Exploitation des techniques de fouille de données pour la compression de contraintes table

Soufia Bennai *, Kamal Amroun **, Samir Loudni ***

*LIMED - Faculté des Sciences Exactes, Université de Bejaia, Algérie
sofia.bennai21@gmail.com
**LIMED - Faculté des Sciences Exactes, Université de Bejaia, Algérie
kamalamroun@gmail.com
***GREYC (CNRS UMR 6072), Université de Caen Normandie, France
samir.loudni@unicaen.fr

Résumé. Dans ce papier, nous proposons une amélioration de l'étape de compression de la méthode proposée par Gharbi et al. (2014) pour la compression des contraintes table. En plus de la fréquence des motifs, nous exploiterons leur couverture dans le FP-Tree afin de faciliter la création des tables fragmentées. Pour décider si un motif fréquent est nécessaire à la compression, nous proposons d'exploiter la métrique de " taux de compression" au lieu du "calcul de gain". Cela permet d'obtenir une compression plus élevée et une meilleure résolution du problème. Les résultats expérimentaux sont encourageants.

1 Introduction

Un problème de satisfaction de contraintes (CSP) peut être défini comme un ensemble de variables avec des domaines finis de valeurs possibles et un ensemble de contraintes définies sur des sous-ensembles de variables. Les contraintes définies en extension (contraintes table) sont largement utilisées (bases de données, problèmes de configuration, etc.). Une liste de tuples des valeurs autorisées ou non-autorisées est donnée. La taille des contraintes table peut être très importante, ce qui peut constituer un obstacle au processus de résolution. De nombreux travaux ont été proposés dans la littérature pour la compression des contraintes table dans le but de faciliter la résolution du CSP correspondant. Nous pouvons citer Sliced tables Gharbi et al. (2014), micro-structure based compression Jabbour et al. (2015), etc.
Dans ce papier, nous proposons une amélioration de l'étape de compression de la méthode proposée par Gharbi et al. (2014) (appelée dans ce papier FPTCM) et qui exploite l'arbre de préfixe FP-Tree pour énumérer les motifs fréquents nécessaires à la compression des contraintes table. Notre première amélioration consiste à construire le FP-Tree autrement. Au lieu de considérer uniquement la fréquence des motifs dans la table des fréquences et dans le FP-Tree, nous exploitons aussi la notion de couverture d'un motif qui représente l'ensemble des indices des tuples où un motif apparaît. Cela permettra de réduire le temps de construction de la table compressée. La deuxième amélioration consiste à calculer le taux de compression estimé qu'on peut obtenir en compressant les tuples dont les indices apparaissent dans la couverture d'un

motif fréquent u au lieu de calculer le gain que ce motif peut offrir (i.e. $|u| * (f-1)$ tel que f est la fréquence de u) afin de décider si u est nécessaire à la compression ou non. Nous procédons ainsi car le calcul du gain qu'offre un nœud et celui qu'offre son nœud parent ne prend pas en considération le même nombre de tuples. Nous avons évalué notre proposition sur des benchmarks reconnus et les résultats obtenus sont très encourageants.
Le reste du papier est organisé comme suit. Dans la Section 2, nous donnons quelques définitions sur les CSP ainsi que sur la fouille de motifs fréquents. Dans la Section 3, nous détaillons l'étape de compression de la méthode proposée par Gharbi et al. (2014) ainsi que ses faiblesses. La Section 4 est consacrée à notre proposition. Les résultats expérimentaux sont présentés dans la section 5. Nous concluons ce papier, qui est un résumé d'un papier accepté à la conférence ICTAI2019 (31^{st} International Conference on Tools with Artificial Intelligence), par quelques remarques et perspectives.

2 Préliminaires

Un CSP (Montanari, 1974) consiste en un ensemble de variables V= $\{v_1, \cdots, v_n\}$ avec des domaines finis $\mathcal{D} = \{D_1, \ldots, D_n\}$ tel que chaque D_i est un ensemble de valeurs qui peuvent être assignées à la variable v_i, et un ensemble de contraintes $\mathcal{C}$. Chaque contrainte $c_i \in \mathcal{C}$ est définie par la paire $(S(c_i), R(c_i))$, tel que $S(c_i)$ (le scope c_i) est un ensemble de variables impliquées dans c_i et $R(c_i)$ est une relation qui définie l'ensemble de tuples autorisés pour les variables de c_i. L'arité d'une contrainte c_i représente la taille de son scope. L'objectif est de trouver une affectation $(v_i = d_i)$ avec $d_i \in D_i$ pour $i = 1, \ldots, n$, tel que toutes les contraintes sont satisfaites.

Exemple 1 : Soit le CSP suivant :
$V = \{v_0, \ldots, v_5\}$, $\mathcal{D} = \{D_0 \ldots, D_5\}$, $D_0 = D_1 = D_2 = D_3 = D_4 = D_5 = \{1, \ldots, 7\}$.
$\mathcal{C} = \{c_0\}$ tel que, $c_0 = \{(v_0, v_1, v_2, v_3, v_4, v_5), R(c_0)\}$ et
$R(c_0) = \{(1\ 2\ 1\ 1\ 1\ 1), (1\ 2\ 1\ 2\ 1\ 1), (1\ 2\ 1\ 3\ 1\ 1), (1\ 3\ 1\ 1\ 1\ 1), (1\ 3\ 2\ 1\ 1\ 1), (1\ 3\ 1\ 2\ 1\ 1), (1\ 3\ 1\ 3\ 1\ 1), (1\ 3\ 1\ 3\ 1\ 2), (1\ 4\ 2\ 1\ 1\ 1), (1\ 4\ 2\ 2\ 1\ 1), (1\ 5\ 2\ 1\ 1\ 1), (2\ 3\ 1\ 1\ 1\ 1), (2\ 3\ 1\ 3\ 1\ 1), (2\ 4\ 2\ 1\ 1\ 1), (2\ 4\ 2\ 2\ 1\ 1), (2\ 4\ 2\ 3\ 1\ 1), (3\ 3\ 1\ 1\ 1\ 1), (4\ 3\ 1\ 1\ 1\ 1), (4\ 3\ 1\ 2\ 1\ 1), (5\ 4\ 2\ 1\ 1\ 1), (7\ 4\ 2\ 3\ 2\ 2)\}$.

Soit I un ensemble de n literaux distincts appelés motifs, un motif de taille l est un sous-ensemble non vide de I. Le langage de motifs correspond à $L_I = 2^I \setminus \emptyset$. Un *base transactionnelle* est un multi-ensemble de m motifs de L_I. Chaque motif de taille l, appelé *transaction* (ou tuple), est une entrée du dataset. Soit T une base transactionnelle, $p \in L_I$ un motif, et $match : L_I \times L_I \mapsto \{true, false\}$ un operateur matching. La couverture de p dans T, notée $cov(p)$, est l'ensemble de transactions de T qui couvrent p : $cov(p) = \{t \in T \mid match(p, t) = true\}$. La fréquence de p est la taille de sa couverture : $f(p) = |cov(p)|$.

Soit le CSP $P = (V, \mathcal{D}, \mathcal{C})$ et la relation $R(c_i)$ associée à la contrainte $c_i \in \mathcal{C}$. $R(c_i)$ peut être représentée sous forme d'une table transactionnelle $\mathcal{C}R_{c_i}$ définie comme suit : (i) l'union des valeurs des domaines des variables impliquées dans le scope de c_i représente l'ensemble des motifs de $\mathcal{I}$, (ii) L'ensemble des valeurs impliquées dans les tuples $t \in R(c_i)$ forment une transaction dans T. La Table 1a est la table transactionnelle associée à la relation $R(c_0)$ de l'Exemple 1. Dans la suite, un motif u de la relation $R(c_i)$ est l'affection de quelques variables impliquées dans le scope de la contrainte c_i.

IdT	v_0	v_1	v_2	v_3	v_4	v_5
t_0	1	2	1	1	1	1
t_1	1	2	1	2	1	1
t_2	1	2	1	3	1	1
t_3	1	3	1	1	1	1
t_4	1	3	2	1	1	1
t_5	1	3	1	2	1	1
t_6	1	3	1	3	1	1
t_7	1	3	1	3	1	2
t_8	1	4	2	1	1	1
t_9	1	4	2	2	1	1
t_{10}	1	5	2	1	1	1
t_{11}	2	3	1	1	1	1
t_{12}	2	3	1	3	1	1
t_{13}	2	4	2	1	1	1
t_{14}	2	4	2	2	1	1
t_{15}	2	4	2	3	1	1
t_{16}	3	3	1	1	1	1
t_{17}	4	3	1	1	1	1
t_{18}	4	3	1	2	1	1
t_{19}	5	4	2	1	1	1
t_{20}	7	4	2	3	2	2

(a) Table transactionnelle $R(c_0)$.

v_0 v_2	v_1 v_3 v_4 v_5
1 1	2 1 1 1
	2 2 1 1
	2 3 1 1
	3 1 1 1
	3 2 1 1
	3 3 1 1
	3 3 1 2

(b) Un fragment de $R(c_0)$.

TAB. 1: Exemple.

L'idée principale basée sur l'extraction de motifs pour une représentation compressée des contraintes table consiste à utiliser les motifs fréquents énumérés comme un **résumé** de l'ensemble des transactions. Ces transactions sont remplacées par chaque motif fréquent qui les couvre. La relation compressée résultante consiste en un ensemble de fragments tel que chaque fragment est constitué par un motif fréquent et une sous-table qui lui correspond.

La sous-table St associée à un motif fréquent u d'une contrainte c_i est un ensemble de tuples de $R(c_i)$ contenant u tel que u est supprimé de ces tuples. Un fragment d'une relation $R(c_i)$ est une paire (u, St) tel que u représente un motif fréquent et St sa sous-table correspondante.

Exemple 2 : La Table 1b montre un fragment correspondant au motif fréquent $u = \{v_0 = 1, v_2 = 1\}$ et sa sous-table résultante.

Soit u un motif fréquent, f étant sa fréquence et T l'ensemble des transactions compressées. $taille_a$ (resp. $taille_b$) correspond à la taille de T après (resp. avant) la compression. Pour évaluer la qualité de compression de l'ensemble des transactions T par un motif fréquent u, nous définissons la métrique suivante :

Definition 1 *Le taux de compression d'un ensemble de transactions T avec un motif fréquent u est défini comme suit :*

$$Taux = 1 - \frac{taille_a}{taille_b} \tag{1}$$

tel que $taille_b = arite * f$.

3 Méthode de compression basée sur l'arbre préfixé

Dans cette section, nous décrivons brièvement l'approche de Gharbi et al. (2014), appellée dans ce papier `FPTCM` pour la compression de contraintes table basée sur l'arbre préfixé. Le principe de `FPTCM` est d'énumérer uniquement les motifs fréquents nécessaires à la compression. Ces motifs sont obtenus comme suit : `FPTCM` calcule d'abord la fréquence f de chaque motif de taille 1, puis ordonne chaque tuple d'une manière décroissante selon les valeurs de fréquence de ces mitifs. Les motifs ayant une valeur de fréquence inférieure au seuil minimal

S_{min} seront supprimés des tuples car ils ne peuvent pas apparaitre dans les motifs fréquents. Dans le reste de ce papier, on considère $S_{min} = 2$. Ensuite, une fois qu'un tuple est ordonné, il est ajouté au FP-Tree qui est un arbre dont chaque branche représente la partie fréquente d'un tuple et chaque nœud contient le nombre de branches qui le partagent (voir Han et al. (2004) pour plus de détails sur les étapes de construction d'un FP-Tree). Chaque arête reliant un nœud parent à son nœud fils est marquée par une valeur. Le nœud root n'est pas marqué. Dans le FP-tree de la méthode FPTCM, chaque nœud contient le nombre de tuples dont le motif représenté par le chemin allant du nœud root au nœud en question apparaît. Une fois que le FP-Tree est construit, les nœuds de l'arbre offrant un moindre gain par rapport à leur parents sont supprimés. Le gain que peut offrir un motif u est égal à $|u| * (f - 1)$ valeurs, tel que f est la fréquence de u. Les motifs restants sont ceux necéssaires à la compression. Enfin, FPTCM crée un fragment pour chaque motif fréquent u en parcourant la relation pour trouver les tuples contenant u, supprimer u de ces tuples. Ensuite, elle ajoute les tuples réduits à la sous-table correspondant à u. Les tuples ne contenant pas de motifs fréquents seront ajoutés à la table par défaut.

4 Amélioration de l'étape de compression de FPTCM

Dans cette section, nous présentons notre approche, appelée FPTCM$^+$, qui est une amélioration de la méthode FPTCM. L'amélioration est double :

Premièrement, FPTCM$^+$ utilise une table de fréquences dans laquelle elle sauvegarde les motifs de taille 1 ainsi que leurs couvertures. En effet, chaque nœud du FP-Tree contient la couverture du motif correspondant au chemin allant de ce nœud au nœud root. Cela améliore considérablement la construction de la table compressée car pour chaque motif fréquent on connait sa couverture. De ce fait, la méthode évite de parcourir la table pour vérifier quels sont les tuples contenant un motif fréquent.

Deuxièmement, au lieu de calculer le gain (en nombre de valeurs) que peut offrir un motif fréquent u, nous proposons dans FPTCM$^+$ d'exploiter le taux de compression qu'on peut atteindre en compressant avec u. FPTCM$^+$ compare le taux de compression estimé par le motif fréquent u aux taux de compression estimés par ses spécialisations (i.e. super-motifs [1]). Si le taux de compression estimé par u est supérieur à celui obtenu par ses super-motifs fréquents, le motif u sera considéré comme étant nécessaire à la compression ; sinon, ses super-motifs fréquents seront considérés. Ce processus est répeté pour chaque super-motif fréquent de u.

Les étapes de notre méthode sont illustrées par l'algorithme 1. Les différences entre FPTCM$^+$ et FPTCM sont surlignées en gras. D'abord, la table des fréquences associée à la table transactionnelle CR_{c_i} correspondant à la contrainte table $R(c_i)$ est creée. Elle est représentée sous forme d'une matrice contenant les motifs de taille 1 ainsi que leurs couvertures. Cette étape nécessite un parcours de la table. Ensuite, dans un second parcours de la table, chaque tuple est ordonné dans un ordre décroissant des valeurs des fréquences et inséré dans le FP-Tree T sous forme d'une branche (lignes 2-3). Si un motif partage un préfixe avec un autre motif de l'arbre, cette partie de la branche sera commune. De plus, chaque nœud de l'arbre T sauvegarde les identifiants des tuples contenant le motif représenté par le chemin allant du nœud root

1. Un super-motif d'un motif u est obtenu en ajoutant à u un motif frequent e_i de taille 1 de tel sorte que $e_i \notin u$ est le label de l'arrêt allant de u à son nœud fils dans le FP-tree

Algorithme 1 : $FPTCM^{+}$ ($\mathcal{CR}_c$. : table, S_{min} : entier)

Data : $LMFreq$: liste de motifs
Result : $R^c(c_i)$ relation compressée
1 **créer la table des fréquences correspondant à** CR_c ;
2 **foreach** $t_j \in \mathcal{CR}_c$ **do**
3 trier le tuple t_j dans un ordre décroissant des valeurs de fréquence des motifs de taille 1 et supprimer les motifs ayant une fréquence inférieure à S_{min} ;
4 **ajouter** t_j **à l'arbre FP-Tree** T ; `// les nœufs représentent la fréquence des motifs`
5 **end**
6 Réduire T en supprimant les nœuds ayant une fréquence inférieure à S_{min} ;
7 **foreach** *nœud fils* n *du nœud root de* T **do**
8 **MFItemset**($n, S_{min}, LMFreq$) ;
9 **end**
10 **foreach** *motif* $mtf \in LMFreq$ **do**
11 **compresser les tuples contenant** mtf,
12 **end**

au nœud en question. Enfin, les nœuds ayant une fréquence inférieure au seuil minimal S_{min} sont supprimés de T, parce qu'ils ne peuvent pas être des motifs fréquents (ligne 6).

L'Algorithme 2 résume les différentes étapes pour l'identification des motifs fréquents nécessaires à la compression à partir du FP-Tree en exploitant la métrique basée sur le taux de compression. D'abord, pour chaque nœud n qui n'est pas un nœud feuille, ajouter à la liste $Child_n$ (soit $nbrCh$ sa taille) tous les nœuds fils ayant une fréquence (la fréquence est donnée par le nœud lui-même) supérieure ou égale à S_{min} (ligne 4). Soit u un motif fréquent correspondant à la branche allant du nœud root au nœud n et f sa fréquence. Ensuite, la somme des fréquences de ces nœuds fils est calculée (soit sum_f cette somme) (ligne 5). Pour décider quel motif fréquent est nécessaire à la compression, nous comparons le taux de compression du motif u (noté $Taux$) à celui de ses super-motifs (noté $ChTaux$) (lignes 6-8). Si la valeur de $ChTaux$ est inférieure à celle de $Taux$, le motif courant u est considéré comme étant nécessaire à la compression, sinon considérer ses super-motifs (voir lignes 8-17). Pour chaque super-motif ch, faire un appel récurssif à la procédure $MFItemsets$. Dans la ligne 14, les transactions qui ne sont pas couvertes par les super-motifs de u (i.e., $sum_f < f$) seront compressées en utilisant le motif u (ligne 15). Dans le cas où le nœud n est une feuille, le motif associé est considéré comme nécessaire à la compression (lignes 1-2). Pour finir la compression, créer un fragment (u, St) pour chaque motif fréquent u identifié. Pour chaque tuple t_i dans $cover(u)$, supprimer u de t_i ensuite l'ajouter à la sous-table St correspondant à u.

4.1 Exploitation du taux de compression

Afin d'identifier les motifs fréquents nécessaires à la compression à partir de l'arbre, nous proposons d'exploiter le calcul de taux de compression. Soit u un motif fréquent qui correspond au chemin allant du nœud root au nœud courant et $arite$ l'arité de la contrainte table à compresser. Nous avons deux cas :

Algorithme 2 : MFItemset (n : nœud, S_{min} : entier, $LMFreq$: liste de motifs)

Result : $LMFreq$:liste de motifs ;
// u un motif correspondant au chemin allant du nœud root à n
if *n est un nœud feuille* **then**
  Ajouter u à $LMFreq$;
else
  ajouter à $child_n$ les neouds fils de n ayant une fréquence supérieure ou égale à S_{min}
  $sum_f \leftarrow \sum_{v \in child_n} f(v)$;
  calculer le taux de compression $Taux$ estimé par u ;
  calculer le taux de compression $ChTaux$ estimé par les super-motifs de u;
  if $(ChTaux < Taux)$ **then**
    Ajouter le motif u à $LMFreq$;
  else
    foreach $v \in child_n$ **do**
      MFItemset (v, S_{min}, $LMFreq$);
    end
    if $sum_f < f$ **then**
      Ajouter le motif u à $LMFreq$;
    end
  end
end

1. Si la compression est obtenue en utilisant le motif u, la taille $taille_a$ des transactions après leur compression est égale à la taille du motif u plus la taille de sa sous-table correspondante. La taille de la sous-table est obtenue en multipliant l'arité de la sous-table $(arite - |u|)$ par la fréquence de u : $taille_a = |u| + (arite - |u|) * f$. La taille $taille_b$ des transactions avant leur compression est égale à $taille_b = arite * f$. Nous obtenons :
$$Taux = 1 - \frac{|u| + (arite - |u|) * f}{arite * f}$$
2. Si la compression est obtenue en utilisant l'ensemble des super-motifs fréquents de u, la taille $taille_a$ des transactions après leur compression est obtenue comme suit : $taille_a = taille_{sup}$ + $taille_{sub}$ + $taille_{tcmp}$, avec
   — $taille_{sup}$ est la taille des super-motifs de u utilisés pour la compression : $taille_{sup}$ = $nbrCh * (|u| + 1)$;
   — $taille_{sub}$ est la taille de toutes les sous-tables obtenues après l'utilisation des super-motifs de u pour la compression : $taille_{sub}$ = $(arite - (|u| + 1)) * sum_f$;
   — $taille_{tcmp}$ correspond à la taille après la compression de toutes les transactions qui ne sont couvertes par aucun super-motif de u : $taille_{tcmp} = (f - sum_f) * |u| + |u|$.
   Nous obtenons :
$$ChTaux = 1 - \frac{taille_{sup} + taille_{sub} + taille_{tcmp}}{arite * f} \quad (2)$$

		Datasets					
		Modified Renault	Renault	ukVg	ogdVg	Rand10-20-10	wordVg
ins_{res}	FPTCM$^+$	44/50	2/2	23/65	17/65	20/20	33/65
	FPTCM	44/50	2/2	19/65	17/65	20/20	33/65
T_{cmp}	FPTCM$^+$	6.8	**5.58**	**2.28**	**13.37**	**5.2**	**0.77**
	FPTCM	6.7	8.62	11.41	19.7	9.35	1.09
T_{rs}	FPTCM$^+$	**217.52**	**0.68**	**146.71**	**52.41**	**3.4**	**31.97**
	FPTCM	279.58	1.21	187.04	70.2	6.48	46.96
T_{tot}	FPTCM$^+$	**224.34**	**6.26**	**148.99**	**65.78**	**8.6**	**32.74**
	FPTCM	286.28	9.83	198.45	89.9	15.83	48.05
$Rate$	FPTCM$^+$	**79.18**	**80.08**	**28.24**	**38.45**	**18.93**	**22.67**
	FPTCM	37.06	37.69	21.97	23.7	18.07	18.84

TAB. 2: Comparaison entre FPTCM$^+$ et FPTCM.

L'utilisation du taux de compression nous permet d'évaluer le gain réel que peut offrir la compression de f transactions en exploitant le motif u par rapport au gain que peut offrir les super-motifs de u, car nous considérons le même nombre de transactions dans les deux cas.

4.2 Résolution des CSP définis sur des contraintes table compressées

La structure de la relation compressée est identique à celle proposée dans Gharbi et al. (2014). Pour résoudre le CSP compressé, nous exploitons l'heuristique $MaxDeg$ Dechter et Itay (1994) pour ordonner les variables du CSP et l'algorithme $Graph{-}based{-}backjumping$ Dechter et Frost (2002) (GBJ) pour le choix de la variable. La différence entre FPTCM et FPTCM$^+$ est juste au niveau de l'étape de compression.

5 Résultats expérimentaux

Les expériences ont été effectuées sur un Intel Core i5 2.5 GHz. Nous comparons FPTCM$^+$ à FPTCM sur les mêmes benchmarks [2]. Les deux méthodes ont été implémentées en Java et utilisent la même méthode de résolution. Dans nos expériences, nous avons fixé la valeur de S_{min} à 2 et nous avons considéré, dans le processus de compression, uniquement les relations ayant au minimum 10 tuples et ayant une arité supérieure à 2. Les temps sont donnés en secondes. La Table 2 donne les résultats de cette comparaison. Pour chaque méthode et pour chaque benchmark, nous rapportons le nombre d'instances résolues en un temps maximal de 1 heure (ins_{res}), le temps CPU moyen de compression d'une instance (T_{cmp}), le temps CPU moyen de résolution d'une instance (T_{rs}), le temps CPU global moyen pour les deux étapes ($T_{tot} = T_{cmp} + T_{rs}$), et le taux moyen de compression ($Taux$). Le taux de compression moyen d'une instance est la taille des relations après compression sur la taille des relations d'origines. Pour chaque benchmark, le taux de compression moyen est calculé en divisant la somme des taux de compression de chaque instance par le nombre total d'instances compressées. Le même nombre d'instances est résolu par les deux méthodes à l'exception du benchmark crossword-uk-vg où FPTCM$^+$ a résolu 4 instances de plus. On peut également remarquer que FPTCM$^+$

2. datasets disponibles sur https://www.cril.univ-artois.fr/~lecoutre/#/benchmarks

prend moins de temps pour compresser les différents benchmarks. Le temps moyen requis pour résoudre les benchmarks avec FPTCM$^+$ est inférieur au temps requis pour résoudre les mêmes benchmarks avec FPTCM. Ces résultats démontrent la supériorité de notre appropche

6 Conclusion

Dans ce papier, nous avons proposé une approche basée sur les techniques de fouille de données pour la compression des contraintes table. Notre approche améliore FPTCM de deux manières : (i) utilisation de la couverture d'un motif dans le FP-Tree au lieu de la fréquence, (ii) utilisation du taux de compréssion pour le calcul des motifs fréquents nécessaires à la compression au lieu du calcul du gain que peut offrir un motif, permettant ainsi une compression plus élevée et une résolution plus efficace du CSP en identifiant des ensembles de motifs fréquents plus précis. Les résultats experimentaux sont encourageants.

Références

Dechter, R. et D. Frost (2002). Backjump-based backtracking for constraint satisfaction problems. *Artificial Intelligence 136*, 147–188.

Dechter, R. et M. Itay (1994). Experimental evaluation of preprocessing algorithms for constraint satisfaction problems. *Artificial Intelligence 68*, 211–241.

Gharbi, N., F. Hemery, C. Lecoutre, et O. Roussel (2014). Sliced table constraints: Combining compression and tabular reduction. In *International Conference on AI and OR Techniques in Constraint Programming Combinatorial Optimization Problems*, Volume 14, pp. 679–696.

Han, J., J. Pei, Y. Yin, et R. Mao (2004). Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data mining and knowledge discovery 8*, 53–87.

Jabbour, S., S. Roussel, L. Sais, et Y. Salhi (2015). Mining to compress table constraints. In *IEEE 27th International Conference ICTAI*, pp. 405–412.

Montanari, U. (1974). Networks of constraints: Fundamental properties and applications to picture processing. *Information sciences 7*, 95–132.

Summary

In this paper, we propose an improvement of the compression step of sliced table method proposed for compressing table constraints. We propose to use of the frequency and the cover of an itemset in the FP-tree to improve the construction step of the resulting compressed tables. Moreover, we propose to exploit the compression rate metric instead of savings to compute frequent itemsets relevant for compression. This allows higher compression. Experimental results show the effectiveness and efficiency of our approach.