Symboling : utiliser des structures symboliques dotées d'une métrique

Paul Bernard, Benjamin Hate, Morgane Laval

Symboling : utiliser des structures symboliques dotées d'une métrique

Paul BERNARD, Benjamin HATE, Morgane LAVAL

RESEARCH REPORT

N° 9499

Février 2023

Project-Team Mnemosyne.

ISSN 0249-6399

Symboling : utiliser des structures symboliques dotées d'une métrique

Paul Bernard[1] , Benjamin Hate[2],
Morgane Laval[3]
Project-Teams Mnemosyne

Research Report N° 9999 — Février 2023 — 18 pages.

Résumé: Nous proposons de manipuler des structures de données symboliques au sein d'algorithmes d'apprentissage automatique habituels, par exemple en considérant un système réactif engagé dans une tâche de résolution de problèmes ouverte et mal définie. Nous définissons les tâches de résolution de problèmes à un niveau géométrique, en considérant que nous sommes situés quelque part dans un espace d'états, dans le but d'atteindre un état final (unique ou alternatif, éventuellement partiellement défini), et de trouver un chemin du premier au dernier tout en respectant les contraintes du chemin.
L'idée centrale de cette définition géométrique est de considérer un espace d'état abstrait où chaque point est une structure de données symboliques, représentant des informations contextuelles sur l'espace physique et sur l'état interne de l'agent. Sélectionner une trajectoire locale correspond à décider, comme étape du processus de résolution du problème, de modifier certaines caractéristiques de l'espace d'états tant au niveau externe (e.g., déplacer un objet) qu'au niveau interne (i.e., modifier la représentation interne).
L'ingrédient principal est de spécifier une distance. Le levier est une notion de distance d'édition, c'est-à-dire le fait qu'une valeur de donnée symbolique est éditée pas à pas pour égaler une autre valeur. De plus, étant donné un type de données, cette spécification inclut la projection d'une valeur de données au voisinage de la région spécifiant un type de données sur celui-ci, comme développé ici.
Ce rapport introduit ces notions plutôt abstraites en les rendant les plus accessibles possibles, notamment en montrant leur relation avec la modélisation en neuroscience computationnelle et en les illustrant à l'aide d'un exemple de dessins et d'un exemple musical.

Mots-clés: Résolution de problèmes. Représentations symboliques. Distance d'édition.

[1] ENSC - Bordeaux INP – pbernard007@ensc.fr; [2]ENSC – pbhate@ensc.fr; [3]ENSC – molaval@ensc.fr;

RESEARCH CENTRE
BORDEAUX - SUD-OUEST
351 Cours de la Libération
Bâtiment A29
33405 Talence Cedex France

Symboling : using symbolic structures equipped with a metric.

Abstract: We consider manipulating symbolic data structure within usual machine learning algorithms, for instance considering a reactive system engaged in some open-ended, ill-defined problem-solving task. We define the problem-solving tasks at a geometric level, considering being located somewhere in a state-space, with the goal of reaching some final (unique or alternative, eventually partially defined) state, and finding a way from the former to the latter while satisfying the path constraints.

The pivotal idea of this geometric definition is to consider an abstract state space where each point is a symbolic data structure, representing contextual information about the physical space and about the agent's internal state. Selecting a local trajectory corresponds to deciding, as a step in the problem-solving process, to modify some characteristics of the state space both at the external level (e.g., moving an object) and at the internal level (i.e., modifying the internal representation).

The primary ingredient is to specify a distance. The lever is the notion of editing distance, i.e., the fact that a symbolic data value is step-by-step edited in order to equal another value. Moreover, given a data type, this specification includes the projection of a data value in the neighborhood of the data type region onto it, as developed here.

This French report introduces these rather absconding notions making them as much as possible accessible, including showing their relation with computational neuroscience modeling and illustrating them using one drawing and one musical example.

Keywords: Problem-Solving. Symbolic representation. Editing distance.

1 Introduction 6
2 Résolution cognitives de problèmes 7
3 L'approche dite "symboling" en résolution de problèmes 10
3.1 Modélisation hiérarchique 10
3.2 Introduction d'une métrique 12
3.3 Calcul effectif de la métrique 14
3.4 Application à la résolution de problèmes 15
4 Expérimentation du logiciel à évaluer 16
4.1 Utilisation de la syntaxe wJSON 16
4.2 Démonstrations illustratives 17
4.2.1 Démonstration graphique : "La tête à Toto vers la tête à Titi" 17
4.2.2 Démonstration musicale : la distance entre deux musiques 18
5 Conclusion 18
6 Contributions 19

1 Introduction

Un des défis du 21e siècle, en matière d'éducation, est de comprendre comment l'humain apprend et résout des problèmes, y compris des problèmes ouverts qui nécessitent des solutions dites créatives. On parle de compétences du 21ème siècle[2], comme le présente Margarida Romero dans la figure ci-dessous[3] :

FIGURE 1 : Présentation des compétences du 21ème siècle selon (Lambropoulos et Romero 2015)

Pour cela, différents modèles cognitifs sont étudiés et reliés à des modèles informatiques d'apprentissage profond. D'un point de vue cognitif, les recherches se focalisent sur des modèles du comportement d'un sujet influencé par une récompense ou dirigé vers un but. En apprentissage profond, ce comportement peut être

[2] Voir https://en.wikipedia.org/wiki/21st_century_skills et une discussion ici https://www.competencesdu21emesiecle.com/decouvrir/qu-est-ce-que-les-competences-du-21eme-siecle

[3] Lambropoulos, N., & Romero, M. (2015). 21st Century Lifelong Learning: Individual, Team and Social skills and competence-based methodologies. Nova Publishers.
Voir aussi pour une introduction https://www.researchgate.net/publication/323174769_Les_competences_pour_le_XXI_e_siecle la figure est reprise de la présentation française du papier en conférence.

modélisé par de l'apprentissage par renforcement, utilisant à la fois des récompenses intermédiaires et des récompenses finales pour atteindre un but. Nous allons présenter un modèle des fonctions exécutives dit PROBe qui met en œuvre ces principes généraux.

Le problème de cette modélisation numérique est qu'il est difficile d'ajouter des connaissances a priori, ou d'interpréter la sémantique des mécanismes numériques. Il est difficile de rendre explicites les interprétations sous-jacentes existantes dans le modèle neuronal. Bref, mélanger données symboliques et numériques est un problème difficile. Une solution proposée et qui sera explicitée ici est d'utiliser un formalisme spécifique pour les données, qui correspond à une description symbolique mais sur lequel on peut faire des calculs numériques correspondant aux algorithmes dont on a besoin. Nous allons détailler ces mécanismes en essayant de les rendre accessibles au maximum.

La proposition est de considérer une structure hiérarchique semblable à ce que permet de représenter la syntaxe JSON, qui sera ensuite interprété comme un ensemble de faits d'une base de connaissance[4] utilisable, entre autres, par un raisonneur afin d'enrichir ces connaissances factuelles. Ici on se concentrera sur la résolution de problèmes et on verra comment transformer cela en un problème géométrique de génération de trajectoires.

Enfin nous montrerons une illustration très concrète de ces mécanismes à la fois pour valider l'implémentation logicielle et permettre de se représenter ce qui est formalisé ici.

2 Résolution cognitives de problèmes

Révisons brièvement comment est formalisé les mécanismes dits des fonctions exécutives qui sont en jeu au niveau cérébral lors de la résolution de problèmes. Ce modèle[5] dit PROBe introduit des patterns d'actions prédéfinies pour une tâche spécifique (Collins et Koechlin, 2012). Ces patterns sont appelés des Task-Sets (TS) et dictent le comportement du sujet.

Chaque TS possède son propre apprentissage par renforcement[6] (RL pour Reinforcement Learning), son propre jugement critique vis-à-vis d'une action. Elle est sa propre loi (appelée "policy") pour sélectionner une action a au regard d'un stimulus reçu s par l'environnement. La récompense associée à une tâche sélectionnée dans un TS est définie par à la loi de mise à jour[7] de (Sutton et Barto, 2018) :

$$\mathbf{Q}(s,a)(t+1) = \mathbf{Q}(s,a)(t) + \alpha(r(t)-\mathbf{Q}(s,a)(t))$$

où Q(s,a)(t) est une estimation des valeurs à l'instant t, α un réel pondérateur selon le contexte, et r(t) la récompense reçue à l'instant t. Cette formule signifie que la valeur d'un choix d'action à un instant t+1 dépend de celui fait à l'instant d'avant pondéré par sa récompense (ou pénalité !) reçue. On parle de mécanisme

[4] La représentation moderne des connaissances distingue les faits ou T-box (pour "terminology component") qui sont ici représenté hiérarchiquement des règles ou A-box (pour "assertion component) qui permettent de déduire des conséquences des faits posés https://en.wikipedia.org/wiki/Abox : on ne considère ici que le composant T-box.
On consultera https://line.gitlabpages.inria.fr/aide-group/wjson/LValue.html pour voir comment avec une interprétation sémantique dite "turtoise" on peut passer d'une représentation hiérarchique à un ensemble de triplets sémantiques comme utilisés dans les ontologies.

[5] Collins, A., & Koechlin, E. (2012). Reasoning, learning, and creativity: frontal lobe function and human decision-making. PLoS biology, 10(3), e1001293.

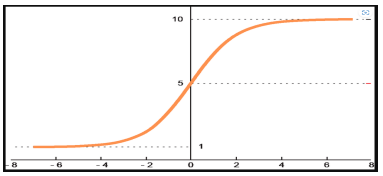
[6] L'apprentissage par renforcement est une branche de l'apprentissage, y compris avec des réseaux de neurones profonds. L'agent apprend par lui-même, par le biais de récompenses reçues en fonction des actions qu'il entreprend. Lorsque l'agent commence, il ne connaît rien de son monde et doit donc l'explorer pour déterminer les actions qui lui ramènent le plus de récompenses dans le temps.
On considère un agent qui interagit avec son environnement. A l'instant t, il reçoit un stimulus s de l'environnement (issu de son interaction avec celui-ci) avec une récompense associée r. À partir de cette réception, il va inférer des causes et préparer une action adéquate a.

[7] Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

Markovien pour expliciter que , les valeurs perçues à t+1 ne dépendent que des valeurs à t (processus de Markov d'ordre 1).

Les actions sélectionnées sont choisies par la loi du jeu de TS par un filtrage softmax[8] sur les valeurs de Q de la fortme:

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{ pour tout } j \in \{1, \ldots, K\}.$$



qui permet de calculer un maximum pondéré sur plusieurs valeurs avec une sortie en forme de sigmoïde.

On peut alors estimer pour chaque TS la probabilité de recevoir la récompense r selon l'action a et le stimulus s reçu :

$$\gamma_i = P(r/s,a,TS_i).$$

Une autre variable λ, appelée signal de responsabilité, mesure la confiance de la personne que son TS est adapté à la situation à laquelle elle est confrontée en se basant sur ses expériences passées :

$$\lambda_i = P(TS_i \text{ correct } / \ r, \text{ past}) \qquad \lambda_i(t+1) = \frac{\gamma_i \lambda_i(t)}{\sum \gamma_j \lambda_j(t)}.$$

On considère maintenant un seuil minimal acceptable de λ qui est 1/2. On appelle TS par défaut (TSd) le TS pour lequel cet indice de confiance est supérieur à ½. Autrement dit, pour un environnement, il existe un ensemble d'actions prédéfinies que l'agent va effectuer par défaut s'il est confronté à celui-ci. Ce TSd dirige le comportement tant que sa valeur reste au-delà de ½. Dans ce modèle, c'est la variable supervisant le comportement et indiquant si un comportement est approprié ou pas.

Lorsque ce n'est plus le cas, il faut changer de tâche. On peut considérer que tous les TSi non pris jusqu'à présent peuvent servir de TSd. En d'autres termes, si les règles ne sont plus les bonnes, on utilise celles que l'on a pas utilisées comme nouveau standard car elles ne sont pas encore invalidées par la réponse extérieure. Pour faire un autre parallèle, lorsque notre réaction à un stimulus est acceptable, on continue de l'exécuter. Lorsqu'il ne devient plus acceptable, on tente d'autres actions qui ont une probabilité supérieure à ½ de l'être. Avec le temps, un nouveau task-set standard émerge et devient le nouveau mécanisme habituel. Cependant ces actions associées ne le deviennent effectivement et ne sont stockés comme telles que lorsque le signal de responsabilité passe au dessus de ½ de manière permanente.

Ce modèle permet de créer, stocker, changer et récupérer des jeux de tâches qui gouvernent le comportement en fonction des retours de l'environnement et d'un passif.

[8] La fonction Softmax, ou fonction exponentielle normalisée est une fonction d'activation régulièrement utilisée dans les réseaux de neurones.

Les simulations de ce modèle montrent que contrairement à l'apprentissage par renforcement classique, les performances augmentent et l'exploration diminue lorsque des contingences extérieures déjà vues reviennent, notamment parce que l'on récupère des tâches apprises en conséquence. Ce modèle imite bien la tendance naturelle que l'on a de rester sur des actions qui fonctionnent plutôt que de tenter quelque chose qui pourrait être inapproprié. Ce modèle apprend aussi très rapidement et flexiblement des nouvelles combinaisons de tâches qui rivalisent avec celles déjà apprises. Par contraste, dans les différents modèles d'apprentissage par renforcement, l'apprentissage de TS se fait en parallèle suivant les signaux de responsabilité. Ce mécanisme en parallèle ralentit considérablement l'apprentissage de nouvelles combinaisons, surtout lorsque l'espace des task-sets est large.

Néanmoins, le signal de responsabilités décrit au-dessus ne réagit qu'a posteriori d'une action. C'est-à-dire que pour qu'un changement d'action s'opère, il faut recevoir subitement un retour négatif, ce qui est assez inconvenant dans un contexte réel (il serait dommage de se brûler la main pour savoir qu'il ne faut pas mettre la main au feu !).
Il faut donc un signal de responsabilité a priori qui donnerai une estimation de future acceptabilité de l'action qui va être employée en fonction des indices perçus et des expériences passées dans un contexte C similaire :

$$\mu_i = \mathrm{P}(\mathrm{TS_i\ correct} / \boldsymbol{C}, \mathrm{past}),$$

On parle ainsi d'ensemble d'actions contextuelles (de valeur Q(C, TSi)). En utilisant l'inférence bayésienne (calcul de la probabilité de confiance en une cause hypothétique à partir de nos connaissances antérieures et de la donnée reçue. Ce processus est itératif et est continuellement alimenté par la confirmation ou l'infirmation de l'hypothèse.) et en faisant la distinction entre la responsabilité ex-ante et ex-post action, on peut améliorer la fonction signal de responsabilité :

$$\mu_i(t+1) = \sum \lambda_j(t)\pi(i \mid j, C)$$

$$\lambda_i(t) = \frac{\gamma_i \mu_i(t)}{\sum \gamma_j \mu_j(t)}$$

avec λ_i le signal de responsabilité à l'instant t pour le task-set i, $\pi(i/j,C)$ la probabilité que le task-set i soit valide à l'instant t+1 sachant que le task-set j était valide à l'instant t et que le contexte C est celui à l'instant t+1. Ainsi, on retrouve dans la formule l'influence du passé (acceptabilité de l'action au temps t) et du contexte futur (probabilité d'acceptabilité au temps t+1), pondérée par le signal de responsabilité. Dans cette configuration, nous sommes toujours dans un mécanisme markovien d'ordre 1 où seul le temps t nous intéresse vis-à-vis du temps t+1.

Ainsi le signal de responsabilité ex-post action sert à renforcer (ou diminuer) le choix et le signal ex-ante action sert à déterminer les actions par défaut associées au contexte.
Ce modèle fait aussi de la création, stockage, changement et récupération de TS pour contrôler le comportement en fonction des indices de l'environnement et du feed-back qu'il renvoie. Cette fonction de contrôle contextuel est une fonction exécutive découverte dans notre cerveau au niveau du cortex préfrontal latéral postérieur.

Si on généralise ce modèle, on peut l'associer à la mémoire épisodique de l'hippocampe en présentant une notion étendue des jeux d'actions, appelé jeux d'épisodes, possédant un pattern enregistré de jeux d'actions pour un contexte épisodique particulier. Cependant les patterns changent en fonction de l'épisode donc il faut un niveau plus haut de fonctions exécutives pour gérer les jeux d'épisodes, au niveau du cortex pré-frontal. Le modèle de base proposé au-dessus sur les ensembles d'action (fonctionnant par a priori et feed-back) peut être dupliqué à un niveau plus élevé pour être appliqué aux jeux d'épisodes. On suppose alors que l'agent est partiellement hypermnésique et va pouvoir retenir des séquences de précédentes entrées qu'il a eu (tous les stimulus avec leurs récompenses), on quitte le modèle Markovien utilisé dans le modèle usuel d'apprentissage

par renforcement. Un tel contrôle épisodique est d'ailleurs aussi une fonction exécutive découverte dans notre cortex préfrontal latéral antérieur. C'est un sujet de recherche ouvert.

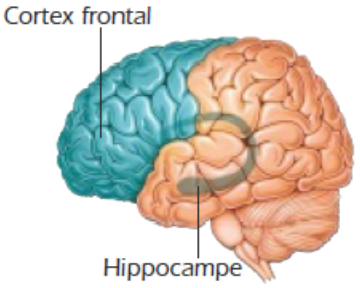


FIGURE 2 : Schéma de l'hippocampe et du cortex frontal. De récentes études ont montré qu'il existe un lien entre l'hippocampe (siège de la mémoire épisodique) et le cortex préfrontal (responsable de certaines fonctions exécutives) au niveau de la résolution contextuelle de problèmes, d'après[9] (Isingrini et Taconnat 2008).

Le formalisme considéré pour notre approche a vocation à s'appliquer justement à la modélisation des fonctions exécutives de décisions contextuelles dans le cadre de résolution de problèmes, et tenir compte du fait qu'on représente des séquences de données multui-modales structurées.

Pour limiter l'explosion de complexité qui serait liée à l'énumération[10] de tous les états possibles nous allons utiliser une représentation de données structurées, munie de relations hiérarchiques, sous forme d'ontologie[11]. Ainsi, la représentation restreint le nombre de données explicitement écrites sans pour autant empêcher les informations implicites (héritage, similarité…) d'enrichir les données initiales.

3 L'approche dite "symboling" en résolution de problèmes

3.1 Modélisation hiérarchique

La structure ressemble à celle d'un dictionnaire (ou tuple) avec pour chaque variable un nom et une valeur. Le type de chaque valeur est prédéterminé par un schéma préconçu et est de plusieurs types : un étiquette, une valeur numérique, une liste ordonnée (ou séquence), un ensemble non-ordonné ou un sous-dictionnaire, rendant cette structure hiérarchique. Le type peut aussi être spécifié de manière plus spécifique, par exemple une valeur littérale parmi une liste prédéterminée, un nombre borné et de précision finie ou même la valeur "undefined" ce qui laisse en attente pour pouvoir y remplir une valeur une fois qu'elle sera définie. Une telle structure est exemplifiée figure 3.:

[9] Isingrini, M., & Taconnat, L. (2008). Mémoire épisodique, fonctionnement frontal et vieillissement Episodic memory, frontal functioning, and aging. Revue neurologique, 164, S91-S95.

[10] Supposons que la représentation interne se fasse à travers un séquence de longueur T de N variables qui prennent M valeurs, on devrait énumérer M^N à chaque instant soit $(M^N)^T$ valeurs ! Or dans en apprentissage par renforcement on doit théoriquement construire une table de transitions pour toutes ces valeurs (ou l'approximer par un calcul numérique très couteux comme un réseau de neurones profond).

[11] Voir https://fr.wikipedia.org/wiki/Ontologie_(informatique) ou https://interstices.info/ontologies-informatiques pour une introduction pluri-disciplinaire.

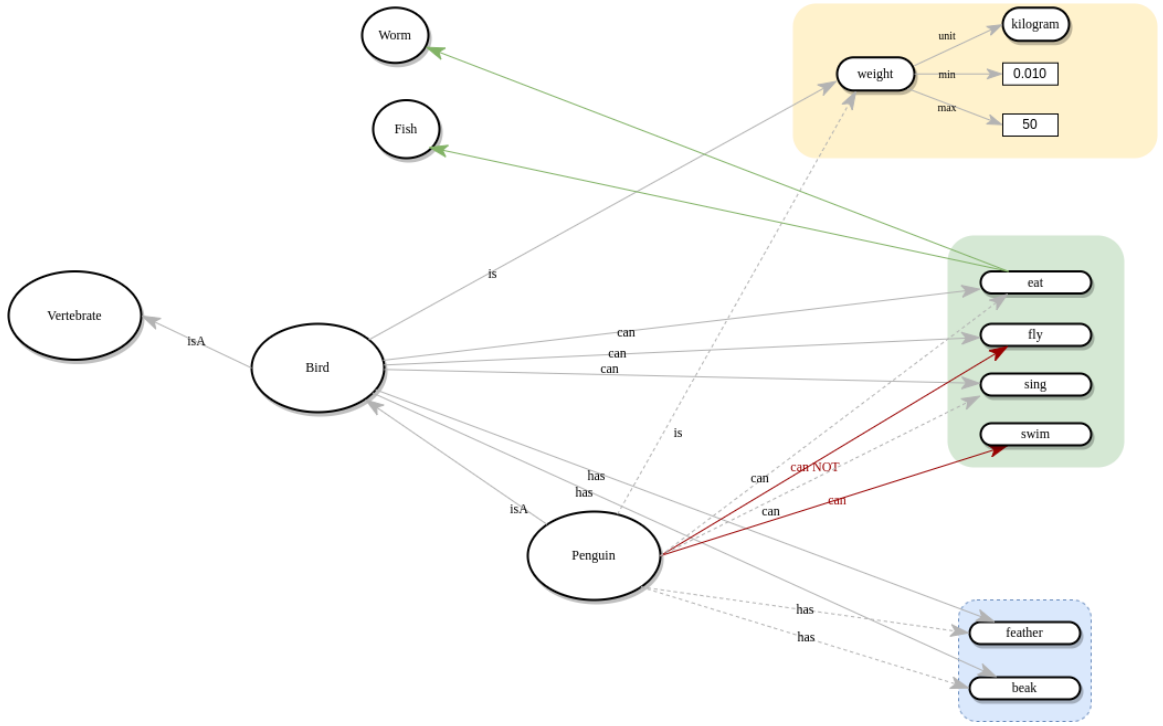


FIGURE 3 : Exemple de données structurées modélisant nos connaissances à propos du concept d'oiseau. Ces concepts sont batis à partir d'information sensori-motrice en lien avec l'objet concret. Dans notre contexte, chaque donnée contient des informations sur la tâche mais aussi sur l'apprenant. Figure proposée par Chloé Mercier.

Les valeurs peuvent être complétées par des méta-informations, qui ne sont pas directement utilisées par l'agent mais servant pour la spécification ou l'interprétation. Les données peuvent ensuite être représentées en syntaxe[12] JSON :

```
bird: {
  is_a: vertebrate
  can: { sing fly eat: { worm fish } }
  has: { feather beak }
  is: { weight : { min: 0.010 max: 50 unit: kilogram } }
}
```

La structure sémantique choisie est la représentation[13] de (McClelland & Rogers 2003) et permet d'expliciter des connaissances sémantiques par relation hiérarchique (est un) avec des capacités (peut), des ressources extrinsèques (possède) et des ressources intrinsèques (est). On nomme ces données des "données symboliques". Les propriétés peuvent être qualitatives et quantitatives. Une telle structure est appelée "type" ou "concept" tel qu'il est défini[14] par (Gardenfors, 2004). Pour reprendre l'exemple ci-dessus, le type "oiseau" est un "vertébré" qui peut "chanter", "voler", "manger" (et manger d'autre types !), il possède des "plumes" et un "bec" et a un "poids" minimum et maximum.

[12] La syntaxe utilisée est celle dite wJSON pour "weak JSON" (voir la section idoine).
[13] McClelland, J. L., & Rogers, T. T. (2003). The parallel distributed processing approach to semantic cognition. Nature reviews neuroscience, 4(4), 310-322.
[14] Gärdenfors, P. (2004) Conceptual Spaces as a Framework for Knowledge Representation. Mind and Matter 2(2):9-27.

Un type est à la fois une région convexe[15] et un prototype dont chacune des caractéristiques possède une valeur par défaut, par-dessus lesquelles on peut réécrire lorsqu'on définit un objet de ce type. Par exemple pour le pingouin qui est un oiseau particulier :

```
penguin: {
 is_a: bird
 can: { fly: false walk }
 }
```

Plusieurs recommandations permettent de spécifier au mieux de telles connaissances :
- Décomposer le plus possible les caractéristiques pour que les valeurs soient atomiques;
- Organiser le plus possible les informations en arbre et sous-arbres plutôt que de tout avoir au même niveau (cf. complexité);
- Renseigner le plus possible des valeurs par défaut;
- Utiliser des noms explicites ou standardisés pour les variables.

Au delà, il est possible d'évoluer dans un contexte supérieur comme définir des règles (de la même manière que l'on définit des types) en définissant en variables une précondition (un état spécifique en accord avec un schéma défini au préalable comme "avoir faim") et une post-condition (une séquence de modification de l'état actuel comme "manger" avec toutes les sous étapes qu'elle suppose).
On peut définir des types incomplets, comme des connaissances incomplètes. On peut aussi définir des connaissances dont le niveau de vérité est approximatif, autrement dit, des croyances !

Au niveau du raisonnement ontologique, chaque tuple est un sujet relié à un autre par une propriété, permettant ainsi de faire des inférences en termes de propriétés en déduisant des relations ontologiques déjà existantes. Ces inférences permettent de décrire une connaissance, voire un comportement de manière symbolique. Au niveau syntaxique[16], il est est aussi possible de faire le chemin inverse en transformant les données symboliques en tuple.

3.2 Introduction d'une métrique

Le point clé est que nous allons introduire une distance d'édition entre deux structures hiérarchiques. Si on considère deux structures la distance d'édition[17] est le nombre minimal d'opérations d'édition pour transformer une structure en une autre, avec des opérations de modification, d'insertion ou de suppression de valeurs, ou dans notre cas hiérarchique de sous-valeurs. On considère que chaque opération n'a pas la même importance, selon la caractéristique qui est modifiée dans la structure. Ainsi,

[15] En géométrie, le concept de région convexe fait référence à un espace où, quelque soient deux points de cet espace, le trajet pour les relier est toujours à l'intérieur de cet espace.
Dans un espace "type", cela signifie que pour passer d'une donnée initiale à une donnée finale, chaque étape est un objet appartenant à l'espace (par exemple, la trajectoire pour passer d'une hirondelle à un merle dans l'espace "oiseau", chaque étape intermédiaire est un spécimen du type "oiseau").Dans un espace "type", cela signifie que pour passer d'une donnée initiale à une donnée finale, chaque étape est un objet appartenant à l'espace (par exemple, la trajectoire pour passer d'une hirondelle à un merle dans l'espace "oiseau", chaque étape intermédiaire est un spécimen du type "oiseau").
[16] Voir https://line.gitlabpages.inria.fr/aide-group/wjson/turtoise.pdf pour une description détaillée.
[17] Voir https://fr.wikipedia.org/wiki/Distance_de_Levenshtein pour une présentation détaillée dans le cas de séquences.

chaque opération possède un coût qui permet de pondérer la transformation, et offre donc une grande flexibilité dans l'approche.

Au niveau de la spécification nous allons considérer que chaque valeur a un type, donc définir un modèle d'objet, et chaque type définit un espace métrique sur ces valeurs, c'est-à-dire qu'il contient la notion de distance entre les objets.
Pour qu'une distance entre deux objets symboliques soit valide, les deux doivent se trouver dans l'espace sémantique définie par ce type.
Cependant, toutes les données récupérées ne sont pas forcément dans cet espace, on a donc aussi besoin de définir un projecteur[18] sur cet espace. Et ce projecteur doit être défini lui aussi sur un type de données qui englobe le type sur lequel on souhaite projeter les données.
Autrement dit, une région au sein de l'espace défini par un type est définie par un sous-type. Et on considère alors qu'il possède un projecteur pour passer d'un point de ce type à une région définie par un sous-type.
Les distances peuvent ainsi se calculer entre deux objets, soit sur les objets eux-mêmes s'ils sont déjà sémantiquement valides, soit sur leur projection dans l'espace sémantique.

On va parler d'espace syntaxique pour définir le sur-type qui permet de faire la projection et d'espace sémantique pour définir le type sur lequel on calcule des distances. Un projecteur syntaxique permet de projeter n'importe quelle valeur dans l'espace syntaxique (souvent en prenant la valeur par défaut, d'où son utilité). Le projecteuir sémantique de trouver une valeur avec laquelle on peut calculer la distance.

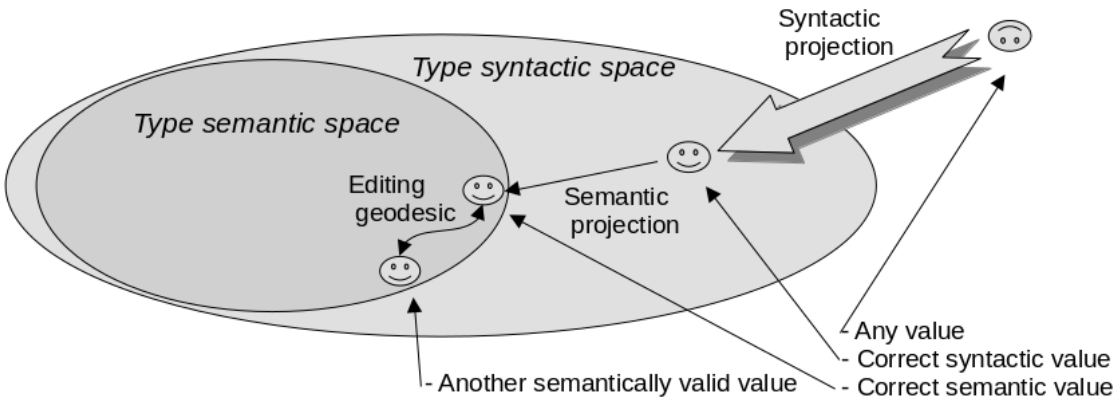


FIGURE 4 : Illustration des mécanismes liés à un type et un sous-type. On voit qu'il faut que deux éléments soient dans l'espace sémantique pour que l'on puisse mesurer la distance qui les sépare. Il existe ainsi deux projecteurs pour pouvoir effectuer une mesure de distance sur leur projection si les objets n'appartiennent pas à l'espace sémantique.

Pour illustrer les choses, considèrons le type "entier positif". Une valeur (comme "10") est syntaxiquement valide si, une fois convertie, c'est une valeur numérique ("10" est bien convertible en 10 !) et est sémantiquement valide si c'est bel et bien un entier positif (10 serait donc sémantiquement valide car il est positif, et c'est un entier !). Si ce n'était pas le cas, par exemple avec la valeur -10, on le projetterait sur la valeur la plus proche, soit 0 dans le cas des distances usuelles.

[18] Un projecteur d'un point sur une région calcule la valeur qui appartienne à la region est de distance minmale, la plus proche, par rapport au point. Il s'avère que au niveau applicatif on peut facilement spécifié un tel calcul pour les données considérées.

Grâce à ces ingrédients on peut calculer la distance d'édition entre deux structures typées quelconques. En fait on contraint la distance d'édition à respecter le type des valeurs (c'est à dire qu'une valeur ne peut pas être modifiée en n'importe quelle autre valeur de n'importe quel type, mais uniquement une valeur de même type). Une telle restriction permet d'assurer la cohérence des opérations d'éditions, mais aussi de permettre d'utiliser les valeurs intermédiaires générées au fur et à mesure de l'application des opérations d'édition. De ce fait on n'obtient pas uniquement la distance entre deux structures mais le *chemin* à l'intérieur de l'espace défini par le type qui permet de passer d'une structure à une autre. Mieux encore ce chemin est une géodésique[19], c'est à dire un chemin de longueur minimale en matière de distance.

3.3 Calcul effectif de la métrique

Il existe des algorithmes permettant de calculer une distance d'édition entre deux objets. Ceux que nous manipulons ici sont l'algorithme Hongrois et l'algorithme de Levenshtein.

L'algorithme Hongrois (ou algorithme de Kuhn) s'applique pour les structures non ordonnées. Considérons une structure non ordonnée initiale et une deuxième structure non ordonnée finale. Le but de cet algorithme est de trouver le chemin de modifications pour passer de l'un à l'autre ayant un coût minimal de transformation. Pour être plus précis, on peut imaginer que pour passer de la première à la deuxième, il est possible de faire différents chemins, c'est-à-dire des séquences de modifications différentes, mais dont les coûts d'édition sont différents.

L'algorithme de Levenshtein lui, s'applique sur des structures ordonnées comme des chaînes de caractères ou des arbres. Le principe est de sélectionner élément par élément l'opération au coût minimal pour au total passer d'une structure ordonnée A à B. Chaque opération peut être un ajout, une suppression ou une modification. A la différence de l'algorithme Hongrois, il faut respecter l'ordre de lecture de la structure. Considérons les listes implicitement ordonnées L = ["un peu", "beaucoup", "*passionnément*", "à la folie"] et M = ["un peu", "beaucoup", "*presque*", "à la folie"]. Pour déterminer un distance d'édition pour passer de la liste L à M selon l'algorithme de Levenshtein, il nous faut respecter l'ordre de considération implicite des termes (modifier "un peu" puis "beaucoup", etc...).

Au delà, pour une structure arborescente la situation est plus complexe. Si on considère nos données sous forme d'arbres c'est-à-dire qu'il existe seulement une relation ancêtre-descendant entre les nœuds. On montre que pour passer de l'arbre A à l'arbre B,, si aucun ordre n'était appliqué et qu'il était possible de modifier les parents indépendamment des enfants et vice-versa, les possibilités de modifications pour passer d'une structure à une autre augmenteraient de manière exponentielle. Les combinaisons de trajectoire seraient à la fois multiples et parfois incohérentes. En effet, dans notre contexte, nous nous intéressons à la résolution de problème par modifications atomiques de l'état du problème. Autrement dit, résoudre un problème depuis une situation initiale se fait par petites étapes qui suivent un ordre logique qui ne détruisent pas la structure logique du problème manipulé.
Pour pallier ce problème de complexité et de logique, une solution est d'utiliser des contraintes de modifications telles que deux nœuds parents-enfants ne peuvent être échangés allié à des contraintes

[19] Voir https://line.gitlabpages.inria.fr/aide-group/symboling/scalarfield.pdf pour une démonstration et une utilisation pour traiter de telles donnés symboliques au niveau numérique. Assez simplement sur une géodésique entre deux valeurs A et B les points C de la géodésiques vérifient:

distance(A, B) = distance(A, C) + D(C, B)

alors que en général il n'y a qu'une inégalité, dite triangulaire.

d'ordre[20] (Ouangraoua et Ferraro, 2009). On peut aussi utiliser des arbres semi-ordonnés, c'est-à-dire que tous les noeuds de l'arbre vérifient une relation semi-ordonnée (pour deux noeuds x et y, la relation x-y est transitive et réflexive) avec son noeud-parent ou noeud-enfant en plus de la relation ancêtre-descendant. De par ces restrictions, les possibilités de séquences de modifications diminuent. Dans cette configuration, la complexité algorithmique est de l'ordre polynomial.
Lorsque nous sommes dans des cas de structure où un ordre est sous-jacent, les feuilles de notre arbre s'ordonnent (les nœuds-enfants s'ordonnent) et on peut appliquer l'algorithme de Levenshtein (voir "I.d. Symboling, métrisation de l'espace").

Ici la contrainte sur le typage des valeurs implique la préservation de la généalogie (deux nœuds parents-enfants ne peuvent être échangés) et on retombe sur un mécanisme très simple qui fait qu'au niveau des tuples la complexité est linéaire.

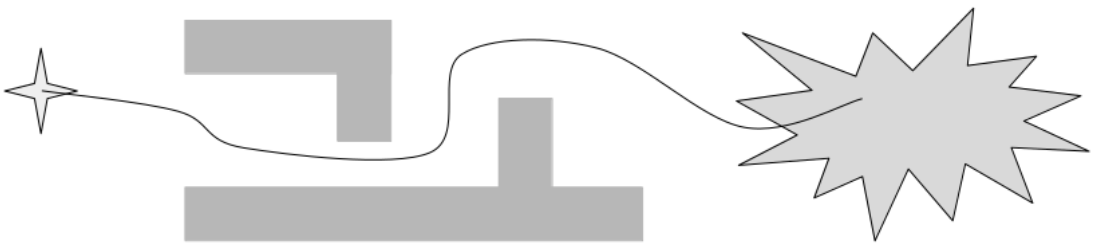
3.4 Application à la résolution de problèmes

Maintenant le cadre défini, nous pouvons passer à la modélisation du problème qui est justement "comment modélise-t-on la résolution de problème?"

Selon la définition[21] de (Newell et Simon, 1972), une situation de résolution de problèmes se caractérise par un ensemble de structures symboliques (les objets de l'espace en question) et un ensemble d'opérateurs sur ces structures. Ces opérateurs prennent en entrée des états et sortent des états. Il se peut que les opérateurs ne s'appliquent pas systématiquement sur tous les états. En effet, ces derniers peuvent ne pas se trouver dans le domaine de définition des opérateurs (par exemple, il se peut que l'on ai besoin de connaître la distance entre deux objets mais l'un n'est pas dans le bon espace. Par conséquent, l'opération de distance n'est pas applicable dessus.).

Une séquence d'opérations définit un chemin entre les différents états dans l'espace d'état (une opération permettant de passer d'un état à un autre).

Définir un problème de manière spatialisée comme ici consiste à définir un ensemble d'état initiaux, d'états finaux, et un ensemble de contraintes sur le chemin. Résoudre un problème signifie donc ici trouver un chemin débutant à n'importe quel état initial défini, satisfaisant les contraintes spatiales et finissant dans n'importe quel état final (qu'il soit unique ou non, partiellement défini ou non). Dans ce mécanisme de recherche de solution par trajectoire, il faut explorer l'espace afin de résoudre le problème



. FIGURE 5 : Représentation de la résolution de problème par approche géométrique. Le problème débute à un point initial de l'espace (représenté par l'étoile à quatre branches) et doit évoluer de sorte à se frayer un chemin en respectant certaines contraintes (représentés ici par des obstacles en gris) jusqu'à l'espace correspondant à un but d'arrivée (symbole "explosion" sur la figure).

[20]Ouangraoua, A., Ferraro, P.: A Constrained Edit Distance Algorithm Between Semiordered Trees. Theoretical Computer Science 410(8-10), 837–846 (2009), publisher: Elsevier

[21] Newell, A., & Simon, H. A. (1972). Human problem solving (Vol. 104, No. 9). Englewood Cliffs, NJ: Prentice-hall.

La spécificité dans notre algorithme est que chaque point de l'espace est une structure symbolique de données, contenant les informations relatives à l'agent (l'objet lui-même) et à l'espace dans lequel il est (sa position relative). Pour refaire le lien avec l'apprentissage et la résolution de problème spatial, chaque structure hiérarchisée nous informe sur l'apprenant (son état) et la direction qu'il entreprend dans sa résolution (l'apprentissage).

Trouver une trajectoire dans cet espace, i.e. résoudre un problème revient à modifier pas à pas la structure de la donnée depuis son état initial jusqu'à arriver à l'état final désiré. L'ingrédient essentiel pour réaliser cela est de concevoir une structure qui peut être métrisable, c'est-à-dire qu'on puisse déterminer une distance. C'est un critère primordial pour, par exemple, la distance entre la donnée d'entrée et son objectif, autrement dit pouvoir quantifier non seulement le chemin parcouru mais aussi celui restant. Il faut aussi pouvoir quantifier des distances intermédiaires, par exemple la distance aux obstacles à éviter.
Ajouter bout à bout les modifications permet de déterminer notre suite d'états formant la trajectoire de résolution.
Notons que pour cette partie, on considère que les données sont sémantiquement valides pour pouvoir y appliquer une notion de distance entre elles.

4 Expérimentation du logiciel à évaluer

Dans cette partie on évalue une implémentation de ces principes sous forme une bibliothèque logicielle[22] en cours d'évaluation.

4.1 Utilisation de la syntaxe wJSON

Une première remarque s'impose vis-à-vis des choix de syntaxe. Afin d'avoir une accessibilité souple pour le programme, un des choix à été de proposer d'utiliser du "weak JSON"[23] plutôt que du JSON classique. En effet, ce dernier possède une plus grande flexibilité syntaxique, permettant à une plus grande part de population plus ou moins à l'aise en programmation de participer au projet ou tout du moins de le comprendre.

[22] Voir https://line.gitlabpages.inria.fr/aide-group/symboling pour une documentation détaillée, avec les guides d'installation en fonction de votre système d'exploitation:
- Installation sous Linux et MacOS : https://line.gitlabpages.inria.fr/aide-group/aidebuild/install.html
- Installation sous Windows : https://gitlab.inria.fr/line/aide-group/aidebuild/-/blob/master/src/install_on_windows.md

[23] Voir https://line.gitlabpages.inria.fr/aide-group/wjson pour une présentation détaillée et l'accès au logiciel et aux sources.

```
{
  first-name: Jean-Pierre
  last-name: Pierrejean
  age: 107
  address: "314 Pi road, Quadrature city"
  friends: [ him, her, "somebody else" ]
  imaginative
  presentation: "
   Under the moonlight
   Write a word in white
   Please borrow my pen
   We are all nice men"
}
```

FIGURE 6 : Exemple de syntaxe wJSON

On retrouve la même présentation en dictionnaire ou "objet" du JSON, peut-être un peu plus intuitive à lire. C'est au niveau de l'écriture que la différence se joue. La présentation en dictionnaire avec un simple retour à la ligne, un nom de variable et ce qu'elle contient est une manière plus intuitive de construire un objet, surtout pour une personne non initiée. Une seule précaution à prendre est d'ajouter un espace après les deux points. Il n'est pas non plus nécessaire de remplir tous les champs. Dans l'exemple ci-dessus, "imaginative" est laissé seul. Pour une personne programmeuse, cela signifie que la variable est à true, quant à une personne non-programmeuse, cela signifie simplement que Jean-Pierre est imaginatif !

4.2 Démonstrations illustratives

Deux démonstrations ont été réalisées afin d'illustrer le traitement de la distance d'édition sur des données non ordonnées (algorithme hongrois) et ordonnées (algorithme de Levenshtein).

4.2.1 Démonstration graphique : "La tête à Toto vers la tête à Titi"

Dans cette partie nous nous proposons d'appliquer symboling à une résolution graphique (sur structure non ordonnée). L'idée est de passer du dessin A au dessin B par une suite de modification de structure :

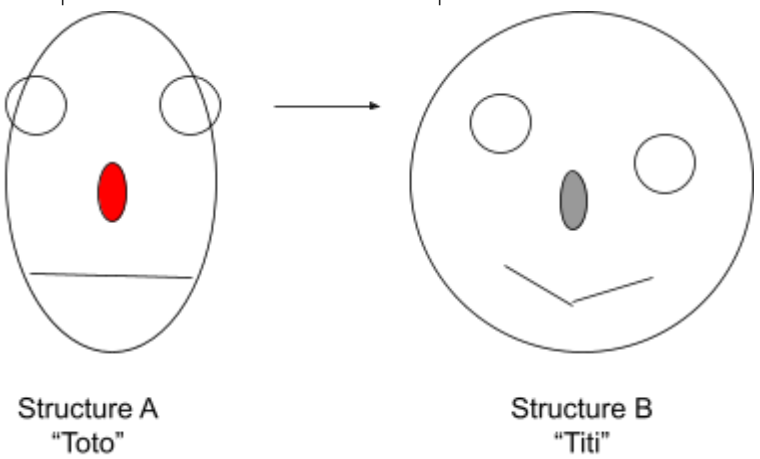


FIGURE 7 : Structure SVG initial "Toto" et finale "Titi"

Les deux dessins ont été générés sous format SVG, utilisé notamment dans le Web pour la représentation des images sous forme vectorielle. Un programme de conversion permet de passer d'une pure représentation SVG à une représentation symbolique en JSON, qui est compatible avec le format wJSON.

Le logiciel écrit en C++, exploitant la bibliothèque symboling, convertit donne une représentation symbolique, un "sens" aux éléments composants la forme JSON des visages Titi et Toto.

Cette démonstration est intégrée à la bibliothèque et on peut accéder:
- à la description au format symboling d'un sous ensemble de la spécification SVG[24];
- à la documentation de son utilisation logicielle[25];
- et à une page de démonstration[26].

4.2.2 Démonstration musicale : la distance entre deux musiques

Dans cette partie l'idée est d'appliquer le programme Symboling à des partitions de musiques. Le programme appliqué à notre partition écrite sous forme JSON va être modifiée jusqu'à donner la partition choisie.
Nous avons tout d'abord réalisé un intergiciel permettant de passer du JSON au midi et inversement. Pour cela, nous nous appuyons sur un package npm existant et avons développé un moyen de l'utiliser directement depuis la console.
Cependant, la structure JSON du midi est complexe et donc difficile à traduire avec les classes symboling, on a donc a ce jour une preuve de concept, mais un mécanisme partiel.

Cette démonstration est intégrée à la bibliothèque et on peut accéder:
- à la description au format symboling d'un sous ensemble de la spécification MIDI[27];
- à la documentation de son utilisation logicielle[28];
- et à une page de démonstration[29].

5 Conclusion

Cette approche du programme Symboling à des cas concrets a permis plusieurs choses: premièrement à réaliser une centralisation des différents concepts sur lesquels reposent le projet, puis à la réalisation d'une documentation pour l'installation sur différents environnements, et enfin à des démonstrations visuelles et concrètes du programme. Pour résumer, cette étape a permis de formaliser et concrétiser la résolution de problème par approche géométrique grâce à la distance d'édition comme l'entend le programme symboling et d'évaluer et valider la version logicielle initiale de l'implémentation, en permettant de remonter des corrections importantes.

Au delà, en reprenant le cadre initial du projet, à savoir un agent dans un monde représenté et relié par ontologie, on pourrait appliquer ce travail à l'apprentissage par renforcement, donc fournir au modèle PROBe une généralisation assez puissante. Plus précisément, l'agent dans son monde évolue en apprenant comment il doit interagir avec les éléments de son environnement en fonction de son expérience passée avec celui-ci mais aussi des inférences qu'il fait vis-à-vis de lui. L'agent est par ailleurs,

[24] https://gitlab.inria.fr/line/aide-group/symboling/-/blob/master/src/SvgType.cpp
[25] https://line.gitlabpages.inria.fr/aide-group/symboling/SvgType.html
[26] https://line.gitlabpages.inria.fr/aide-group/symboling/visualmorphing/index.html
[27] https://gitlab.inria.fr/line/aide-group/symboling/-/blob/master/src/MidiType.cpp
[28] https://line.gitlabpages.inria.fr/aide-group/symboling/MidiType.html
[29] https://line.gitlabpages.inria.fr/aide-group/symboling/musicmorphing/index.html

avec la notion de distance, capable de relier, voir reconnaître un objet similaire à un autre déjà rencontré par le passé grâce à des caractéristiques partagées (caractéristiques sous-entendu dans l'ontologie). Une perspective à ce projet pourrait être de fusionner les deux approches. En effet, d'un côté nous possédons un programme capable de relier un état initial à un état-objectif et de l'autre un programme simulant un agent apprenant de son environnement par exploration/récompense. On pourrait envisager la création d'un programme dans lequel l'agent possède un état de départ et un état d'arrivée et qu'il doive passer de l'un à l'autre grâce à des éléments de son environnement qu'il découvre et reconnaît. Par exemplet, cela revient à laisser un naufragé ignorant sur une île dont l'objectif est de survivre et reprendre la mer : l'état initial peut être "sans maison" et l'agent doit chercher et reconnaître le bois pour atteindre l'état final "abrité" :)

6 Contributions

Le travail bibliograhique a été realisé par Morgane Laval, et le travail d'évaluation du logiciel et de conception des démonstrations par Paul Bernard pour la démonstration graphique et Benjamin Hate pour la démonstration musicale, sachant que le travail est dans son ensemble avant tout un travail d'équipe.

Ce travail a été co-encadré par Axel Palaude et Chloé Mercier, tous deux en thèse dans l'équipe Mnemosyne avec l'aide de Thierry Viéville.

7 Références

Collins, A., & Koechlin, E. (2012). Reasoning, learning, and creativity: frontal lobe function and human decision-making. PLoS biology, 10(3).

Isingrini, M., & Taconnat, L. (2008). Mémoire épisodique, fonctionnement frontal et vieillissement Episodic memory, frontal functioning, and aging. Revue neurologique, 164, 91-955.

Gärdenfors, P. (2004) Conceptual Spaces as a Framework for Knowledge Representation. Mind and Matter 2(2):9-27.

Lambropoulos, N., & Romero, M. (2015). 21st Century Lifelong Learning: Individual, Team and Social skills and competence-based methodologies. Nova Publishers.

Newell, A., & Simon, H. A. (1972). Human problem solving (Vol. 104, No. 9). Englewood Cliffs, NJ: Prentice-hall.

Ouangraoua, A., Ferraro, P.: A Constrained Edit Distance Algorithm Between Semiordered Trees. Theoretical Computer Science 410(8-10), 837–846 (2009), Elsevier.

Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

RESEARCH CENTRE
BORDEAUX - SUD-OUEST

351 Cours de la Libération
Bâtiment A29
33405 Talence Cedex France

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr
ISSN 0249-6399