Extension de Compact-Table aux tables négatives et concises

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

Ces dernières années, plusieurs algorithmes pour la contrainte table ont été proposés pour assurer la propriété de cohérence d'arc généralisée (GAC). Compact-Table (CT) [1] est un algorithme récent de l'état-de-l'art, que nous avons étendu dans notre article [4], intitulé "Extending Compact-Table to Negative and Short Tables" et publié à AAAI-17, aux tables concises (contenant des supports courts) et aux tables négatives (contenant des conflits).

1 Introduction

La contrainte table, ou contrainte en extension, exprime explicitement pour les variables impliquées, soit les combinaisons de valeurs acceptées (*supports*), soit les combinaisons de valeurs rejetées (*conflits*). En théorie, toute contrainte peut se reformuler sous forme de contrainte table, c'est l'une des raisons pour lesquelles ce type de contrainte est très important en programmation par contraintes.

Beaucoup d'efforts ont été consentis au développement d'algorithmes de filtrage pour les contraintes tables, le plus efficace ayant été demontré être Compact-Table [1], proposé en 2016. L'un des inconvénients avec les tables est que leur taille, et donc l'utilisation mémoire requise, peut potentiellement augmenter de manière exponentielle par rapport à l'arité. Pour pallier ce problème, plusieurs méthodes de compressions ont été proposées : les *tries*, les Diagrammes de Décision Multi-valeurs (*MDDs*) et les Automates Finis Déterministes (*DFAs*), qui sont des représentations pouvant faciliter le processus de filtrage.

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

D'autre approches, basées sur le concept de produit cartésien, ont également été envisagées : tuples compressés (*compressed tuples*), supports courts (*short support*), tables découpées (*sliced tables*), tables intelligentes (*smart tables*),...

Dans cet article, nous étendons l'algorithme Compact-Table, initialement introduit pour les tables positives sans aucune compression [1], afin de pouvoir gérer :

— les tables négatives (i.e. les tables de conflits) ;
— et/ou les tables concises, i.e. les tables avec supports courts qui sont des tuples contenant la valeur universelle * représentant n'importe quelle valeur pour une variable.

2 Gérer les tables concises : CT*

De manière intéressante, nous avons pu observer que CT peut être facilement adapté pour gérer les tables positives avec supports courts et cela sans dégradation de la complexité : elle reste en $\mathcal{O}(rd\frac{t}{w})$, où r est l'arité, d la taille du domaine le plus grand, t le nombre de tuples dans la table concise (contenant les *) et w la taille d'un mot processeur (e.g. $w = 64$).

La modification se caractérise par une nouvelle version de la règle de mise à jour de la table des tuples encore acceptables : pour une variable donnée x, le tuple τ est retiré de l'ensemble des tuples encore acceptables si $\tau(x) \neq *$ et $\tau(x) \notin D_x$, où D_x est le domaine de la variable x.

Les résultats obtenus, sur 600 instances variées et aléatoirement générées, avec CT*, ShortSTR2 (extension de STR2 gérant des supports courts [2]), CT et STR2 (pour ces deux derniers, les tables en entrée sont les tables équivalentes décompressées), sont

visibles sur la figure 1, sous forme de profiles de performance.

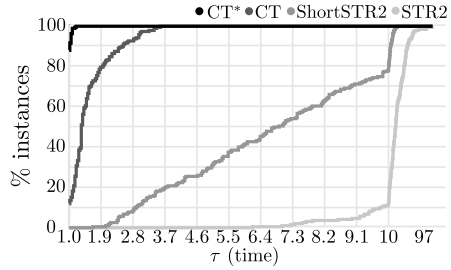

FIGURE 1 – Résultats sur tables positives concises

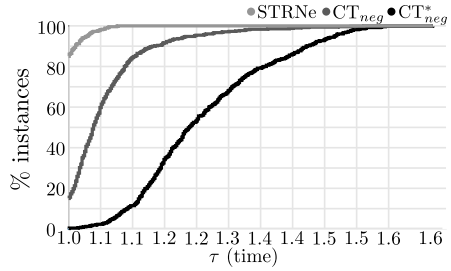

FIGURE 2 – Résultats sur tables négatives (instances avec solutions multiples)

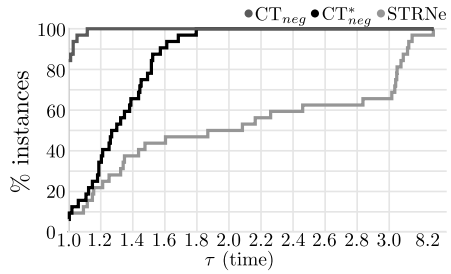

FIGURE 3 – Résultats sur tables négatives (instances sans solutions)

3 Gérer les tables négatives : CT$_{neg}$

Le changement se situe ici lors du processus de filtrage. Pour déterminer si un litéral (x,v) doit être conservé, il faut vérifier qu'il existe au moins un tuple valide, considérant l'état courant des domaines, qui ne corresponde pas à un conflit. Cette vérification peut être réalisée en comptant le nombre de conflits toujours valides contenant ce litéral. On compare cette valeur au produit des tailles courantes des domaines de toutes les variables, excepté celle du litéral. Une égalité entre ces deux nombres implique qu'il n'y a aucun tuple réalisable avec le litéral qui ne soit pas un conflit.

La complexité devient $\mathcal{O}(rd\frac{t}{w}k)$, k étant la complexité pour compter le nombre de 1 présents dans un mot processeur. Dans notre implémentation, avec l'utilisation de Long.bitCount, $k = log(w)$, mais sur certaines architecture, nous avons même $k = 1$.

Les résultats obtenus sur des instances variées et aléatoirement générées sont donnés par les figures 2 et 3. On peut y constater que nous gagnons vraiment en efficacité par rapport à STRNe [3] lorsque nous traitons des instances incohérentes (i.e. sans solutions).

4 Gérer les tables négatives et concises : CT$^*_{neg}$

Les modifications faites pour gérer les conflits avec des supports courts se basent sur l'agrégation des deux idées à la base de CT* et CT$_{neg}$ moyennant l'hypothèse de ne pas avoir de tuples se superposant. Cette hypothèse est nécessaire pour garder un comptage trivial des tuples.

5 Conclusion

Nous avons proposé une extension de Compact-Table qui exploite l'efficacité des opérations bit à bit. Cela permet d'opérer, de manière compétitive, sur les tables concises (CT*), les tables négatives (CT$_{neg}$) et les tables négatives concises (CT$^*_{neg}$). Nous pensons qu'elle sera implantée dans les solvers car les tables consises et/ou négatives vont devenir de plus en plus populaires, apportant à l'utilisateur une certaine facilité de modélisation. Pour plus de détails concernant l'implémentation et les résultats, n'hésitez pas à lire l'article originel.

Références

[1] J. Demeulenaere, R. Hartert, C. Lecoutre, G. Perez, L. Perron, J.-C. Régin, and P. Schaus. Compact-table : efficiently filtering table constraints with reversible sparse bit-sets. In *Proceedings of CP'16*, pages 207–223, 2016.

[2] C. Jefferson and P. Nightingale. Extending simple tabular reduction with short supports. In *Proceedings of IJCAI'13*, pages 573–579, 2013.

[3] Hongbo Li, Yanchun Liang, Jinsong Guo, and Zhanshan Li. Making simple tabular reduction works on negative table constraints. In *Proceedings of AAAI'13*, pages 1629–1630, 2013.

[4] Hélène Verhaeghe, Christophe Lecoutre, and Pierre Schaus. Extending compact-table to negative and short tables. In *Proceedings of AAAI'17*, 2017.