Compact-Diagram
Propagateur efficace pour la contrainte (s)MDD

Hélène Verhaeghe[1*] **Christophe Lecoutre**[2] **Pierre Schaus**[1]

[1]UCLouvain, ICTEAM, Place Sainte Barbe 2, 1348 Louvain-la-Neuve, Belgique
[2]CRIL-CNRS UMR 8188, Université d'Artois, F-62307 Lens, France
[1]{*prenom.nom*}@*uclouvain.be* [2]*lecoutre@cril.fr*

Résumé

Les diagrammes de décision multi-valués (MDD) représentent une structure utile pour la modélisation de problèmes combinatoires sous contraintes. Dans notre article [3] intitulé "Compact-MDD : Efficiently Filtering (s)MDD Constraints with Reversible Sparse Bit-Set" et publié à IJCAI-18, nous proposons une structure graphique proche, appelée sMDD, où la couche centrale peut s'avérer non-déterministe. Nous montrons tout d'abord qu'il est simple et efficace de transformer toute table (ensemble de tuples) en sMDD. Nous introduisons ensuite un nouvel algorithme de filtrage, appelé Compact-Diagram (CD), exploitant des vecteurs de bits et pouvant être appliqué aux contraintes sMDD. Nos expérimentations montrent l'intérêt pratique de notre approche.

1 Introduction

Une contrainte table, également nommée contrainte en extension, exprime explicitement soit les combinaisons de valeurs acceptées (*supports*), soit les combinaisons de valeurs rejetées (*conflits*) par les variables impliquées dans la contrainte. En théorie, toute contrainte peut se (re-)formuler sous forme de contrainte table. C'est l'une des raisons pour lesquelles ce type de contrainte est particulièrement prisé en programmation par contraintes.

Une contrainte MDD repose sur une structure différente qui est décrite ci-dessous. Un MDD (Multi-Valued Decision Diagram) est un graphe orienté acyclique. Lorsqu'une telle structure est associée à une contrainte, elle est organisée en $r+1$ couches de noeuds et r couches d'arêtes, où r représente le nombre de variables de la contrainte. Chaque couche d'arêtes est associée à une variable, chacune d'entre elles étant étiquetée par une valeur pouvant potentiellement être affectée à la variable. Chaque arête lie un noeud d'une couche i à un noeud de la couche $i+1$. Au niveau des noeuds, la première couche ne contient que la *racine* et la dernière couche ne contient que le noeud *terminal*. Chaque chemin allant de la racine au noeud terminal correspond donc à une instanciation possible des variables, ce qui est équivalent à un tuple dans une table.

Beaucoup d'efforts ont été consentis pour développer des algorithmes de filtrage dédiés aux contraintes tables et aux contraintes MDD. Actuellement, les algorithmes les plus efficaces sont respectivement Compact-Table (CT) [1] et MDD4R [2]. Dans cet article, nous proposons un nouveau propagateur pour les contraintes MDD, appelé Compact-Diagram (CD) et empruntant certains mécanismes à CT. Nous introduisons également une structure légèrement plus générale que MDD : le semi-MDD (sMDD), structure sur laquelle CD peut être appliqué.

2 Les semi-MDD (sMDD)

Un semi-MDD (ou sMDD) est, tout comme un MDD, un graphe orienté acyclique organisé en couches. Contrairement aux MDD, un semi-MDD n'est pas uniquement composé de noeuds déterministes (dits de décision). Un noeud est dit déterministe en entrée (respectivement, déterministe en sortie) s'il y a au niveau du noeud au plus une arête entrante (respectivement, sortante) étiquetée par une même valeur. S'il y a au moins deux arêtes entrantes (respectivement, sortantes) avec la même étiquette (valeur), alors le noeud est dit non-déterministe en entrée (respectivement, non-déterministe en sortie). Les noeuds d'un MDD sont par définition tous déterministes en sortie.

*Papier doctorant : Hélène Verhaeghe[1] est auteur principal.

Pour un semi-MDD d'arité r (et donc pourvu de $r+1$ couches de noeuds), les $\lfloor \frac{r}{2} \rfloor$ premiers niveaux sont déterministes en sortie, les deux niveaux suivants sont non-déterministes en entrée et sortie et pour finir, les $\lfloor \frac{r-1}{2} \rfloor$ derniers niveaux sont déterministes en entrée. Nous proposons une adaptation de *pReduce*, l'algorithme pour générer un MDD à partir d'une table. Cet algorithme que nous appelons *sReduce* permet de transformer toute table en sMDD.

3 Le propagateur Compact-Diagram

CD (Compact-Diagram) utilise, tout comme CT (Compact-Table), des vecteurs de bits pour améliorer les performances. Un vecteur de bits (basé sur un sparse set) réversible est associé à chaque couche du sMDD. Chacun des bits est associé à l'une des arêtes de la couche et permet de maintenir l'état de l'arête (à savoir si elle est encore valide et/ou accessible). Des vecteurs de bits pré-calculés supplémentaires regroupent les arêtes étiquetées de manière identique, ou encore les arêtes avec une même source ou une même destination.

Le filtrage se déroule en deux étapes. La première étape, de mise à jour, permet d'éliminer les arêtes qui ne sont plus accessibles. Sont éliminées tout d'abord les arêtes qui ont comme étiquette les valeurs supprimées des domaines depuis le dernier appel au propagateur. Cette première opération s'effectue de manière similaire à l'étape de mise à jour de CT. Ensuite, une propagation du retrait de ces arêtes doit être faite afin de conserver l'intégrité du sMDD (toute arête restante doit appartenir à un chemin allant de la racine au noeud terminal). Pour cette seconde opération, tout comme dans MDD4R, un double parcours (de haut en bas puis de bas en haut) de la structure est nécessaire.

La deuxième étape, le filtrage à proprement parler, se déroule comme pour CT, en effectuant l'intersection du vecteur de bits identifiant les arêtes encore valides et le vecteur de bits `supports` de la valeur testée. Si cette intersection est vide, la valeur peut être éliminée du domaine.

4 Résultats

Pour commencer, nous avons comparé les MDD et les sMDD générés par *sReduce* (une option permet de choisir si un 'simple' MDD doit être généré). Nous nous intéressons au nombre de noeuds et au nombre d'arêtes. Nous avons pu observer que pour 70% des instances, les MDD générés comportent au moins 7 fois plus de noeuds. De plus, il n'existe aucune instance dans notre benchmark où le nombre de noeuds s'est avéré plus petit dans les MDD générés. Concernant les arêtes, généralement les MDD générés en comportent moins que les sMDD. Toutefois, pour 85% des sMDD le nombre d'arêtes est au maximum 1,5 fois celui des MDD. Pour les 15% restants, le nombre d'arêtes est au plus 4 fois plus important.

Ensuite, nous avons comparé CD à MDD4R. MDD4R appliqué aux contraintes MDD est comparé à CD appliqué aux contraintes MDD et sMDD. De manière générale, CD est plus performant que MDD4R. De plus, le choix de la structure sMDD (plutôt que MDD) est typiquement un avantage. En partie, cela est dû au nombre réduit de noeuds.

CD a également été comparé à CT. CT reste globalement plus performant, malgré une réduction de l'écart de performance entre les deux types de représentations (tables versus diagrammes).

5 Conclusion

Nous avons proposé une nouvelle structure graphique, appelée sMDD, combinant deux demi-MDD ainsi qu'un algorithme permettant de générer de telles structures à partir de tables. Nos résultats montrent qu'un MDD a généralement plus de noeuds et moins d'arêtes qu'un sMDD correspondant. Nous avons également proposé un nouveau propagateur générique pour les contraintes sMDD. Nos résultats montrent l'obtention d'un gain de performance par rapport à MDD4R. Pour plus de détails concernant les structures algorithmes et résultats, n'hésitez pas à vous reporter à l'article original.

Remerciements

Le second auteur est financé par le projet CPER Data des Hauts-de-France.

Références

[1] Jordan Demeulenaere, Renaud Hartert, Christophe Lecoutre, Guillaume Perez, Laurent Perron, Jean-Charles Régin et Pierre Schaus : Compact-Table : efficiently filtering table constraints with reversible sparse bit-sets. *In Proceedings of CP'16*, pages 207–223, 2016.

[2] Guillaume Perez et Jean-Charles Régin : Improving GAC-4 for Table and MDD constraints. *In Proceedings of CP'14*, pages 606–621, 2014.

[3] Hélène Verhaeghe, Christophe Lecoutre et Pierre Schaus : Compact-MDD : Efficiently filtering (s)MDD constraints with reversible sparse bit-sets. *In IJCAI18*, pages 1383–1389, 2018.