties d'optimalité sur les solutions trouvées.

Notre étude montre que les méthodes à base d'apprentissage automatique entraînées sur de petites instances offrent une bonne généralisation sur des instances de tailles similaires. Néanmoins, des difficultés importantes sont identifiées pour i) l'entraînement sur des instances comportant plus de villes, ii) généraliser à des instances de grande taille suite à l'entraînement sur des instances de petite taille. De plus, la généralisation à des instances issues de distributions différentes reste à évaluer - les paradigmes d'apprentissage automatique considèrent très souvent des instances indépendantes et identiquement distribuées. Un challenge majeur porte ainsi sur la définition de modèles adaptés au traitement de plus grandes instances et capables d'exprimer de fortes capacités de généralisation. On peut conjecturer que la difficulté de généralisation est due au fait que la politique obtenue par la phase d'apprentissage semble, dans son comportement, fortement dépendante de la taille de l'instance [9] ; ceci laisse présupposer que des invariants importants pour la résolution du problème ne semblent pas avoir été distingués par les architectures et modèles testés. Comme l'ont montré certains travaux, une coopération entre les algorithmes issus de la RO et de l'apprentissage automatique peut s'avérer plus fructueuse que l'utilisation d'un modèle d'apprentissage automatique de bout en bout. On peut donc distinguer les approches constructives qui favorisent un apprentissage de bout en bout et les approches d'amélioration qui suggèrent d'employer le modèle dans un schéma de résolution plus global où l'algorithme d'apprentissage aide la prise de décision ; e.g. la sélection de la région sur laquelle effectuer une recherche locale. Une piste de recherche pourrait considérer des schémas d'hybridation plus sophistiqués à la manière dont ceux-ci sont pensés pour les métaheuristiques [24].

Parmi les nombreux algorithmes d'inférence développés, le plus élémentaire exploitant un décodage glouton ne semble pas donner les meilleurs résultats. Les plus efficaces sont aujourd'hui *beam search* et l'échantillonnage qui permettent une meilleure exploration de l'espace des solutions. D'autres pistes pour les algorithmes d'inférences ont été imaginées, par exemple *Active Search* [4] qui continue d'optimiser la politique du modèle en phase d'inférence. D'autres algorithmes d'inférence méritent d'être étudiés, et leurs liens avec le modèle, prédisant très souvent les distributions de probabilités, mieux compris.

En ce qui concerne les *benchmarks* utilisés, la majorité sont générés aléatoirement selon une distribution uniforme, bien que certains algorithmes ont pu être testés sur des instances de la littérature, parfois réelles. L'avantage des instances de la littérature repose sur le fait que leur génération suit un protocole particulier qui leur garantit certaines propriétés rendant le test des algorithmes plus légitime. Cependant, force est de constater que l'entraînement des algorithmes d'apprentissage automatique nécessite une quantité d'instances supérieure à celle habituellement fournie dans les benchmarks du TSP. Par exemple, dans TSPLib on retrouve une centaine d'instances pour le TSP symétrique alors qu'un modèle tel que Ptr-Net [28] a nécessité un million d'instances d'entraînement. Des pistes de recherche peuvent être envisagées : i) créer plus d'instances ressemblant à celles de l'état de l'art actuel du TSP (e.g. [3]), ii) chercher des algorithmes capables d'apprendre une méthode efficace de résolution avec peu d'instances. La deuxième alternative est particulièrement attrayante et peut s'avérer très utile pour des problèmes du monde réel pour lesquels peu d'instances existent. Cela est néanmoins un véritable challenge ; pour rappel les méthodes de type *policy gradient* aujourd'hui extrêmement plébiscitées dans le domaine restent aussi réputées pour leur voracité en termes d'instances d'entraînement.

5 Conclusion

L'état de l'art proposé présente les principales approches récentes à base d'apprentissage automatique appliquées à la résolution du TSP ; un problème de choix compte tenu des apports théoriques et pratiques que son étude peut offrir au domaine de l'optimisation combinatoire. Il constitue par ailleurs un premier pas vers la résolution de problèmes plus complexes encore, ayant un fort intérêt pratique, tels que les problèmes de tournées.

Même si les approches présentées restent encore aujourd'hui i) exploratoires pour le traitement de problématiques d'optimisation combinatoire, ii) souvent peu comprises d'un point de vue théorique, et iii) moins performantes que les approches état de l'art de la RO, notamment pour la résolution d'instances du TSP d'intérêt en pratique, notre étude souligne l'engouement certain dont atteste la richesse des propositions, ainsi qu'une évolution graduelle et stimulante des performances des systèmes proposés. De manière générale, notre étude souligne l'intérêt de catalyser des interactions disciplinaires plus étroites entre la Recherche Opérationnelle et l'Apprentissage Automatique pour le traitement de problématiques d'optimisation combinatoire.

Références

[1] David L Applegate, Robert E Bixby, Vasek Chvatal, and William J Cook. *The traveling salesman problem : a computational study*. Princeton university press, 2006.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv :1409.0473*, 2014.

[3] Pouya Baniasadi, Vladimir Ejov, Michael Haythorpe, and Serguei Rossomakhine. A new benchmark set for traveling salesman problem and hamiltonian cycle problem. *arXiv preprint arXiv :1806.09285*, 2018.

[4] Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv :1611.09940*, 2016.

[5] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimiza-

tion : a methodological tour d'horizon. *arXiv preprint arXiv :1811.06128*, 2018.

[6] Hanjun Dai, Bo Dai, and Le Song. Discriminative embeddings of latent variable models for structured data. In *International conference on machine learning*, pages 2702–2711, 2016.

[7] Michel Deudon, Pierre Cournut, Alexandre Lacoste, Yossiri Adulyasak, and Louis-Martin Rousseau. Learning heuristics for the tsp by policy gradient. In *International conference on the integration of constraint programming, artificial intelligence, and operations research*, pages 170–181. Springer, 2018.

[8] Tiande Guo, Congying Han, Siqi Tang, and Man Ding. Solving combinatorial problems with machine learning methods. In *Nonlinear Combinatorial Optimization*, pages 207–229. Springer, 2019.

[9] Chaitanya K Joshi, Thomas Laurent, and Xavier Bresson. An efficient graph convolutional network technique for the travelling salesman problem. *arXiv preprint arXiv :1906.01227*, 2019.

[10] Yoav Kaempfer and Lior Wolf. Learning the multiple traveling salesmen problem with permutation invariant pooling networks. *arXiv preprint arXiv :1803.09621*, 2018.

[11] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*, pages 6348–6358, 2017.

[12] Wouter Kool, Herke Van Hoof, and Max Welling. Attention, learn to solve routing problems ! *arXiv preprint arXiv :1803.08475*, 2018.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[14] Guillaume Lample and François Charton. Deep learning for symbolic mathematics. *arXiv preprint arXiv :1912.01412*, 2019.

[15] Qiang Ma, Suwen Ge, Danyang He, Darshan Thaker, and Iddo Drori. Combinatorial optimization by graph pointer networks and hierarchical reinforcement learning. *arXiv preprint arXiv :1911.04936*, 2019.

[16] Gorker Alp Malazgirt, Osman S Unsal, and Adrian Cristal Kestelman. Tauriel : Targeting traveling salesman problem with a deep reinforcement learning inspired architecture. *arXiv preprint arXiv :1905.05567*, 2019.

[17] Anton Milan, S Hamid Rezatofighi, Ravi Garg, Anthony Dick, and Ian Reid. Data-driven approximations to np-hard problems. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[18] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[19] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv :1312.5602*, 2013.

[20] Alex Nowak, Soledad Villar, Afonso S Bandeira, and Joan Bruna. A note on learning algorithms for quadratic assignment with graph neural networks. *stat*, 1050 :22, 2017.

[21] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet : Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[22] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[23] Richard S Sutton and Andrew G Barto. *Reinforcement learning : An introduction*. MIT press, 2018.

[24] E-G Talbi. A taxonomy of hybrid metaheuristics. *Journal of heuristics*, 8(5) :541–564, 2002.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[26] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv :1710.10903*, 2017.

[27] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782) :350–354, 2019.

[28] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in neural information processing systems*, pages 2692–2700, 2015.

[29] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4) :229–256, 1992.

[30] Yaoxin Wu, Wen Song, Zhiguang Cao, Jie Zhang, and Andrew Lim. Learning improvement heuristics for solving the travelling salesman problem. *arXiv preprint arXiv :1912.05784*, 2019.

[31] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks : A review of methods and applications. *arXiv preprint arXiv :1812.08434*, 2018.

Vers une utilisation éco responsable des objets connectés par la mutualisation de leurs composants physiques : Une approche basée sur le concept d'artefact

R. Fontaine[1], N. Aky[1], R. Courdier[1] , D. Payet[1]

[1] Université de la Réunion, LIM

Résumé

À l'échelle planétaire, l'empreinte environnementale du numérique représente un continent de 2 à 3 fois la taille de la France et 5 fois le poids du parc automobile français. La cause majeure de cette catastrophe écologique serait la surabondance d'objets connectés, liée notamment à un usage irraisonné amplifié par une logique de surconsommation. Afin de limiter l'impact négatif de cette surabondance et d'optimiser le potentiel informatique émanant de notre environnement, nous proposons donc dans cet article un modèle d'architecture, basé sur le concept d'artefact, visant à favoriser la mutualisation des composants présents au sein de nos appareils connectés. Pour montrer sa faisabilité et les avantages liés à son utilisation, nous proposons une implémentation de ce modèle pour la plateforme Android.

Mots-clés

Green IT Informatique Ubiquitaire Mutualisation Meta-Model Agent & Artefact Système multi-agents en temps réel

Abstract

On a global scale, the environmental footprint of digital technology represents a continent 2 to 3 times the size of France and 5 times the weight of the French car fleet. The major cause of this ecological disaster would be the overabundance of connected objects, linked in particular to an irrational use amplified by a logic of over-consumption. In order to limit the negative impact of this overabundance and to optimize the computing potential emerging from our environment, we propose in this article an architecture model, based on the concept of artifact, in order to promote the mutualization of the components present within our connected devices. To show its feasibility and advantages related to its use, we propose an implementation of this model for the Android platform.

Keywords

Green T Ubiquitous computing Agents & Artifacts meta-model Real-time multi-agents system

1 Introduction

Nous sommes actuellement à l'aube de l'ère de l'informatique ubiquitaire [38]. Cette ère se traduit notamment par une nouvelle tendance à intégrer l'informatique dans les objets physiques de la vie quotidienne grâce aux capacités de calcul, de communication sans fil et d'interaction plus naturelle entre l'homme et l'ordinateur. Cependant, cette avancée technologique apporte avec elle une problématique jusqu'alors passée sous silence qui est celle de l'impact environnemental [7]. En effet, alors que les ordinateurs, imprimantes et autres objets numériques usuels constituaient les principales sources de déchets liées aux nouvelles technologies avant 2015, un basculement s'opère avec principalement une amplification de la présence de 3 types d'appareils [6] :

- Les télévisions représentaient 5 à 15 % des impacts en 2010 contre 9 % à 26 % d'ici 2025 ;
- Les smartphones représentaient 2 % à 6 % des impacts en 2010 contre 4 % à 16 % d'ici 2025 ;
- Les objets connectés représentaient 1 % des impacts en 2020 contre 18 % à 23 % d'ici 2025.

Une des solutions pour limiter cet impact serait de diminuer le nombre d'objets connectés en favorisant leurs mutualisations dans le but de réduire la redondance de leurs composants souvent difficile à recycler et dont les méthodes de production ont une incidence forte sur les populations autochtones. Dans cet article, nous proposons un modèle permettant de mutualiser les éléments de l'environnement intelligent. Chaque objet, simple (capteur, effecteur simple) ou composite (smartphone, télévision, etc.), sera en mesure de partager ses différentes fonctionnalités au sein d'un espace de travail. Dans ce qui suit, nous présenterons tout d'abord une vision globale de la proposition de mutualisation des composants. Cette partie permettra de motiver et expliciter nos objectifs avec un exemple illustratif avant d'entrer plus précisément dans la description et l'analyse de l'état de l'art des infrastructures "IoT" au sens large. Par la suite, nous décrirons le modèle conceptuel proposé pour la mutualisation de nos objets connectés, avant de détailler ses différents avantages conceptuels. L'article se poursuivra par l'implémentation de la solution proposée sur Android et par une discussion sur ses apports. Enfin nous conclurons et exposerons les nouvelles perspectives

liées à ce modèle.

2 Un environnement connecté riche en opportunités

Dans le cadre d'un système ambiant, la principale source d'interaction avec le monde réel est celle produite par les capteurs et les effecteurs disséminés dans l'environnement. La démarche actuelle consiste à ajouter de nouveaux équipements à chaque fois que l'on doit produire un nouveau service. Cependant, notre environnement est déjà empli de

Type de capteurs	Détection
HRM	Rythme cardiaque
Microphone	Son/Volume sonore/souffle
Luxmetre	Mesure de luminosité
Iris/Empreinte	Authentification
Accéléromètre	Mouvement
Capteurs météorologiques	Humidité/température /pression atmosphérique

FIGURE 1 – Différents capteurs présents dans le Samsung galaxy s8

divers capteurs et effecteurs. Pour nous en rendre compte, nous pouvons citer l'exemple des smartphones dont la liste de capteurs embarqués est impressionnante. Dans la Figure 1, nous pouvons voir une liste non exhaustive, des capteurs présents dans un Samsung Galaxy s8, commercialisé en 2017. De la même manière, comme nous pouvons le voir dans la Figure 2 notre environnement fourmille d'éléments capables de transmettre de l'information ou d'agir sur son environnement. Habituellement, nous ne voyons dans ces objets que leurs fonctionnalités initiales imposées par leurs concepteurs. Cependant, même si leurs possibilités peuvent être limitées par les composants électroniques, leurs usages quant à eux peuvent être infinis. Ainsi, nous montrons que chacun de nos objets connectés possède des capacités potentielles et inexploitées à effectuer des tâches génériques que nous nommerons des dispositions. Dans le but de mettre à profit ce potentiel, nous proposons ici de les mutualiser virtuellement afin de dissocier leurs capacités des contextes où elles peuvent être exploitées.

2.1 Exemple illustratif d'une mutualisation des composants : Un agent de surveillance

Pour illustrer nos propos, nous allons nous intéresser à un agent de surveillance A chargé du bien-être d'une personne. Nous nous concentrons sur l'un de ses objectifs, qui est d'alerter en cas de fuite de gaz. Cet agent A a, parmi tous ses objectifs, l'objectif Ω de prévenir la personne en cas de fuite de gaz. Cet objectif peut être divisé en trois objectifs nécessaires ω qui sont :

- ω_1 : détecter une anomalie de gaz
- ω_2 : alerter l'utilisateur par un son
- ω_3 : alerter l'utilisateur à l'aide d'une lumière clignotante

Les objectifs ω_2 et ω_3 sont inactifs au démarrage du système et sont activés si l'objectif ω_1 est atteint. En outre, l'agent a accès à un ensemble de dispositifs disséminés dans son environnement. Dans notre cas, nous allons nous intéresser à un sous-ensemble composé de trois objets connectés : une ampoule, une télévision, et un capteur de gaz. Ainsi, nous pouvons décrire les objets Obj de l'environnement au travers des dispositions n qu'elles présentent :

- Obj^1 : Le détecteur de gaz
  - n_1 détecter la présence de gaz dans la zone
- Obj^2 : La télévision
  - n_2 produire du son
  - n_3 afficher une image
- Obj^3 : L'ampoule connectée
  - n_4 faire de la lumière

Dans un cadre usuel, le service serait géré par une seule entité qui intégrerait l'ensemble des objectifs Ω et les limiterait à ceux-ci. Dans ce cas, il est possible de dissocier le service de surveillance, la captation d'information et la réaction du système. Lorsque le système démarre, l'agent A cherche à accomplir l'objectif Ω. Au vu de la présence du capteur de gaz Obj^1 dans son environnement et de sa disposition n_1 à détecter une fuite de gaz l'objectif ω_1 est rendu actif en permanence. Si une fuite de gaz est détectée, l'agent doit réagir et ses objectifs ω_2 et ω_3 deviennent actifs. Dans un cas optimal, où l'ampoule Obj^3 et la télévision Obj_2 sont accessibles, l'agent a la possibilité de déclencher les comportements associés aux objectifs d'alerte (ω_2 et ω_3). Si, dans un autre cas, la lampe n'est plus utilisable par l'agent, la télévision Obj_2 offre toujours la possibilité d'utiliser l'écran pour déclencher une alerte clignotante. Dans cet environnement, l'alerte sonore, qui est nécessaire pour l'objectif ω_2, pourra toujours être déclenchée grâce à la télévision. De cette façon, l'attention de la personne est attirée sur le problème de la fuite de gaz. De la même manière, cette même ampoule Obj^3 et cette même télévision Obj_2 peuvent servir de source d'alerte pour d'autres types de suivi tels que l'intrusion, la température, l'hydrométrie, etc. Sans avoir à ajouter forcément de nouveaux éléments dans notre environnement, nous sommes en capacité de mener à bien plusieurs services simultanément qui à priori ne sont pas proposés dans la conception initiale et individuelle des objets. Cette vision de notre environnement nous permet ainsi de diminuer les redondances au sein d'un système en privilégiant une utilisation plus intelligente et raisonnée des objets qui nous entourent.

2.2 Un défi écologique mais aussi un défi technique

Cette mutualisation nécessite cependant une transformation de nos habitudes au travers d'une solution permettant sa réalisation. Ainsi, il faut une solution dont le déploiement soit possible dès maintenant sur le plus d'appareils. Pour cela il faut donc éviter des modifications, physiques ou de bas niveaux, trop importantes, car cela limiterait son utilisation. De plus, à la vu du caractère ubiquitaire de notre

Type d'effecteur	Objets caractéristiques	Utilisations
Lumière	téléphone, télévision, ampoule connectée	Attire l'attention sur un emplacement
Chenillard	Électroménager	Transmettre un message simple
Écran	Électroménager, téléphone, télévision	Transmettre un message concret
Écran tactile	Tablette, smartphone	Apporte un message complet, avec possibilité d'interaction avec l'utilisateur
Haut-parleurs	Télévision, smartphone, alarme	Attirer l' attention sur un problème détecté ou transmettre un message
Vibration	smartphone, tablette, pda	Attirer l'attention vers l'écran de l'appareil

FIGURE 2 – Ensemble d'éléments réutilisables présents dans l'habitat

cadre d'usage, cela doit être suffisamment léger afin de permettre son déploiement sur des appareils disposant de faible de capacité de calcul. Enfin chaque entité doit être le plus autonome possible afin de pouvoir décider individuellement et de manière autonome des dispositions qu'il souhaite mutualiser et ne pas dépendre d'une plateforme unique.

3 État de l'art

Il existe de nombreux travaux liés à l'informatique ambiante appliqués à divers domaines tels que : la santé [19] ; la gestion intelligente des villes[4] ; le bureau intelligent[30]. Cette section va traiter des moyens courants, permettant la mise à disposition des objets connectés disséminés dans l'environnement. Pour cela, nous allons nous intéresser aux opportunités ainsi qu'aux limites pouvant être offertes à divers niveaux d'abstraction tels que : les infrastructures réseau, les middlewares IoT, les agents, le concept d'artefact.

3.1 Accès direct

Afin de mettre à disposition les objets connectés, le plus simple conceptuellement est de connecter chaque dispositif matériel, soit par liaison filaire, soit via une liaison sans fils (Wifi, Bluetooth, LoRa, 5G, etc.) à une « centrale de commande ». Ainsi, nous pouvons spécifier chaque fonctionnalité du système en amont, sans aucun problème de cohérence. Cette proposition, privilégiée par les solutions commerciales, permet un contrôle total des objets connectés et des services mis à disposition. Les exemples sur le marché sont nombreux comme les ampoules connectées, les thermostats ou les systèmes d'alarme, nécessitant un contrôleur domotique. Cette démarche commerciale, malgré son côté pratique pour le client, ne va pas dans le sens des prérogatives soulevées précédemment quant à la mutualisation des objets connectés. Dans certains cas, elle va à même l'encontre en proposant des solutions fermées, faisant barrière à la généricité, qui empêche d'avoir un accès direct aux divers composants.

3.2 Middleware pour gérer les environnements ambiants

Dans l'usage générique, le middleware (ou intergiciel) [26, 11] est le logiciel utilisé pour la communication entre deux applications dépourvues initialement de passerelles. Dans les systèmes ambiants, le middleware est en général considéré comme une couche générique qui fournit des fonctions de base pour cacher les détails de bas niveau de la couche physique [2, 1]. Cette approche est couramment utilisée dans la gestion d'environnement intelligent [3, 17, 9, 22, 17], mais aussi dans d'autres domaines tels que la robotique[25], l'Internet des objets[12] et les réseaux de capteurs [37]. Cependant une limite à l'utilisation des middlewares vient du fait qu'un système ne dispose généralement pas d'un seul middleware [20], mais en utilise autant qu'il y a de problèmes de communication [25]. Par conséquent, plus un système fait intervenir des entités de manière hétérogène, plus il risque de contenir un nombre important de middlewares afin d'en masquer les problèmes de communication. Dans notre cas, l'idée étant de mutualiser toutes les dispositions présentes au sein de nos objets connectés, il serait difficile de proposer un middleware permettant la mise à disposition de l'ensemble des types d'objets existants et à venir. Des propositions comme celle de FraSCAti proposent la mise en place d'un "intergiciel d'intergiciels hétérogènes" afin de proposer une interface unique aux applications [35]. Cependant même si ce type d'intergiciel propose un très grand nombre de fonctionnalités, il se complexifiera de plus en plus au fur et à mesure de l'apparition de nouvelles technologies. De plus, dans le cas d'une utilisation en contexte ubiquitaire, nous n'avons au préalable aucune connaissance sur les problématiques pouvant être rencontrées. Chaque appareil connecté devra alors supporter la charge totale du dispositif. Cette particularité limitera grandement son déploiement et amènera à écarter les appareils possédant de faibles capacités de calcul.

3.3 Agent

En informatique, un agent est une "entité computationnelle" capable de capter de l'information, d'agir « rationnellement » en fonction des données récupérées et de déclencher une action pour optimiser ses chances de succès. Dans un contexte d'intelligence ambiante, l'utilisation d'agents, grâce à leurs capacités à être autonomes, à agir en temps réel, à être social et proactif ([34, 36, 15, 18, 39]) devient une alternative solide aux logiciels traditionnels. En règle générale, un agent fait partie d'un système multi-agents (SMA ou *MAS* en anglais) qui est une collection organisée d'agents. Grâce à cette distribution de l'intelli-

gence, dans un contexte ubiquitaire, les agents représentent généralement des entités logicielles qui animent le système de manière autonome en fonction des capacités qu'elles possèdent. En se basant sur ce constat, certaines propositions tentent de dépasser les problèmes de mise à disposition des objets connectés en les liant à des agents intelligents [23, 21, 24, 29]. Ainsi chaque objet connecté est virtuellement représenté par un agent "interface" qui fait office de médiateur entre l'objet et les agents "standards" (FIGURE 3). Dans "The agentified use of internet of things" [21], les auteurs proposent une solution basée sur le concept Agent afin de permettre une mutualisation des composants disponibles. Cette approche, dont la problématique est similaire à celle proposée dans cet article, propose de lier chaque objet connecté à un agent. Ce dernier, ayant le contrôle total de l'objet, permet une utilisation de chaque composant de l'objet connecté de manière indépendante. Cependant, il est difficile de déployer des systèmes agents sur des dispositifs embarqués en raison de leur faible capacité de calcul [8]. De plus, sur le plan conceptuel, conformément aux définitions traditionnelles [40], il est déconseillé de modéliser un objet en utilisant le concept d'agent, car il n'en présente pas les caractéristiques [10][13] [18][39].

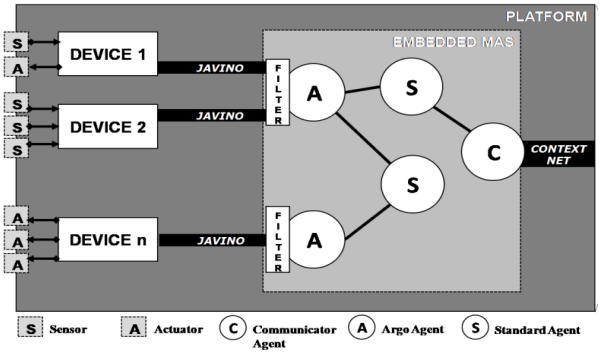


FIGURE 3 – Exemple d'utilisation d'agent utilisé comme médiateur entre objet et agents [29]

3.4 Les artefacts

Le méta-modèle A&A (Agent & Artifact) [27] se caractérise par trois principales abstractions :

- Les agents : composants proactifs des systèmes, encapsulant l'exécution des activités dans un environnement donné ;
- les artefacts : composants passifs des systèmes, tels que les ressources et les médias, voués à être utilisés par des agents pour soutenir leurs comportements ;
- Les workspaces : conteneurs conceptuels d'agents et d'artefacts, utiles pour définir les topologies de l'environnement et les notions de localité.

Selon la définition des auteurs, un artefact A&A est une entité de calcul destinée à être utilisée par des agents A&A. Le concept d'artefact est conçu pour s'adapter à n'importe quelle situation, ainsi, les artefacts sont modélisés sans but précis. C'est en effet au moment de leur conception que chaque artefact est lié à ses fonctions par le concepteur. Enfin, de manière explicite ou implicite, un artefact expose son interface, afin de mettre ses fonctions à la disposition des agents. On peut donner un exemple de son utilisation au travers du framework Cartago [31] au sein de JaCaMo[5] pour la programmation de systèmes multi-agents opérant dans des environnements distribués [33]. Le concept d'artefact, de par sa nature abstraite et n'étant pas conçu dans un but précis, ne résout malheureusement pas directement la problématique de la mutualisation des dispositions des objets présents dans notre environnement. Cependant, sa cohérence en termes de modélisation agent/objet et sa nature polymorphe en font un outil théorique intéressant pour notre proposition de modèle.

4 Description du concept et modèle d'architecture

Afin de diminuer l'impact environnemental des objets connectés, une solution est de proposer une communication plus intelligente entre les éléments présents tout autour de nous. Cela afin de créer un système où un même objet pourrait être lié à une multitude de services sans limites d'accès purement architectural. À la vu des contraintes apportées par une telle proposition et les éléments soulevés précédemment, nous pouvons déduire que les solutions actuelles ne répondent pas complètement aux exigences liées à notre proposition. À partir de ce constat, nous proposons

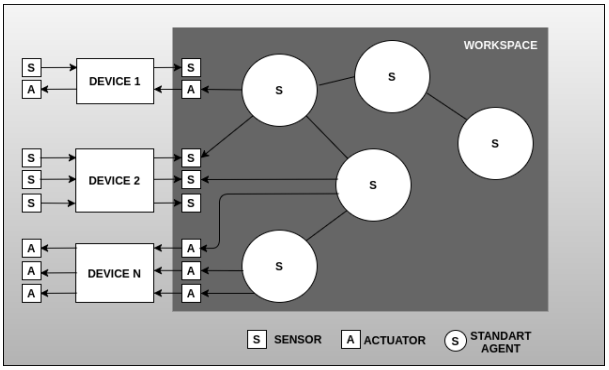


FIGURE 4 – Une mutualisation des composants

un modèle d'architecture permettant la représentation virtuelle des objets connectés sous la forme d'un ensemble d'artefacts. À la différence de l'utilisation des agents, la mutualisation se fait ici au travers d'un paradigme cohérent. De plus, les artefacts, de par leurs uniformités et leurs hauts niveaux d'abstraction, permettent de lutter contre la surabondance de middlewares. Nous allons ici décrire de manière plus détaillée le modèle d'architecture lié aux artefacts permettant la mutualisation des dispositions d'un objet connecté. À la vu de la similarité avec le méta-modèle A&A ,qui à été développer dans l'état de l'art, nous ne rappellerons que brièvement le fonctionnement global du système mettant en oeuvre ,en plus des artefacts, les agents,

représentant l'entité proactive générant le service , et les workspaces. Comme décris dans l'état de l'art, le workspace est utilisé comme conteneur conceptuel d'agents et d'artefacts. Son rôle est pour cela grandement lié aux fonctions permettant respectivement de rendre disponible l'artefact ou de l'enlever du workspace. Un agent ne pourra alors accéder qu'aux artefacts présents au sein du workspace auquel il appartient (FIGURE 4). Grâce à cela nous évitons une diffusion incontrôlée des dispositions de nos objets connectés, via un filtrage de l'accès. De plus les workspaces peuvent être thématisés afin de mettre en avant certaines caractéristiques tels que le type de service (sécurité, bien-être, etc.) ou encore de lieu (cuisine, salle de bain, etc.).

4.1 Le modèle d'architecture

Le modèle est composé de huit classes principales (Figure 5) : Artifact, Composite Artifact, Main Artifact, Final Artifact, SensorArtifact, ActuatorArtifact, ServiceArtifact.

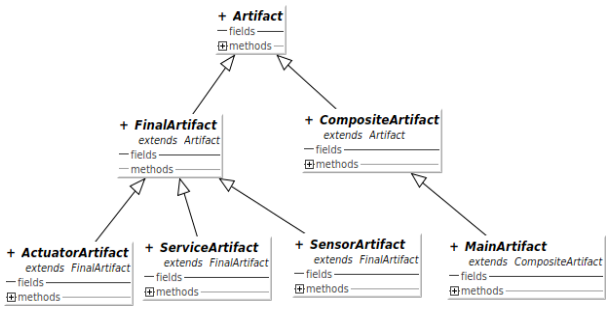


FIGURE 5 – Diagramme de classe du modèle proposé illustrant les relations d'héritage

Artifact Cette classe définit l'ensemble des primitives utiles pour programmer le comportement observable des artefacts selon les frameworks ou les environnements d'implémentation. Cette première classe permet la mise en place du lien de dépendance entre artefacts via $addMaster$, ainsi que son état observable au sein des workspaces grâce aux méthodes $exposeArtifact$ et $unexposeArtifact$.

Composite Artifact La classe CompositeArtifact est principalement un conteneur d'Artifact. Son rôle est de garder une trace de la structure de l'objet connecté originel. Il est d'ailleurs possible qu'il soit lui-même sous la gestion d'un artefact s'il est de plus un élément composant.

MainArtifact La classe MainArtifact représente virtuellement l'objet connecté. Elle contient les informations liées à l'objet (nature, état, adresse mac, etc.). Cette classe est celle en charge de la sécurité et de l'accès aux artefacts. En effet, il n'y a que cette classe qui est en lien direct avec les workspaces.

Final Artifact Les éléments de cette classe représentent les artefacts directement mis à disposition dans le workspace. Toutes les classes qui en héritent représentent les feuilles de l'arbre de décomposition de l'objet originel et ne peuvent donc avoir des artefacts qui les composent.

SensorArtifact Les Sensors ont comme fonction unique de notifier aux éléments intéressés les changements opérés au travers de sa captation d'informations. Pour cela, le concepteur est libre de choisir le type de moyen de diffusion. Soit de manière directe via la méthode $getObservableState$ ou encore grâce à un système de Publish/Subscrib. L'utilisation des artefacts possède de plus l'avantage de proposer une solution à deux problèmes clés liés au concept Publish/Subscrib que l'on utilisera généralement pour définir les moyens de transmission de l'information des capteurs. Premièrement, le problème de la mise en relation efficace d'un événement avec un grand nombre d'abonnés sur un type d'événement peut être réglé en "délocalisant" sur un autre appareil l'objet Sensor associé. L'objet connecté n'aura alors qu'à gérer un seul Subscriber par capteur. La diffusion vers les entités intéressées se faisant au travers des artefacts Sensors. Deuxièmement, le problème de la multidiffusion efficace d'événements au sein d'un réseau de fournisseurs d'événements est réglé par le fait que l'agent se met en relation directement avec le capteur souhaité. Ainsi les entités intéressées n'auront accès qu'aux informations pertinentes pour elles.

Actuator Artifact Les Actuators ont comme fonction de réagir aux stimuli reçus en exécutant la tâche demandée. Pour cela les agents les actionnent via les fonctions $genEvent$. L'actuator renvoie un booléen reflétant, en règle générale, une indication quant à la prise en compte de l'action à effectuer. En effet, à la manière d'un bouton "on/off" lié à une lampe, cette fonction permet de confirmer le changement d'état du bouton sans pour autant donner des informations sur le changement d'état effectif de la lampe.

Service Artifact Les Services représentent les artefacts ne pouvant ni être catalogué en tant que Sensor ni en tant qu'Actuator (FIGURE 6). Cela peut être le cas d'artefacts gérant par exemple le codage de flux vidéo ou l'accès à des services externes tels que la reconnaissance vocale. Cette demande se fait au travers des méthodes $genService$ qui retournent à l'agent un objet répondant à sa demande.

Type de service	Utilisation
Reconnaissance vocal	Demande de confirmation
Notification	Alerte non bloquante avec retour possible
Pop-up	Alerte bloquante
Authentification	Vérification d'identité
Messagerie	Dialogue

FIGURE 6 – Différents services possibles

5 Une implémentation sur Android

Une implémentation sur Android a été réalisée afin de mettre à l'épreuve la faisabilité du modèle. Le choix de la plateforme Android n'est pas anodin, en effet en 2019 Google a annoncé avoir atteint les 2,5 milliards de terminaux Android actifs, ce qui en fait le système d'exploitation

Modèle	Samsung galaxy s8	Galaxy Tab A (2016)	Acer TravelMate P257
Cadence Processeur	2,3 Ghz	1.6GHz	2,4 GHz
Système	Android 9.0	Android 7.0	Ubuntu 16.04
RAM	4 Go	2 Go	6Go
Stockage	64 Go	16 Go	500Go
Connectivité (indicatif)	4G LTE Advanced, NFC, Bluetooth 5.0, GPS, Wi-Fi 802.11 , lecteur d'empreintes digitales, reconnaissance d'iris/faciale, USB Type C	ANT+, USB 2.0,GPS, Glonass, prise jack 3.5mm Stereo,Bluetooth v4.2, Wi-Fi 802.11	Wi-Fi AC, Bluetooth 4.0, Webcam, Jack 3.5mm, RJ45, USB 2.0, USB 3.0, HDMI Femelle, VGA (D-sub 15 Femelle)
Dimensions	148.9 x 68.1 x 8 mm pour 152 g	254.2 x 155.3 x 8.2 pour 525 g	256 x 381.6 mm x 29.2 mm pour 2.4 kg

FIGURE 7 – Descriptif des appareils utilisés lors de la phase de test

le plus répandu. De plus le système est conçu pour les objets connectés tels que les smartphones, tablettes tactiles, ordinateurs, télévisions (Android TV), voitures (Android Auto), Chromebook et autre montres connectées (Wear OS). Ce panel de produits illustre parfaitement les types d'appareils les plus à même d'être porteur de capteurs et d'effecteurs dont les usages ne sont pas mutualisés. Afin de mettre en place ce modèle générique, nous avons basé la phase d'implémentation sur la plateforme "Software Kit for Ubiquitous Agent Development" (SKUAD) qui permet de créer des agents ambiants capables de manipuler des capteurs et des effecteurs[14] dont les performances et la conception permettent un usage embarqué. Nous allons en premier lieu décrire les spécificités de cette implémentation sur Android avant de les illustrer par la phase de test.

5.1 Implémentation

L'application (format APK) pèse 9.13 Mo, pour une consommation de 30 Mo de RAM en moyenne ainsi qu'un maximum de 100 Mo de RAM pour une séance de 3h d'utilisation. Elle permet, pour l'instant, la détection de tous les capteurs embarqués au sein de l'appareil et propose sept effecteurs et deux services. Du point de vue de l'implémentation, la classe Artifact comporte, en plus des primitives de base liées au comportement observable des artefacts, l'ensemble des méthodes spécifiques liées aux contraintes d'Android telles que la création d'IHM, les accès aux permissions et au stockage de l'appareil. En héritant de la classe Artifact, nous avons implémenté les autres classes décrites dans le modèle. Dans le but d'avoir une vue globale, nous allons nous concentrer sur la classe AndroidMainArtifact (classe fille de MainArtifact) ainsi que sur celles héritant de FinalArtifact. Afin de s'adapter à tous types d'appareils Android la classe AndroidMainArtifact a pour fonction principale d'analyser les possibilités offertes par l'objet connecté et de demander à l'utilisateur les autorisations associées à leurs fonctionnements. Les artefacts liés aux dispositions sont alors créés dynamiquement et automatiquement en introspectant l'appareil. Au sein du programme, les artefacts sont créés au travers de classes héritant de FinalArtifact : ArtifactActuator, AndroidSensor, ArtifactService. Nous pouvons donner comme exemple les classes Light (ArtifactActuator) et VoiceRecognition (ArtifactService) représentant respectivement les catégories d'artefacts liés à la production de lumière et à la reconnaissance vocale. Si un composant est disponible et que les autorisations le permettent, la classe AndroidMainArtifact procède alors à la création des artefacts en tant qu'instance de la classe associée. Il revient alors à l'utilisateur de choisir, via l'interface graphique, quels artefacts mettre à disposition et quels agents activer au sein du workspace. L'utilisateur et les agents peuvent aussi voir et utiliser les artefacts proposés par d'autres appareils, mais ils ne seront pas en mesure de modifier leurs caractéristiques ou leurs visibilités dans le workspace.

5.2 Mise en contexte

Pour illustrer les principaux concepts proposés, nous allons présenter un exemple d'utilisation. Les appareils utilisés lors de cette phase de test sont : un smartphone de type Samsung galaxy S8, une tablette de type Samsung tab A (2016) et un ordinateur portable de type Acer TravelMate P257 (FIGURE 7). Au terme de la phase d'analyse des dispositions des appareils sous Android le galaxy S8 propose vingt-six Sensors et le Tab A en propose cinq. Avec deux appareils Android, nous en sommes donc déjà à quarante-neuf artefacts pouvant être mis à disposition de l'utilisateur et des agents. Afin d'illustrer le concept, nous avons effectué une utilisation de l'accéléromètre, servant habituellement pour déterminer l'orientation de l'écran ou stabiliser la prise de photographie, en tant que détecteur de chocs. Ainsi, l'entité intelligente, dans cette implémentation un agent SKUAD, devient à même, en ajustant les paramètres de sensibilité, de discriminer différents types de chocs (chute, hauteurs, sens, présence d'un choc final). En cas de choc, l'agent cherchera à attirer l'attention de l'utilisateur via le flash ou le vibreur des appareils Android ou encore en faisant clignoter l'écran de l'ordinateur. Si l'agent n'a pas de réponse, il peut alors utiliser la synthèse vocale ainsi que la reconnaissance vocale afin de dialoguer avec un utilisateur. Pour ce cas, l'agent pose une suite de questions fermées donnant le choix à la personne interrogée de répondre parmi un ensemble de réponses prédéfinies. Chaque réponse prononcée par l'utilisateur sera transformée en texte puis analysée par l'agent afin de faire un constat de la situation.

5.3 Des résultats encourageants

À la vu des résultats obtenus, nous avons constaté qu'un simple smartphone ou une tablette suffirait à animer bon nombre de services. Nous pouvons donner comme exemple le cas, pourtant simple, présenté précédemment où nous nous trouvons virtuellement et schématiquement au sein d'un système ayant 3 processeurs, 3 batteries, 12 Go de Ram ainsi de 580 Go de stockage (FIGURE 7). Cette puissance est cependant peu utile dans un usage courant. En effet, les entités de captation et de diffusion peuvent être externalisées sans que l'on ait besoin de leurs ajouter de composants les dotant d'une capacité de traitement. Cela permettrait notamment de diminuer le nombre de batteries, d'écrans et de composants liés au calcul (RAM, processeur, stockage). La dynamique actuelle cherchant à rendre l'environnement intelligent passerait alors, non pas par une intégration systématique de puissance de calcul au sein de chaque objet, mais plutôt par un usage plus raisonné et mutualisé des possibilités de nos appareils. De plus, cette solution offre une alternative au cloud computing, car les traitements peuvent être faits en local sur les périphériques qui en ont la capacité. Cela permet de diminuer l'empreinte énergétique, puisque qu'on ne sollicite ni le réseau internet et ni les data center, tout en limitant les fuites de donnés personnelles.

6 Conclusion et Perspectives

Dans cet article, nous avons proposé une nouvelle vision de l'interaction avec notre environnement de plus en plus informatisé. À l'heure actuelle où la tendance converge vers une communautarisation des technologies, cette approche a pour but de limiter l'impact environnemental des objets connectés en suggérant leurs mutualisations. Grâce à un modèle d'architecture inspiré du concept d'artefact, nous proposons de mettre en avant virtuellement les dispositions de nos appareils connectés afin de diminuer les redondances au sein d'un environnement. Nous avons effectué une preuve du concept au travers de l'implémentation du modèle sous Android. Cette étape nous a permis de mettre en avant le fait qu'une solution serait une utilisation plus intelligente de la multitude d'opportunités non exploitées résidant au sein de nos environnements. Nous pensons désormais étendre la phase de test du modèle à d'autres types d'objets tels que les télévisions, les montres, et autres objets connectés. D'autre part, nous pensons que les fonctions offertes par les artefacts seraient enrichies par l'ajout d'une sémantique. En effet, de nombreuses recherches ont démontré l'intérêt des ontologies dans le cadre de l'exploitation des objets par un logiciel [32] [28] [16]. L'utilisation d'une ontologie serait donc tout indiquée dans le cas d'une mutualisation pour permettre une description précise et cohérente des possibilités offertes par les artefacts.

Références

[1] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4) :263–277, 2007.

[2] Christian Becker and Gregor Schiele. Middleware and application adaptation requirements and their support in pervasive computing. In *23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings.*, pages 98–103. IEEE, 2003.

[3] Fabio Bellifemine, Giovanni Caire, Agostino Poggi, and Giovanni Rimassa. Jade : A software framework for developing multi-agent applications. lessons learned. *Information and Software Technology*, 50(1-2) :10–21, 2008.

[4] Marc Böhlen and Hans Frei. Ambient intelligence in the city overview and new perspectives. In *Handbook of ambient intelligence and smart environments*, pages 911–938. Springer, 2010.

[5] Olivier Boissier, Rafael H Bordini, Jomi F Hübner, Alessandro Ricci, and Andrea Santi. Multi-agent oriented programming with jacamo. *Science of Computer Programming*, 78(6) :747–761, 2013.

[6] Frédéric Bordage. Study - The environmental footprint of the digital world. *GreenIT*, page 39, 2019.

[7] Laure Cailloce. 2.1. numérique : le grand gâchis énergétique. *CNRS Le journal*, 2018.

[8] Davide Calvaresi, Mauro Marinoni, Aldo Franco Dragoni, Roger Hilfiker, and Michael Schumacher. Real-time multi-agent systems for telerehabilitation scenarios. *Artificial Intelligence in Medicine*, 2019.

[9] LM Camarinha-Matos, Hamideh Afsarmanesh, et al. Telecare : Collaborative virtual elderly care support communities. *The Journal on Information Technology in Healthcare*, 2(2) :73–86, 2004.

[10] Krzysztof Cetnarowicz. From algorithm to agent. In *International Conference on Computational Science*, pages 825–834. Springer, 2009.

[11] Anthony Chandor, John Graham, Robin Williamson, et al. *Dictionary of computers*. Eng.] Penguin Books, 1970.

[12] Mauro AA da Cruz, Joel José PC Rodrigues, Jalal Al-Muhtadi, Valery V Korotaev, and Victor Hugo C de Albuquerque. A reference model for internet of things middleware. *IEEE Internet of Things Journal*, 5(2) :871–883, 2018.

[13] Scott A DeLoach. Multiagent systems engineering : A methodology and language for designing agent systems. Technical report, Department of Electrical and Computer Engineering of Air Force Institute of Technology, 1999.

[14] Payet Denis. Skuad, software kit for ubiquitous agent development. http ://skuad.onover.top/.

[15] Jacques Ferber. *Les Sysèmes Multi Agents : vers une intelligence collective*. InterEditions, 1995.

[16] Artur Freitas, Alison R Panisson, Lucas Hilgert, Felipe Meneguzzi, Renata Vieira, and Rafael H Bordini. Integrating ontologies with multi-agent systems through cartago artifacts. In *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, volume 2, pages 143–150. IEEE, 2015.

[17] Benjamin Gateau, Yannick Naudet, and Jarogniew Rykowski. Ontology-based smart iot engine for personal comfort management. In *2016 11th International Workshop on Semantic and Social Media Adaptation and Personalization (SMAP)*, pages 35–40. IEEE, 2016.

[18] Nicholas R Jennings. On agent-based software engineering. *Artificial intelligence*, 117(2) :277–296, 2000.

[19] Achilles Kameas and Ioannis Calemis. Pervasive systems in health care. In *Handbook of ambient intelligence and smart environments*, pages 315–346. Springer, 2010.

[20] Kristian Ellebæk Kjær. A survey of context-aware middleware. In *Proceedings of the 25th conference on IASTED International Multi-Conference : Software Engineering*, pages 148–155. ACTA Press, 2007.

[21] Jöel Kwan, Yassine Gangat, Denis Payet, and Rémy Courdier. A agentified use of the internet of things. In *Full Paper in 9th IEEE International Conference on Internet of Things(iThings 2016)*. IEEE CS, 2016.

[22] Gokce B. Laleci, Asuman Dogac, Mehmet Olduz, Ibrahim Tasyurt, Mustafa Yuksel, and Alper Okcan. SAPHIRE : a multi-agent system for remote healthcare monitoring through computerized clinical guidelines. In *Agent technology and e-health*, pages 25–44. Springer, 2007.

[23] Zakaria Maamar, Noura Faci, Khouloud Boukadi, Emir Ugljanin, Mohamed Sellami, Thar Baker, and Rafael Angarita. How to agentify the internet-of-things ? In *2018 12th International Conference on Research Challenges in Information Science (RCIS)*, pages 1–6. IEEE, 2018.

[24] Zakaria Maamar, Noura Faci, Slim Kallel, Mohamed Sellami, and Emir Ugljanin. Software agents meet internet of things. *Internet Technology Letters*, 1(3) :e17, 2018.

[25] Nader Mohamed, Jameela Al-Jaroodi, and Imad Jawhar. Middleware for robotics : A survey. In *RAM*, pages 736–742, 2008.

[26] Peter Naur and Randell B. Software engineering-report on a conference sponsored by the nato science committee garimisch, germany. *http ://homepages. cs. ncl. ac. uk/brian. randell/NATO/nato1968. PDF*, 1968.

[27] Andrea Omicini, Alessandro Ricci, and Mirko Viroli. Artifacts in the a&a meta-model for multi-agent systems. *Autonomous agents and multi-agent systems*, 17(3) :432–456, 2008.

[28] Alison R Panisson, Artur Freitas, Daniela Schmidt, Lucas Hilgert, Felipe Meneguzzi, Renata Vieira, and Rafael H Bordini. Arguing about task reallocation using ontological information in multi-agent systems. In *12th International Workshop on Argumentation in Multiagent Systems*, volume 108, 2015.

[29] Carlos Eduardo Pantoja, Heder Dorneles Soares, José Viterbo, and Amal El Fallah-Seghrouchni. An architecture for the development of ambient intelligence systems managed by embedded agents. In *SEKE*, pages 215–214, 2018.

[30] Carlos Ramos, Goreti Marreiros, Ricardo Santos, and Carlos Filipe Freitas. Smart offices and intelligent decision rooms. In *Handbook of Ambient Intelligence and Smart Environments*, pages 851–880. Springer, 2010.

[31] Alessandro Ricci, Michele Piunti, Mirko Viroli, and Andrea Omicini. Environment programming in cartago. In *Multi-agent programming*, pages 259–288. Springer, 2009.

[32] Alessandro Ricci, Mirko Viroli, and Andrea Omicini. Carta go : A framework for prototyping artifact-based environments in mas. In Danny Weyns, H. Van Dyke Parunak, and FabienEditors Michel, editors, *Environments for Multi-Agent Systems III*, volume 4389, page 67–86. Springer Berlin Heidelberg, 2007.

[33] Mário Lucio Roloff, Marcelo Ricardo Stemmer, Jomi Fred Hübner, Robert Schmitt, Tilo Pfeifer, and Guido Hüttemann. A multi-agent system for the production control of printed circuit boards using jacamo and prometheus aeolus. In *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*, pages 236–241. IEEE, 2014.

[34] Stuart J Russell. Rationality and intelligence. *Artificial Intelligence*, 1(94) :57–77, 1997.

[35] Lionel Seinturier, Philippe Merle, Romain Rouvoy, Daniel Romero, Valerio Schiavoni, and Jean-Bernard Stefani. A Component-Based Middleware Platform for Reconfigurable Service-Oriented Architectures. *Software : Practice and Experience*, 42(5) :559–583, May 2012.

[36] Yoav Shoham. Agent-oriented programming. *Artificial intelligence*, 60(1) :51–92, 1993.

[37] Miao-Miao Wang, Jian-Nong Cao, Jing Li, and Sajal K Dasi. Middleware for wireless sensor networks : A survey. *Journal of computer science and technology*, 23(3) :305–326, 2008.

[38] Mark Weiser. Some computer science issues in ubiquitous computing. *Communications of the ACM*, 36(7) :75–84, 1993.

[39] Michael Wooldridge. Agent-based software engineering. *IEE Proceedings-software*, 144(1) :26–37, 1997.

[40] Michael Wooldridge and Nicholas R Jennings. Intelligent agents : Theory and practice. *The knowledge engineering review*, 10(2) :115–152, 1995.

Vers une utilisation éco responsable des objets connectés par la mutualisation de leurs composants physiques : Une approche basée sur le concept d'artefact