Découverte de sous-groupes avec les arbres de recherche de Monte Carlo

Guillaume Bosc, Jean-François Boulicaut, Chedy Raïssi, Mehdi Kaytoue

Découverte de sous-groupes avec les arbres de recherche de Monte Carlo

Guillaume Bosc*, Jean-François Boulicaut* Chedy Raïssi**
Mehdi Kaytoue*

*Université de Lyon, CNRS, INSA-Lyon, LIRIS, UMR5205, F-69621, France
**INRIA Nancy - Grand Est, Villers-lès-Nancy, F-54600, France
prenom.nom@insa-lyon.fr, chedy.raissi@inria.fr

Résumé. Découvrir des règles qui distinguent clairement une classe d'une autre reste un problème difficile. De tels motifs permettent de suggérer des hypothèses pouvant expliquer une classe. La découverte de sous-groupes (Subgroup Discovery, SD), un cadre qui définit formellement cette tâche d'extraction de motifs, est toujours confrontée à deux problèmes majeurs: (i) définir des mesures de qualité appropriées qui caractérisent la singularité d'un motif et (ii) choisir une heuristique d'exploration de l'espace de recherche correcte lorsqu'une énumération complète est irréalisable. À ce jour, les algorithmes de SD les plus efficaces sont basés sur une recherche en faisceau (Beam Search, BS). La collection de motifs extraits manque cependant de diversité en raison de la nature gloutonne de l'exploration. Nous proposons ici d'utiliser une technique d'exploration récente, la recherche arborescente de Monte Carlo (Monte Carlo Tree Search, MCTS). Le compromis entre l'exploitation et l'exploration ainsi que la puissance de la recherche aléatoire permettent d'obtenir une solution disponible à tout moment et de surpasser généralement les approches de type BS. Notre étude empirique, avec plusieurs mesures de qualité, sur divers jeux de données de référence et du monde réel démontre la qualité de notre approche.

1 Introduction

L'extraction de groupes d'objets caractéristiques d'un attribut de classe a été intensément étudiée en fouille de données (Novak et al., 2009). La découverte de sous-groupes (Subgroup Discovery, SD) est une instance de ce problème (Wrobel, 1997). Étant donné un ensemble d'objets décrits par des attributs et associés à un ou plusieurs labels de l'attribut de classe, un sous groupe est un sous ensemble d'objets respectant une description sur les attributs. Le caractère discriminant d'un sous groupe est évalué par une mesure de qualité (F1 mesure, précision, etc.). Jusqu'à présent, puisque la taille de l'espace de recherche est exponentielle, les algorithmes les plus efficaces en SD sont basés sur une recherche en faisceau (Beam Search, BS) (van Leeuwen et Knobbe, 2012; Meeng et al., 2014; Duivesteijn et al., 2016).

Les problèmes principaux des approches heuristiques en SD sont (i) le manque de diversité des motifs extraits et (ii) la redondance : (i) une faible partie des optimums locaux de l'espace de recherche sont détectés, et (ii) plusieurs sous-groupes sont similaires à un même

optimum local détecté. Alors que plusieurs solutions ont été proposées pour éliminer la redondance (van Leeuwen et Knobbe, 2012), aucune recherche n'a été menée à notre connaissance pour concevoir de nouvelles méthodes d'exploration heuristique visant à améliorer la diversité. L'exploration gloutonne effectuée par BS ne permet pas la découverte de nombreux optimums locaux (diversité) mais favorise en plus une redondance importante dans les motifs calculés.

La recherche arborescence de Monte Carlo (Monte Carlo Tree Search, MCTS) est une méthode d'exploration (Browne et al., 2012) qui parcourt partiellement l'espace de recherche en construisant incrémentalement un arbre asymétrique respectant le compromis exploration / exploitation donné par le calcul de la limite supérieure de confiance (Upper Confidence Bound, UCB) (Kocsis et Szepesvári, 2006). MCTS est basée sur des simulations aléatoires qui explorent l'espace de recherche. Ainsi, l'expansion de l'arbre dépend des gains obtenus durant les simulations pour à la fois exploiter des solutions intéressantes, mais aussi explorer des zones peu visitées de l'espace de recherche. MCTS a principalement été utilisée en Intelligence Artificielle pour des applications qui peuvent être représentées par des arbres de décisions séquentielles, par exemple les jeux ou les problèmes de planification. Cette exploration s'est montrée particulièrement efficace pour le jeu de Go (e.g., l'équipe AlphaGo de Google) pour lequel aucune fonction d'évaluation heuristique d'un état du jeu n'est connue.

Notre contribution principale est de montrer comment adapter efficacement MCTS pour SD afin d'améliorer la diversité par rapport à une approche de type BS. Puisque MCTS est conçue pour être capable de gérer de grands facteurs de branchement, nous pensons qu'il peut être capable d'augmenter la diversité dans les motifs extraits en détectant un plus grand nombre d'optimums locaux. MCTS permet également d'obtenir des résultats à tout instant. A notre connaissance, il s'agit de la première tentative d'utiliser MCTS en fouille de motifs. MCTS diffère des techniques d'échantillonnage, qui permettent elles-aussi d'obtenir des résultats à tout instant. Ces dernières considèrent une loi de distribution sur l'espace des motifs qui donne plus de chance à un motif intéressant (i.e., avec une bonne mesure de qualité) d'être tiré. Cependant cette loi de distribution doit être définie au préalable par l'utilisateur en fonction de la structure des motifs et de la mesure de qualité utilisée (Moens et Boley, 2014). D'autres travaux considèrent une exploration interactive de l'espace de recherche pour améliorer la diversité. Galbrun et al. propose à l'utilisateur de guider l'exploration en faisceau (Galbrun et Miettinen, 2012). De même, Dzyuba et al. utilise l'apprentissage des préférences de l'utilisateur pour rendre l'algorithme DSSD interactif (Dzyuba et al., 2014). Cependant, l'exploration restant de type BS, la diversité reste insuffisante.

La suite de cet article est organisée comme suit. Les Section 2 et Section 3 présentent respectivement les bases de SD et MCTS. La Section 4 explique comment adapter MCTS à SD et met en évidence les différentes stratégies pouvant être utilisées. Les résultats expérimentaux sont discutés en Section 5. Une version étendue de ce travail est disponible (Bosc et al., 2016b).

2 Découverte de sous-groupes

Soient $\mathcal{O}$, $\mathcal{A}$ et C respectivement un ensemble d'objets, un ensemble d'attributs et un attribut de classe nominal. Le domaine d'un attribut $a \in \mathcal{A}$ est $Dom(a)$ où a est numérique si $Dom(a)$ est pourvu d'un ordre total, nominal sinon. Chaque objet est associé à un label parmi $Dom(C)$ par $class : \mathcal{O} \mapsto Dom(C)$. $\mathcal{D}(\mathcal{O}, \mathcal{A}, C, class)$ est un jeu de données. Par souci de simplicité et de lisibilité, nous nous plaçons ici dans le cadre du SD traditionnel où il n'y a

qu'un seul attribut de classe (Wrobel, 1997). Cependant, la méthode Exeptional Model Mining (EMM) généralise SD dans le cas où il y a plusieurs attributs de classe (Leman et al., 2008). La transposition de ce travail à EMM est simple et immédiate.

Définition : Sous-groupe. Un sous-groupe est donné par sa description $d = \langle f_1, \ldots, f_{|\mathcal{A}|} \rangle$ où chaque f_i est une restriction sur le domaine de valeurs de $a_i \in \mathcal{A}$. Une restriction est soit un sous ensemble du domaine d'un attribut nominal, soit un intervalle inclus dans le domaine de définition d'un attribut numérique. L'ensemble des objets couverts par une description d est le support du sous-groupe $supp(d) \subseteq \mathcal{O}$. L'ensemble de tous les sous-groupes forme un treillis.

La mesure de qualité φ utilisée évalue la singularité d'un sous-groupe par rapport à C. Le choix de la mesure de qualité dépend de l'application (Fürnkranz et al., 2012). Différentes mesures de qualité peuvent être utilisées comme la mesure F1, WRAcc, ou la divergence KL.

Le problème de SD. Étant donné un jeu de données $\mathcal{D}(\mathcal{O}, \mathcal{A}, C, class)$, $minSupp$, φ et k, l'objectif est d'extraire les k meilleurs sous-groupes par rapport à la mesure de qualité φ lorsque la taille du support est supérieure ou égale à $minSupp$.

Exemple. Considérons le jeu de données de la Table 1 composé de $\mathcal{O} = \{1, 2, 3, 4, 5, 6\}$ l'ensemble des objets et $\mathcal{A} = \{a_1, a_2, a_3\}$ l'ensemble des attributs. Chaque objet est associé à un label parmi $Dom(C) = \{l_1, l_2, l_3\}$. Le support de la description $d = \langle a_1 \leqslant 151,23 \leqslant a_2 \rangle$ est $\{2, 3, 5, 6\}$. Pour plus de lisibilité nous ne mentionnons pas les restrictions inutiles. $WRAcc(d, l_2) = \frac{|supp(d)|}{|\mathcal{O}|} \times (p^{l_2} - p_0^{l_2})$ avec p^{l_2} et $p_0^{l_2}$ définis par $p^{l_2} = \frac{|\{o \in supp(d) | class(o) = l_2\}|}{|supp(d)|}$ et $p_0^{l_2} = \frac{|\{o \in \mathcal{O} | class(o) = l_2\}|}{|\mathcal{O}|}$, ainsi $WRAcc(d, l_2) = 4/6 \times (3/4 - 1/2) = 0.25$.

Algorithmes. Pour traiter le problème de SD, différents algorithmes ont été utilisés (Herrera et al., 2011; Atzmüller et Puppe, 2006). Des algorithmes exhaustifs ont été proposés dont certains comme SD-Map sont particulièrement efficaces (Atzmüller et Puppe, 2006). Cependant, ils sont confrontés au problème de la redondance : beaucoup de sous-groupes sont similaires à un même optimum local (similarité des supports et des descriptions). Pour cela, un post-traitement est nécessaire afin de conserver seulement le meilleur sous-groupe parmi les sous-groupes redondants. De plus, lorsque les jeux de données deviennent trop grands, les approches exhaustives ne sont plus performantes (voire inutilisables) puisque la taille de l'espace de recherche est exponentielle. Pour cela, des algorithmes heuristiques ont été implémentés, la plupart basés sur une approche type beam search (van Leeuwen et Knobbe, 2012; Meeng et al., 2014). La recherche consiste à explorer l'espace de recherche (i.e., le treillis) de haut en bas (du sous-groupe le plus général au plus spécifique) de manière gloutonne. A chaque niveau du treillis, seulement certains des meilleurs sous-groupes vont être explorés au niveau suivant (la largeur du faisceau). En plus du problème de redondance, ce type d'exploration manque cruellement de diversité : seulement une partie des optimums locaux du treillis sont atteints. A chaque niveau, le choix glouton menant aux meilleurs fils est réalisé, empêchant ainsi la découverte d'optimum locaux dont l'évolution de la mesure de qualité n'est pas croissante. Nous définissons alors la diversité comme étant la proportion d'optimums locaux trouvés dans l'espace de recherche. Plus cette proportion est proche de 1 (tous les optimums locaux ont été trouvés), plus l'approche est dite diversifiée.

Exemple. La Figure 1 présente une partie de l'espace de recherche où chaque nœud est un sous-groupe. Les nœuds rouges sont les optimums locaux. Les zones vertes autour des optimums locaux contiennent les sous-groupes similaires avec les optimums locaux. Le faisceau jaune simule l'exploration de type Beam Search. Seulement les nœuds présents dans ce faisceau sont explorés : seulement 2 optimums locaux parmi les 4 du treillis sont trouvés. Parmi tous les

ID	a_1	a_2	a_3	C
1	150	21	11	$\{l_1\}$
2	128	29	9	$\{l_2\}$
3	136	24	10	$\{l_2\}$
4	152	23	11	$\{l_3\}$
5	151	27	12	$\{l_2\}$
6	142	27	10	$\{l_1\}$

TAB. 1 – *Exemple.*

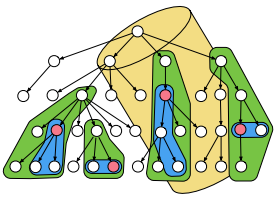

FIG. 1 – *Beam search.*

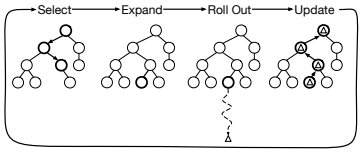

FIG. 2 – *Une itération MCTS.*

nœuds explorés dans le faisceau une grande partie sont redondants (i.e., similaires) avec les 2 optimums locaux trouvés. Pour supprimer la redondance, nous utilisons le post-traitement décrit dans (Bosc et al., 2016a).

3 Recherche Arborescente de Monte Carlo (MCTS)

MCTS est basée à la fois sur la puissance des simulations aléatoires et sur le compromis entre l'exploration de l'espace de recherche et l'exploitation d'une bonne solution (Browne et al., 2012). MCTS a principalement été utilisée en Intelligence Artificielle pour des applications qui peuvent être représentées par des arbres de décisions séquentielles, e.g., les jeux. Elle construit itérativement un arbre de recherche en fonction des résultats, appelés gains, obtenus durant les simulations. Chaque nœud s de l'arbre correspond à un état du jeu et est pourvu de deux paramètres : (i) le nombre de fois $N(s)$ où s a été visité (exploration) et (ii) le gain moyen obtenu $Q(s)$ lors des simulations jouées depuis s (exploitation). Les fils d'un nœud s correspondent aux états du jeu accessibles depuis s en jouant une action possible. A chaque itération, un nœud s' est ajouté dans l'arbre, et une simulation est jouée à partir de s'. Le gain de la simulation est propagé à tous les parents de s'. Une itération est réalisée en quatre étapes :

— La méthode SELECT, à partir du nœud racine de l'arbre, sélectionne récursivement une action menant à un fils jusqu'à que tous les fils du nœud sélectionné aient été créés. Le choix du fils (i.e., de l'action à jouer), est basé sur une mesure de limite supérieure de confiance (UCB) prenant en compte les deux paramètres Q et N de chaque fils s. Cette mesure UCB a été introduite pour des problèmes de machines à sous (Auer et al., 2002).

— La méthode EXPAND. Un fils s', obtenu en jouant une action disponible, du nœud sélectionné durant la méthode précédente est ajouté à l'arbre de recherche.

— La méthode ROLLOUT. A partir de s', une simulation aléatoire est jouée jusqu'à atteindre un nœud terminal (fin du jeu). Le gain Δ (1 si victoire, 0 si défaite) est renvoyé.

— La méthode UPDATE. Le gain Δ est propagé à tous les nœuds parents : de s' jusqu'à la racine de l'arbre. Le paramètre Q de chacun de ces nœuds est mis à jour en fonction de Δ et leur paramètre N est augmenté de 1.

Chacune des ces méthodes est effectuée itérativement jusqu'à ce que le budget de calcul soit atteint (e.g., un nombre d'itérations, un budget temps, etc.). Ensuite, l'action menant au meilleur fils de la racine est choisie pour être effectivement jouée.

Example. La Figure 2 présente une itération de MCTS. A partir d'un nœud, deux actions sont possibles. Puisque l'arbre de recherche contient déjà 8 nœuds (en plus de la racine), nous allons

commencer la 9ème itération de MCTS. En débutant par la racine, nous allons récursivement sélectionner un fils : le fils gauche de la racine, puis son fils droit. La méthode SELECT s'arrête à ce dernier nœud s puisque ses fils n'ont pas tous été créés. La méthode EXPAND ajoute un fils s' à s, i.e., son fils gauche. Une simulation est jouée à partir de s' jusqu'à atteindre un état terminal du jeu. Le gain Δ obtenu est ensuite propagé à s' et ses parents.

4 MCTS appliquée à la découverte de sous-groupes

L'espace de recherche des sous-groupes est un treillis : la racine s_0 est le sous-groupe avec la description la plus générale couvrant tous les objets. Sans perte de généralité, nous considérons ici simplement des attributs numériques. La description de s_0 est donc $d_0 = \langle f_1, \ldots, f_{|\mathcal{A}|} \rangle$ où $f_i = [min(Dom(a_i)), max(Dom(a_i))]$. Les fils d'un sous-groupe s sont les sous-groupes directement plus spécifiques que s. On considère que $a \in \mathcal{A}$ est un attribut avec $Dom(a = \{\alpha_1, \ldots, \alpha_{|a|}\}$ où α_i est une valeur prise par un objet dans $\mathcal{D}$ et $\alpha_i < \alpha_j, \forall 0 \leqslant i < j \leqslant |a|$. Soit $f = [\alpha_i, \alpha_j]$ $(i \neq j)$ une restriction sur a, il existe deux restrictions directes possibles : (i) en augmentant la borne inférieure $f^{bi} = [\alpha_{i+1}, \alpha_j]$ ou (ii) en diminuant la borne supérieure $f^{bs} = [\alpha_i, \alpha_{j-1}]$. Ainsi, chaque sous-groupes peut être spécialisé de $2 \times |\mathcal{A}|$ manières différentes. Pour répondre à un problème de SD, il est nécessaire d'explorer ce treillis.

MCTS construit un arbre de recherche sur ce treillis en l'explorant de haut en bas. Puisque la taille du support diminue en spécialisant les sous-groupes, les feuilles de l'arbre sont les sous-groupes dont aucun fils n'est fréquent : ce sont les **nœuds terminaux** en SD. Contrairement aux jeux où seuls les nœuds terminaux peuvent être évalués, en SD la mesure de qualité peut et doit s'appliquer à tous les nœuds de l'arbre. Puisque les mesures de qualité φ ne sont pas monotones sur l'ordre partiel des sous-groupes, aucun élagage selon φ n'est possible.

Appliquer MCTS pour SD. Étant donnés un jeu de données $\mathcal{D}(\mathcal{O}, \mathcal{A}, C, class)$, un nombre d'itérations N_{iter} (le budget de calcul), un seuil de support minimum $minSupp$, une mesure de qualité φ et un entier k, l'objectif est de construire un arbre de recherche contenant un ensemble de nœuds relatifs à des sous-groupes d'une bonne qualité (l'exploitation de MCTS) et d'une grande diversité (l'exploration de MCTS). Le résultat est composé des k meilleurs sous-groupes de l'arbre avec une grande diversité et une faible redondance. Pour éliminer la redondance, nous utilisons ensuite le même post-traitement que (Bosc et al., 2016a).

4.1 Construction de l'arbre de recherche

Nous détaillons l'adaptation de MCTS pour extraire des sous-groupes intéressants, divers et non redondants. Nous devons modifier le fonctionnement des quatre étapes de MCTS pour tenir compte des spécificités liées à SD.

SELECT. Cette méthode cherche à atteindre le nœud de l'arbre de recherche (ou sous-groupe) qu'il faut développer le plus urgemment. En SD, chaque nœud possède $2 \times |\mathcal{A}|$ fils (deux restrictions possibles pour chaque attribut numérique). Il faut donc récursivement choisir parmi les fils d'un nœud jusqu'à atteindre un nœud dont les fils ne sont pas tous encore créés dans l'arbre. Pour choisir le fils à sélectionner, MCTS est basée sur une UCB *Upper Confidence Bound for Tree*, baptisée UCT (Kocsis et Szepesvári, 2006). UCT permet de prendre en compte

le compromis exploration/exploitation. L'UCT d'un fils s' d'un nœud s est donnée par :

$$UCT(s, s') = Q(s') + 2\sqrt{\frac{\ln(N(s))}{N(s')}}$$

où $N(s)$ est le nombre de fois où le nœud s a été visité, et $Q(s) \in [0, 1]$ est la valeur agrégée des gains obtenus aux simulations issues de s. L'UCT est calculée pour chacun des fils s' de s et le fils avec la plus grande UCT est sélectionné. Ce procédé se répète tant que tous les fils du nœud sélectionné sont créés dans l'arbre.

EXPAND. Lorsque la méthode SELECT renvoie le dernier nœud (ou sous-groupe) sélectionné s, un fils s', spécialisation de s, est créé dans l'arbre. Le fils créé est choisi aléatoirement parmi les fils non encore créés de s dans l'arbre. On peut choisir différentes manières de spécialiser s pour obtenir s'. On peut simplement procéder à la restriction directe qui correspond au fils s' (augmentation directe de la borne inférieure ou diminution directe de la borne supérieure), ou l'on peut forcer le sous-groupe fils s' à avoir un support différent du père s. Ainsi on s'assure que s et s' n'appartiennent pas à la même classe d'équivalence (Pasquier et al., 1999).

ROLLOUT. Cette méthode consiste à parcourir aléatoirement une branche du treillis et renvoyer un gain. Dans le cas des jeux, le calcul du gain se fait obligatoirement sur un nœud terminal. Cependant en SD, tous les nœuds du treillis ont une mesure de qualité via φ. Ainsi, nous considérons que la simulation explore un chemin aléatoire dans le treillis $p = (s_1, ..., s_n)$ de taille n depuis le nœud s_1 (appelé s' avant) créé avec EXPAND vers le nœud final du chemin s_n. Nous proposons alors différentes stratégies pour créer ce chemin et pour calculer le gain :

— La stratégie RANDOMONE. La taille du chemin n est choisie aléatoirement dans l'intervalle $[1, |supp(s_1)|]$. Si le nœud s_n obtenu n'est pas fréquent, la méthode ROLLOUT est relancée. Le gain renvoyé est $\Delta = \varphi(s_n)$.

— La stratégie ROLLOUTMEAN. La taille du chemin n'est pas fixée, mais le chemin s'arrête lorsqu'un nœud terminal est atteint (s_n est un nœud terminal). Le gain renvoyé correspond à la moyenne de la qualité des nœuds du chemin, $\Delta = (\sum_{s \in p} \varphi(s))/n$.

— La stratégie ROLLOUTMAX. Le chemin s'arrête lorsqu'un nœud terminal est atteint. Le gain renvoyé correspond à la qualité maximale des nœuds du chemin, $\Delta = \max_{s \in S} \varphi(s)$.

— La stratégie ROLLOUTLARGE. Le chemin "saute" certains nœuds du treillis et s'arrête lorsqu'un nœud terminal est atteint. Le gain renvoyé est $\Delta = \max_{s \in S} \varphi(s)$. Cette stratégie est utile lorsque les nœuds terminaux sont très profonds dans le treillis (i.e., longs chemins).

UPDATE. Cette étape consiste à mettre à jour le nœud créé s' durant l'EXPAND ainsi que tous ses parents en fonction du gain Δ obtenu par la simulation. Pour chacun de ces nœuds s, on incrémente son nombre de visite *N(s)++*, et on met à jour son paramètre $Q(s)$ en fonction de Δ. Pour cela nous avons défini deux stratégies. La stratégie QMEAN considère que $Q(s)$ est la moyenne de tous les gains obtenus jusqu'à présent pour s : $Q(s) = (\Delta + ((N(s) - 1) \times Q(s)))/N(s)$. La stratégie QMAX considère que $Q(s)$ contient le maximum de tous les gains obtenus jusqu'à présent pour s : $Q(s) = max(Q(s), \Delta)$. Dans le cas où il y a une très faible proportion d'optimums locaux dans l'espace de recherche, l'utilisation de QMAX peut permettre d'identifier plus facilement les zones proches des optimums locaux.

Extraire les résultats de l'arbre. Une fois que le budget de calcul est atteint, il suffit d'extraire les k meilleurs sous-groupe de l'arbre partiellement construit en utilisant la mesure de similarité pour éliminer la redondance comme dans (Bosc et al., 2016a).

4.2 L'algorithme MCTS4SD

Pour illustrer le principe de l'algorithme, nous allons procéder à quelques itérations en considérant le jeu de données de la Table 1. Dans cet exemple, on considère simplement la mesure $\varphi = WRAcc$ sur le label l_2 avec $minSupp = 3$. Nous choisissons d'utiliser les stratégies ROLLOUTMAX et QMAX. Avant de commencer la première itération, l'arbre ne contient que la racine s_0 correspondant au sous-groupe avec la description la plus générale $d_0 = \langle\rangle$ ($supp(s_0) = \mathcal{O}$). On a alors, $\varphi(s_0, l_2) = 0$.

(i) - La première itération commence sur s_0. Puisque les fils de s_0 ne sont pas tous créés, la méthode SELECT renvoie s_0 comme étant le nœud sélectionné. Parmi les 6 fils possibles de s_0, un fils est aléatoirement choisi, par exemple le fils s_1 est créé avec $d_1 = \langle[128 \leqslant a_1 \leqslant 151]\rangle$, i.e., la borne supérieure de la restriction sur a_1 est diminuée. Ainsi, la mesure de qualité est $\varphi(s_1, l_2) = \frac{5}{6} \times (0.6 - 0.5) = 0.08$. $N(s_1)$ et $Q(s_1)$ sont initialisés à 0. A partir de s_1, on va procéder à une simulation jusqu'à un nœud terminal. Les nœuds visités durant la simulation ne sont pas stockés dans l'arbre. Par exemple, la simulation va parcourir les sous-groupes s_1^1 avec $d_1^1 = \langle[128 \leqslant a_1 \leqslant 151]$,$[23 \leqslant a_2 \leqslant 29]\rangle$ ($supp(s_1^1) = 4$ et $\varphi(s_1^1, l_2) = 0.17$), puis s_1^2 avec $d_1^2 = \langle[128 \leqslant a_1 \leqslant 151]$, $[23 \leqslant a_2 \leqslant 27]\rangle$ ($supp(s_1^2) = 3$ et $\varphi(s_1^2, l_2) = 0$). Le sous-groupe s_1^2 est un nœud terminal. La qualité maximale obtenue durant cette simulation est $\varphi(s_1^1, l_2) = 0.17$, donc le gain renvoyé est $\Delta = 0.17$. La méthode UPDATE met à jour les nœuds parents s_0 et $s_1 : N(s_1) = N(s_0) = 1$ et $Q(s_1) = Q(s_0) = 0.17$.

(ii) - La seconde itération commence sur s_0. Puisque les fils de s_0 ne sont pas encore tous créés, s_0 est le nœud sélectionné. Un fils parmi ceux qui n'ont pas encore été créés est ajouté à l'arbre, e.g., s_2 avec $d_2 = \langle[21 \leqslant a_2 \leqslant 27]\rangle : supp(s_2) = 5$, $\varphi(s_2, l_2) = -0.08$, $N(s_2) = 0$ et $Q(s_2) = 0$. On considère que le chemin suivi par la simulation considère s_2^1 avec $d_2^1 = \langle[21 \leqslant a_2 \leqslant 24]\rangle$ ($supp(s_2^1) = 3$ et $\varphi(s_2^1, l_2) = -0.08$). s_2^1 est un nœud terminal. La qualité maximale rencontrée durant la simulation est -0.08. Puisque le gain doit être compris entre 0 et 1, il peut être nécessaire de normaliser la mesure dans l'intervalle $[0, 1] : \Delta = 0$. Lors de la méthode UPDATE le gain est propagé à s_2 et $s_0 : N(s_2) = 1, N(s_0) = 2, Q(s_2) = 0$ et $Q(s_0)$ n'est pas modifié car $\Delta < Q(s_0)$.

Les itérations suivantes se font de manière similaire. Chacune ajoute un nœud à l'arbre de recherche. Pendant les 6 premières itérations, s_0 sera toujours le nœud sélectionné par la méthode SELECT puisque certains de ses fils n'auront pas encore été ajoutés. A la 7ème itération, le fils de s_0 ayant la plus grande UCT sera sélectionné.

5 Validation empirique

Jeux de données benchmarks et réels. Afin d'évaluer les performances de notre algorithme dénommé MCTS4DM, nous l'avons expérimenté sur différents jeux de données habituellement utilisés en SD (voir Table 2). Ces jeux de données sont soit issus du répertoire "Mulan" (mulan.sourceforge.net), soit du répertoire UCI. De plus, dans le contexte d'une collaboration avec un neuro-scientifique et un chimiste, nous disposons d'un jeu de données appelé *Olfaction* détaillé dans (Bosc et al., 2016a). Les objets sont des molécules odorantes décrites par des propriétés physicochimiques (les attributs) et associés à des odeurs (la classe).

Paramètres par défaut. Les expérimentations ont été réalisées sur une machine dotée d'un processeur Intel Core i7 2.2 GHz avec 16GB de RAM (la taille maximum du tas de la JVM

est de 4GB) tournant sur Mac OS X El Capitan version 10.11.4. Afin de procéder à des comparaisons équitables, nous avons utilisé l'algorithme BS4SD basé sur une approche type beam search dont l'implémentation a été réalisée de manière similaire à MCTS4DM et possédant le même post-traitement pour éliminer la redondance (Bosc et al., 2016a). Puisque MCTS4DM est basé sur de l'aléatoire, nous avons effectué 10 fois les mêmes expérimentations et nous discutons seulement les résultats moyens obtenus (la variance des résultats est faible). Conformément aux recommandations données dans (Meeng et al., 2014), la largeur du faisceau de BS4SD est fixée à 100. Par défaut, nous fixons le nombre d'itérations à $100k$, $minSupp = 15$, $k = 100$ et les stratégies *RollOutMax* et *QMax* sont utilisées.

Extraction de motifs à chaque instant. Un des avantages principaux de l'utilisation de MCTS est la possibilité d'obtenir des motifs à chaque instant dont la qualité s'améliore avec le temps. La Figure 3 (gauche et centre) présente l'évolution du temps d'exécution et de la qualité des sous-groupes extraits pour BS4SD (gauche) et MCTS4DM (centre) sur le jeu de données *Mushroom* (les résultats sont similaires pour les autres jeux de données). La mesure de qualité WRAcc est utilisée. On s'aperçoit que MCTS4DM est capable d'extraire de meilleurs sous-groupes en un temps plus faible que BS4SD : seulement une vingtaine de secondes (soit 50k itérations) sont nécessaires pour trouver le meilleur sous-groupe, alors que BS4SD ne parvient pas à le trouver même avec une largeur de faisceau de 100 et un temps d'exécution d'environ 200 secondes. On s'aperçoit également que la qualité des motifs extraits s'améliore avec le nombre d'itérations réalisées. De plus, le temps d'exécution est linéaire en fonction du nombre d'itérations. La Figure 3 (droite) montre les résultats de MCTS4DM sur le jeu de données *Olfaction* avec $\varphi = F_\beta - Score$ (Bosc et al., 2016a).

Consommation mémoire. Chaque itération réalisée avec MCTS4DM conduit à la création d'un nœud représentant un sous-groupe : l'utilisation mémoire est donc linéaire en fonction du nombre d'itérations. Expérimentalement, avec une taille du tas de la JVM de 4GB, cela permet d'exécuter environ 4 millions d'itérations, ce qui n'est clairement pas nécessaire pour la plupart des jeux de données. A titre de comparaison, une approche de type Beam Search crée à chaque niveau d'exploration seulement w sous-groupes où w est la largeur du faisceau.

Passage à l'échelle. Les jeux de données utilisés durant ces expérimentations sont de tailles variées que ce soit pour le nombre d'attributs ou pour le nombre d'objets. Le nombre d'attributs a un impact direct sur le comportement de MCTS4DM puisqu'il est nécessaire de créer tous les fils d'un nœud avant de passer à un niveau suivant de l'espace de recherche. Ainsi, si les optimums locaux sont situés à une profondeur importante, il est nécessaire de procéder à un grand nombre d'itérations pour atteindre ces nœuds. Concernant le nombre d'objets du jeu de données, l'impact est linéaire sur le temps d'exécution. La Figure 5 montre ce comportement lorsque l'on duplique les objets du jeu de données *BreastCancer*. Il est important de mentionner que l'axe des abscisses est linéaire puis exponentiel en fonction du nombre d'objets, ce qui explique le comportement exponentiel pour les derniers points de la courbe.

Taille des descriptions. L'un des pré-requis en SD est que la description des sous-groupes soit facilement interprétable par un expert. Pour cela, la taille des descriptions, i.e., le nombre de restrictions, doit être raisonnable. Les approches existantes utilisent un seuil pour gérer la taille maximale autorisée. Bien que nous donnons dans MCTS4DM la possibilité de choisir un seuil, il est possible de désactiver cette possibilité : la Figure 6 montre que la taille des descriptions alors obtenues reste faible et donc que l'interprétation est facilitée.

Qualité et diversité des motifs. Afin de montrer la diversité obtenue dans les motifs extraits,

nous comparons les deux algorithmes MCTS4DM et BS4SD sur des mêmes jeux de données avec des paramètres menant à des temps d'exécution similaires. La Figure 4 présente les 50 meilleurs motifs non redondants (après le même post-traitement pour filtrer la redondance) obtenus pour chacune des deux méthodes. La Figure 4 (gauche) concerne le jeu de données *Nursery*. On s'aperçoit qu'avec BS4SD, seulement 2 motifs non redondants sont extraits : seulement 2 optimums locaux ont été détectés. Avec MCTS4DM, 50 motifs non redondants, soit 50 motifs relatifs à 50 optimums locaux différents, sont détectés. MCTS4DM a donc détecté 25 fois plus d'optimums locaux que l'approche basée sur beam search. Ceci est dû à l'intérêt de la mesure UCT qui prend en compte le compromis entre l'exploration de zones peu visitées du treillis et l'exploitation de solutions intéressantes.

Cependant, lorsque les optimums locaux sont nombreux et situés à des niveaux très profonds de l'espace de recherche, il est nécessaire de procéder à un grand nombre d'itérations pour que l'arbre construit atteigne ces nœuds (voir la Figure 4 (centre) concernant le jeu de données *Cal500* avec la mesure WKL). On constate que BS4SD est capable de trouver 18 sous-groupes non redondants (soit relatifs à 18 optimums locaux différents).MCTS4DM est capable d'extraire une trentaine de motifs non redondants (donc relatifs à 30 optimums locaux différents) mais la qualité de ces motifs extraits est très faible. En fait, ce sont des motifs menant aux vrais optimums locaux qui sont situés à une profondeur importante de l'espace de recherche. MCTS4DM est en train d'atteindre ces optimums locaux, mais il faudrait davantage d'itérations pour les ajouter à l'arbre de recherche. La Figure 4 (droite) présente le même type d'information pour le jeu de données *Olfaction* avec la mesure de qualité F_β-Score : MCTS4DM est capable d'identifier de meilleurs optimums locaux que BS4SD.

Impact des différentes stratégies. Nous avons défini et expérimenté plusieurs stratégies pour les différentes méthodes de MCTS. La Table 3 montre les résultats sur le jeu *Olfaction* avec $200k$ itérations. On voit que la stratégie RANDOMONE n'est pas adaptée : puisque la méthode ROLLOUT est répétée tant que la contrainte de support n'est pas vérifiée, le temps d'exécution est long et le gain retourné est la mesure de qualité d'un nœud généré aléatoirement, donc potentiellement peu informatif sur la qualité de cette zone de du treillis. Les autres stratégies semblent similaires. La stratégie QMEAN semble conduire vers de moins bons motifs que QMAX (une baisse de qualité de 5%). La stratégie ROLLOUTLARGE est plus rapide puisqu'elle génère de moins longues simulations en permettant de "sauter" des nœuds du treillis.

Comparaison avec l'état de l'art. Nous avons comparé MCTS4DM à BS4SD afin notamment d'examiner la diversité. Considérons les approches existantes dans l'état de l'art. Nous avons implémenté une exploration en profondeur exhaustive pourvue du même post-traitement pour éliminer la redondance dans les motifs extraits. Sur le jeu *Nursery*, le temps d'exécution est de 29 minutes environ. Avec MCTS4DM, mille itérations suffisent (6 secondes) pour détecter les deux meilleurs motifs obtenus avec l'approche exhaustive. Avec $100k$ itérations (94 secondes), 8 des 10 meilleurs motifs obtenus avec l'approche exhaustive sont détectés.

Nous avons également utilisé des algorithmes existants en SD : le logiciel Cortana (Meeng et al., 2014) qui implémente différentes stratégies de type Beam Search et l'application Vikamine (Atzmüller et Puppe, 2006) qui propose l'algorithme SD-Map avec une exploration exhaustive très efficace basée sur FP-Growth. Cependant, ces approches ne possèdent pas de post-traitement pour éliminer la redondance et une comparaison des motifs extraits n'est donc pas possible. Pour cela, nous discutons simplement des temps d'exécution (même si le temps d'exécution du post-traitement pour la redondance utilisé dans MCTS4DM est pris en compte)

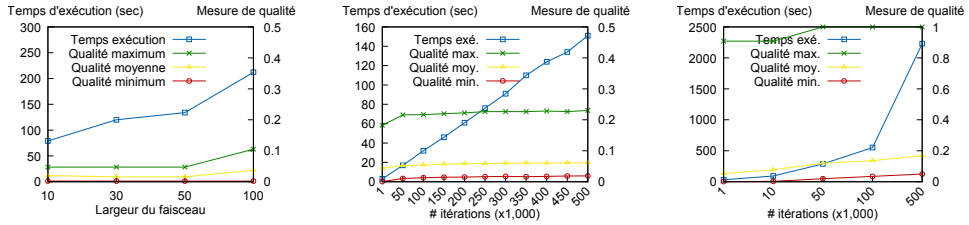

FIG. 3 – *Le temps d'exécution et quelques statistiques (minimum, maximum et moyenne) de la mesure de qualité des résultats sur le jeu Mushroom avec la mesure $WRAcc$ (gauche) et (droite) : (gauche) obtenus par* BS4SD*, et (centre) obtenus par* MCTS4DM*. La figure à droite concerne le jeu Olfaction avec* MCTS4DM *et la mesure F_β.*

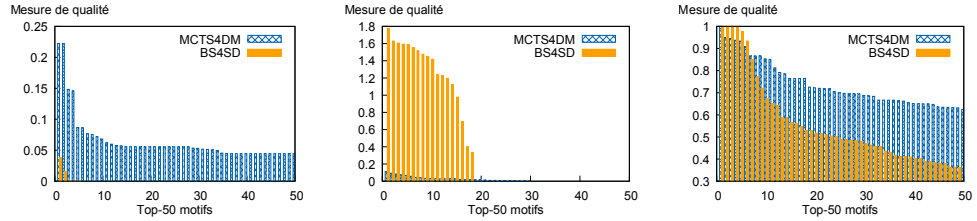

FIG. 4 – *Qualité des 50 meilleurs sous-groupes obtenus avec* MCTS4DM *et* BS4SD *: (gauche) sur* Nursery *avec $WRAcc$, (centre) sur* Cal500 *avec WKL, et (droite) sur* Olfaction *avec F_β.*

et de la qualité du meilleur motif extrait. Tout d'abord, nous nous intéressons à SD-Map sur le jeu *Mushroom*. Le temps d'exécution est de 5 minutes. C'est bien plus efficace que l'approche exhaustive naïve que nous avons développé. Cependant, il faut signaler qu'en ne gérant pas les attributs numériques SD-Map procède à une discrétisation de ceux-ci. MCTS4DM reste plus rapide pour extraire 8 des 10 meilleurs motifs dans cet l'espace de recherche. Ensuite, avec la stratégie *Cover based beam selection* implémentée dans Cortana, le meilleur motifs extrait avec une largeur du faisceau à 100 (temps d'exécution à 282s) correspond au même motif qu'avec MCTS4DM (temps d'exécution à 20s). Le temps d'exécution de l'approche ROC-Search (Meeng et al., 2014) implémentée dans Cortana est beaucoup plus long (supérieur à notre budget temps de 15 minutes), mais, dans les résultats intermédiaires, le meilleur motif à été trouvé. Ainsi, en général MCTS4DM est capable de trouver de bons motifs plus rapidement que les approches existantes, qu'elles soient exhaustives ou de type Beam Search.

6 Conclusion

Après avoir constaté la faible diversité des collections de motifs calculés par des approches de type Beam Search en SD, nous avons implémenté un nouvel algorithme d'exploration heuristique basé sur la recherche arborescente de Monte Carlo. Cette méthode peut être adaptée pour utiliser un grand nombre de mesures et de stratégies pour les quatre étapes de MCTS. Elle permet d'obtenir des motifs à tout instant et conduit en général vers de meilleurs résultats que

Jeu de données	$\lvert\mathcal{O}\rvert$	$\lvert\mathcal{A}_{bin}\rvert$; $\lvert\mathcal{A}_{num}\rvert$	$\lvert DomC\rvert$
Adult	30,162	99 ; 6	2
BreastCancer	683	0 ; 9	2
Mushroom	5,644	98 ; 0	2
Nursery	12,960	27 ; 0	5
Cal500	502	0 ; 68	174
Emotions	593	0 ; 72	6
Yeast	2,417	0 ; 103	14
Olfaction	1689	13	69 ; 74

TAB. 2 – *Jeux de données.*

ROLLOUT stratégie	UPDATE stratégie	Qualité moyenne	Temps (sec)
RollOutOne	QMax	0.122	4 325
RollOutMean	QMax	0.151	982
RollOutMean	QMean	0.143	964
RollOutMax	QMax	0.151	1028
RollOutMax	QMean	0.144	985
RollOutLarge	QMax	0.150	924

TAB. 3 – *Impact des différentes stratégies.*

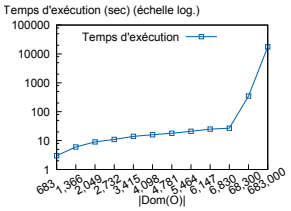

FIG. 5 – *Temps d'exécution en fonction du nombre d'objets*

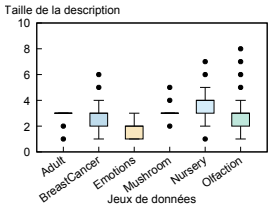

FIG. 6 – *Taille des descriptions obtenues.*

les approches existantes. Les perspectives sont nombreuses et il reste encore un grand nombre d'améliorations possible à étudier (mise au point de nouvelles stratégies pour les étapes de MCTS, prise en compte de nouveaux types de données comme des séquences ou des graphes).

Remerciements. Ce travail a été partiellement soutenu par le CNRS (projet Préfute) et l'*Institut rhônalpin des systèmes complexes* (IXXI).

Références

Atzmüller, M. et F. Puppe (2006). Sd-map - A fast algorithm for exhaustive subgroup discovery. In *ECML/PKDD*, pp. 6–17.

Auer, P., N. Cesa-Bianchi, et P. Fischer (2002). Finite-time analysis of the multiarmed bandit problem. *Machine Learning 47*(2-3), 235–256.

Bosc, G., J. Golebiowski, M. Bensafi, C. Robardet, M. Plantevit, J. Boulicaut, et M. Kaytoue (2016a). Local subgroup discovery for eliciting and understanding new structure-odor relationships. In *DS*, pp. 19–34.

Bosc, G., C. Raïssi, J. Boulicaut, et M. Kaytoue (2016b). Any-time diverse subgroup discovery with monte carlo tree search. *CoRR abs/1609.08827*.

Browne, C., E. J. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. P. Liebana, S. Samothrakis, et S. Colton (2012). A survey of monte carlo tree search methods. *IEEE Trans. Comput. Intellig. and AI in Games 4*(1), 1–43.

Duivesteijn, W., A. Feelders, et A. J. Knobbe (2016). Exceptional model mining - supervised descriptive local pattern mining with complex target concepts. *Data Min. Knowl.*

Discov. 30(1), 47–98.

Dzyuba, V., M. van Leeuwen, S. Nijssen, et L. D. Raedt (2014). Interactive learning of pattern rankings. *Int. Journal on Artif. Intell. Tools 23*(6).

Fürnkranz, J., D. Gamberger, et N. Lavrač (2012). *Foundations of Rule Learning*. Springer.

Galbrun, E. et P. Miettinen (2012). Siren : an interactive tool for mining and visualizing geospatial redescriptions. In *KDD*, pp. 1544–1547.

Herrera, F., C. J. Carmona, P. González, et M. J. del Jesús (2011). An overview on subgroup discovery : foundations and applications. *Knowl. Inf. Syst. 29*(3), 495–525.

Kocsis, L. et C. Szepesvári (2006). Bandit based monte-carlo planning. In *ECML*, pp. 282–293.

Leman, D., A. Feelders, et A. J. Knobbe (2008). Exceptional model mining. In *ECML/PKDD 2*, pp. 1–16.

Meeng, M., W. Duivesteijn, et A. J. Knobbe (2014). ROCsearch - a ROC-guided search strategy for subgroup discovery. In *SIAM DM*, pp. 704–712.

Moens, S. et M. Boley (2014). Instant exceptional model mining using weighted controlled pattern sampling. In *IDA*, pp. 203–214.

Novak, P. K., N. Lavrač, et G. I. Webb (2009). Supervised descriptive rule discovery : A unifying survey of contrast set, emerging pattern and subgroup mining. *J. Mach. Learn. Res. 10*, 377–403.

Pasquier, N., Y. Bastide, R. Taouil, et L. Lakhal (1999). Discovering frequent closed itemsets for association rules. In *ICDT*, pp. 398–416.

van Leeuwen, M. et A. J. Knobbe (2012). Diverse subgroup set discovery. *Data Min. Knowl. Discov. 25*(2), 208–242.

Wrobel, S. (1997). An algorithm for multi-relational discovery of subgroups. In *PKDD*, pp. 78–87.

Summary

Discovering descriptions that highly distinguish a class label from another is still a challenging task. Such patterns enable the building of intelligible classifiers and suggest hypothesis that may explain the presence of a label. Subgroup Discovery (SD), a framework that formally defines this pattern mining task, still faces two major issues: (i) to define appropriate quality measures characterizing the singularity of a pattern; (ii) to choose an accurate heuristic search space exploration when a complete enumeration is unfeasible. To date, the most efficient SD algorithms are based on a beam search. The resulting pattern collection lacks however of diversity due to its greedy nature. We propose to use a recent exploration technique, Monte Carlo Tree Search (MCTS). To the best of our knowledge, this is the first attempt to apply MCTS for pattern mining. The exploitation/exploration trade-off and the power of random search leads to any-time mining (a solution is available any-time and improves) that generally outperforms beam search. Our empirical study on various benchmark and real-world datasets shows the strength of our approach with several quality measures.