# Autour de la décomposition des contraintes non-binaires en contraintes binaires équivalentes *

**Achref El Mouelhi**
Aix-Marseille Université, LSIS UMR 7296
Avenue Escadrille Normandie-Niemen
13397 Marseille Cedex 20 (France)
achref.elmouelhi@lsis.org

**Résumé**

Des efforts considérables en recherche ont été déployés pour la conversion d'une instance CSP non-binaire en une instance CSP binaire équivalente. Ces travaux peuvent être subdivisés en deux. Les premiers ont été consacrés à l'étude du codage binaire des instances non-binaires. Trois codages ont été proposés à savoir le codage dual, le codage par variables cachées et le codage double. Malheureusement, ces codages ne permettent pas d'utiliser des propriétés et des résultats intéressants restrictifs au cas binaire. Les deuxièmes consistent à transformer chaque contrainte non-binaire en un ensemble de contraintes binaires, l'instance obtenue est appelée primale. Malheureusement, cette transformation ne préserve pas la satisfiabilité.

Dans cet article, nous proposons deux conditions dont la présence garantit de pouvoir remplacer une contrainte non-binaire en un ensemble de contraintes binaires, tout en préservant la satisfiabilité. Une étude expérimentale prouve que notre approche n'est pas artificielle puisque certains benchmarks ternaires peuvent être transformés en instances binaires équivalentes et par la suite être efficacement résolues par des algorithmes de l'état de l'art comme MAC.

**Abstract**

Considerable research efforts have been focused on the translation of non-binary CSP into an equivalent binary CSP. Most of this work was devoted to studying the binary encoding of non-binary CSP. Three encodings have been proposed namely dual encoding, hidden variable encoding and double encoding. Unfortunately, such encodings do not allow to use some properties and interesting results defined only for the binary case. Another work consists in transformating each non-binary constraint into a set of binary constraints, the obtained CSP is called primal. Unluckily, this transformation does not preserve satisfiability.

In this paper, we will propose some conditions, if they hold, a non-binary constraint can be decomposed into a set of binary constraints while preserving satisfiability. An experimental study proves that our approach is not artificial since some ternary benchmarks can be transformed into equivalent binary instances and effectively solved by MAC.

*Ce travail est soutenu par l'Agence Nationale de la Recherche dans le cadre du projet TUPLES (ANR-2010-BLAN-0210).

## 1 Introduction

Le *problème de satisfaction de contraintes* (CSP, [21]) constitue un formalisme important pour exprimer et résoudre efficacement plusieurs problèmes du monde réel. La majorité de ces problèmes s'expriment sous la forme d'instances CSP d'arité quelconque. Théoriquement, il est bien connu que toute instance d'arité quelconque peut être transformée en temps polynomial en instance binaire. Pour ce fait, nous avons principalement deux approches : soit en utilisant un des codages binaires connus tel que le codage dual [8], le codage par variable cachée [9] et le codage double [24] ou en convertissant (on dit aussi en décomposant) chaque contrainte non-binaire en un ensemble de contraintes binaires [8].

La première approche consiste à définir des nouvelles contraintes binaires sans convertir les contraintes originelles. Elle est basée sur les codages binaires, inspirés des représentations graphiques des instances non-binaires, pour obtenir une instance binaire équivalente sans décomposer aucune contrainte originelle (ou sa relation associée). Malheureusement, aucun de ces co-

dages n'a permis d'utiliser explicitement certaines propriétés intéressantes, comme la substitution et l'interchangeabilité [14] ou quelques autres qui portent sur l'identification de classes polynomiales ou l'application de certaines cohérences. En effet, appliquer certains niveaux de cohérences [25, 1] ou prouver l'appartenance à certaines classes polynomiales [6] de certaines instances non-binaires est souvent NP-difficile, ceci est peut être dû à la non-conversion de contraintes non-binaires. La seconde approche vise à décomposer toutes les contraintes non-binaires en un ensemble de contraintes binaires. Naturellement, cette approche n'a pas été développée par la suite vu qu'elle ne préserve pas la satisfiabilité.

Récemment, certains travaux ont montré l'intérêt de l'utilisation d'une version binaire des instances non-binaires d'un point de vue graphique [11]. En effet, la microstructure [19] d'une instance non-binaire exige l'utilisation des hypergraphes alors qu'une instance binaire peut être représentée par un graphe simple. Dans la littérature, la théorie des graphes s'avère être plus riche que celle des hypergraphes et l'utilisation de ces derniers semble être plus compliquée que les graphes. En plus, il est plus simple de calculer certains paramètres graphiques comme la largeur ou de vérifier quelques propriétés telle que l'acyclicité d'un graphe plutôt qu'un hypergraphe. Par ailleurs, certains autres travaux ont prouvé que l'extension de quelques classes polynomiales aux instances non-binaires peut être efficacement réalisée en utilisant les microstructures basées sur les codages binaires [10]. Cette tâche reste difficile à accomplir avec la définition de Cohen [3] qui s'appuie sur le complément d'un hypergraphe et qui ne se réfère pas aux codages binaires. En fait, cette notion pose plusieurs questions dont la principale consiste à savoir s'il faudrait considérer toutes les hyperarêtes qui correspondent aux relations universelles, à l'image de la notion de complémentaire de graphe dans le cas binaire. Mais dans ce cas, la taille de l'hypergraphe serait potentiellement exponentielle en fonction de la taille de l'instance.

Dans ce papier, nous nous intéressons à la conversion des contraintes non-binaires en contraintes binaires en préservant la satisfiabilité. Pour cela, nous allons dans un premier temps nous référer à la théorie des bases de données relationnelles pour définir une première règle pour la décomposition d'une contrainte en un ensemble de contraintes binaires sans perte de satisfiabilité. Pour le cas ternaire (instance d'arité trois), nous montrerons que chaque contrainte satisfaisant cette règle peut être remplacée par deux contraintes binaires. Dans un second temps, nous allons introduire une nouvelle règle, différente de la première, pour décomposer aussi les contraintes non-binaires, en préservant la cohérence. Pour le cas ternaire, une telle contrainte sera décomposée en trois contraintes binaires. Une partie de ce travail sera consacrée à étudier des propriétés autour de la décomposition, comme la complexité et la relation avec certaines classes polynomiales existantes.

Dans la section suivante, nous rappelons certaines définitions et notations nécessaires qui seront par la suite utilisées dans le reste de ce papier. Dans les sections 3 et 4, nous proposerons les deux règles permettant la décomposition des contraintes sans modifier la cohérence du problème de départ. En plus de certains résultats théoriques, notre étude est accompagnée de quelques résultats expérimentaux prouvant l'applicabilité de notre approche et certains liens avec les classes polynomiales. Avant de conclure, nous appliquons nos règles sur certaines contraintes bien particulières tel que les contraintes globales.

## 2 Préliminaires

Le *problème de satisfaction de contraintes* constitue un formalisme important pour exprimer et résoudre efficacement plusieurs problèmes réels en intelligence artificielle et en recherche opérationnelle.

Formellement, une *instance CSP* est définie comme suit :

**Définition 1 (instance CSP)** *Une instance CSP est un triplet $I = (V, D, C)$, où $V = \{V_1, ..., V_n\}$ est un ensemble fini de n* ***variables****, $D = \{D_1, ..., D_n\}$ est un ensemble fini de* ***domaines*** *contenant au plus d* ***valeurs****, un pour chaque variable et $C = \{C_1, ..., C_e\}$ est un ensemble de e* ***contraintes****. Chaque contrainte $C_i$ est un couple $(S(C_i), R(C_i))$ avec :*

- *$S(C_i) = \{V_{i_1}, ..., V_{i_{n_i}}\} \subseteq V$, la* ***portée*** *de la contrainte,*
- *$R(C_i) \subseteq D_{i_1} \times ... \times D_{i_{a_i}}$, la* ***relation*** *qui autorise r tuples (compatibilité de valeurs).*

Nous supposons que toute variable apparaît au moins dans la portée d'une contrainte. $|S(C_i)|$ est l'**arité** de la contrainte $C_i$ (c'est-à-dire, le nombre de variables sur lesquelles porte la contrainte $c_i$) et elle sera notée $a_i$ . Si la contrainte est d'arité deux, elle est dite binaire et elle sera notée $C_{ij}$ avec $S(C_{ij}) = \{V_i, V_j\}$. Si toutes les contraintes d'une instance $I$ sont binaires alors $I$ est dite **binaire**. Sinon (cas général), $I$ est dite **non-binaire** (ou **d'arité quelconque**). Nous signalons que le cas ternaire est évoqué quand toutes les contraintes sont d'arité inférieure ou égale à trois.

Nous continuons avec les deux notations suivantes qui seront nécessaires pour la suite :

**Notation 1 (projection d'un(e) tuple/relation)** *Étant donnés une contrainte $C_i$, un tuple $t_i \in R(C_i)$*

*et un ensemble de variables* $\{V_{i_1}, ..., V_{i_k}\} \subseteq S(C_i)$ *:* $t_i[\{V_{i_1}, ..., V_{i_k}\}] = (v_j \in t_i \mid V_j \in \{V_{i_1}, ..., V_{i_k}\})$ *est la* ***projection*** *du tuple* $t_i$ *sur* $\{V_{i_1}, ..., V_{i_k}\}$. $R(C_i)[\{V_{i_1}, ..., V_{i_k}\}] = \{t_i[\{V_{i_1}, ..., V_{i_k}\}] \mid t_i \in R(C_i)\}$ *est la projection de* $R(C_i)$ *sur* $\{V_{i_1}, ..., V_{i_k}\}$.

**Notation 2 (restriction d'une contrainte)**
*Étant donnée une contrainte* $C_i$, $C_i[\{V_{i_1}, ..., V_{i_k}\}]$ *est la restriction de* $C_i$ *sur* $\{V_{i_1}, ..., V_{i_k}\}$ $(\subseteq S(C_i))$. *Si nous notons* $C_\ell = C_i[\{V_{i_1}, ..., V_{i_k}\}]$, *alors* $S(C_\ell) = \{V_{i_1}, ..., V_{i_k}\}$ *et* $R(C_\ell) = R(C_i)[\{V_{i_1}, ..., V_{i_k}\}]$.

Étant donnée une instance $I$, la question fondamentale est de décider si $I$ a une solution (une affectation d'une valeur à chaque variable de $I$ qui satisfait toutes les contraintes), problème bien connu comme étant NP-complet même pour le cas binaire.

Dans [8], les auteurs ont introduits une nouvelle méthode pour convertir une instance non-binaire en une instance binaire. Malheureusement, cette transformation ne préserve pas la satisfiabilité, c'est-à-dire l'ensemble de solutions de l'instance initiale n'est pas forcément égal à celui de l'instance transformée.

| $R(C_\ell)$ | | |
|---|---|---|
| $V_i$ | $V_j$ | $V_k$ |
| $v_i$ | $v_j$ | $v_k$ |
| $v'_i$ | $v_j$ | $v'_k$ |
| $v'_i$ | $v'_j$ | $v_k$ |

| $R(C_{ij})$ | $R(C_{jk})$ | $R(C_{ik})$ |
|---|---|---|
| $V_i\ V_j$ | $V_j\ V_k$ | $V_i\ V_k$ |
| $v_i\ v_j$ | $v_j\ v_k$ | $v_i\ v_k$ |
| $v'_i\ v_j$ | $v_j\ v'_k$ | $v'_i\ v'_k$ |
| $v'_i\ v'_j$ | $v'_j\ v_k$ | $v'_i\ v_k$ |

FIGURE 1 – Une relation associée à une contrainte non-binaire $R(C_\ell)$ transformée en trois contraintes dont les relations sont $R(C_{ij})$, $R(C_{jk})$ et $R(C_{ik})$.

La figure 1 montre que les trois relations ($R(C_{ij})$,$R(C_{jk})$ et $R(C_{ik})$), obtenues après avoir décomposée la contrainte non-binaire $C_\ell$, autorise l'affectation $(v'_i, v_j, v_k)$ qui n'est pas autorisée par $R(C_\ell)$. Par conséquent, cette conversion ne préserve pas la satisfiabilité. Dans les sections suivantes, nous proposerons deux conditions pour décomposer les contraintes tout en préservant la satisfiabilité.

Pour des raisons de simplicité, nous supposons, dans la suite, que toutes les contraintes sont données en extension. Nous rappelons que toute contrainte en intension peut être transformée en une contrainte en extension.

## 3 Décomposition basée sur la dépendance multivaluée

Dans cette partie, nous illustrerons la première règle, basée sur la dépendance multivaluée, pour décomposer les contraintes non-binaires en contraintes binaires équivalentes. Dans la théorie des bases de données relationnelles, le concept de dépendance multivaluée a été initialement introduit par Fagin dans [13] pour décomposer et éliminer certaines redondances autorisées par la forme normale de Boyce Codd [2]. Du fait qu'une relation satisfait la dépendance multivaluée, elle est décomposable en deux relations sans perte de satisfiabilité. Ici, nous appliquons cette règle sur les instances non-binaires afin d'obtenir des instances binaires.

Nous commençons par une règle permettant de décomposer une contrainte $C_\ell$ en deux contraintes d'arité inférieure à $a_\ell$. Avant cela, nous énonçons la notation suivante :

**Notation 3** *Nous notons par* $V_I$ *un sous-ensemble non-vide de* $a_I$ *variables de* $V$ $(\{V_{i_1}, ..., V_{i_{a_I}}\} \mid \forall 1 \leq k \leq a_I, V_{i_k} \in V\})$. $v_I$ *note le tuple* $(v_{i_1}, ..., v_{i_{a_I}})$ *contenant une valeur pour chaque variable de* $V_I$.

Cette notation est nécessaire pour les contraintes d'arité quelconque.

**Définition 2 (dépendance multivaluée [13])**
*Une contrainte d'arité quelconque* $C_\ell$ *satisfait la dépendance multivaluée (MvD pour multivalued dependency) s'il existe trois sous-ensembles disjoints de variables* $V_I$, $V_J$ *et* $V_K$ *tel que* $S(C_\ell) = V_I \cup V_J \cup V_K$ *et* $V_I \twoheadrightarrow V_J, V_K$ *et* $\forall v_I \in R(C_\ell)[V_I], \forall v_J, v'_J \in R(C_\ell)[V_J]$ *et* $\forall v_K, v'_K \in R(C_\ell)[V_K]$ *tel que si*

- $(v_I, v_J, v_K) \in R(C_\ell)$
- $(v_I, v'_J, v'_K) \in R(C_\ell)$

*alors,*

- $(v_I, v'_J, v_K) \in R(C_\ell)$ *et*
- $(v_I, v_J, v'_K) \in R(C_\ell)$

*Dans ce cas, nous pouvons noter* $V_I \twoheadrightarrow V_J, V_K$. *Une instance non-binaire* $I = (V, D, C)$ *satisfait la dépendance multivaluée si chaque contrainte d'arité quelconque* $C_\ell \in C$ *satisfait MvD.*

Plus clairement, satisfaire MvD permet de décomposer une contrainte $C_\ell$ en deux contraintes équivalentes d'arité inférieure à $a_\ell$.

**Proposition 1** *Si une contrainte d'arité quelconque* $C_\ell$ *satisfait MvD, alors elle est décomposable en deux contraintes* $C'_\ell$ *et* $C''_\ell$ *d'arité inférieure à* $a_\ell$ *en préservant la satisfiabilité.*

**Preuve :** (par l'absurde) Soit $C_\ell$ une contrainte d'arité quelconque qui satisfait MvD, nous prouvons par l'absurde que la décomposition de contraintes MvD préserve la satisfiabilité. Donc, nous supposons qu'il existe une affectation $\mathcal{A}$ qui ne viole aucune des nouvelles contraintes (obtenues après décomposition) sans satisfaire la contrainte originelle $C_\ell$. Comme $C_\ell$ satisfait

MvD, il existe trois sous-ensembles disjoints de variables $V_I$, $V_J$ et $V_K$ tel que $S(C_\ell) = V_I \cup V_J \cup V_K$ et $V_I \twoheadrightarrow V_J, V_K$. Ainsi, nous pouvons exprimer $\mathcal{A}$ comme $(v_I, v_J, v_K)$. Comme $\mathcal{A}$ viole $C_\ell$, nous devons évidemment avoir deux tuples $t_1$ et $t_2$ appartenant à $R(C_\ell)$ tel que

- $t_1[V_I] = v_I$, $t_1[V_J] = v_J$ et $t_1[V_K] = v'_K$,
- $t_2[V_I] = v_I$, $t_2[V_J] = v'_J$ et $t_2[V_K] = v_K$.

Nous précisons que $v_K$ doit absolument être différent de $v'_K$ (pareillement pour $v_J$ et $v'_J$) sinon $\mathcal{A}$ ne viole pas $C_\ell$. Dans cette direction, nous avons

- $(v_I, v_J, v'_K) \in R(C_\ell)$
- $(v_I, v'_J, v_K) \in R(C_\ell)$

Or, par définition de MvD, nous aurons

- $(v_I, v_J, v_K) \in R(C_\ell)$ (1) et
- $(v_I, v'_J, v'_K) \in R(C_\ell)$ (2).

Finalement, (1) contredit notre hypothèse. Par conséquent, la décomposition de contraintes MvD préserve la satisfiabilité. □

Sémantiquement, une contrainte $C_\ell$ est décomposable s'il existe deux sous-ensembles de variables non-disjoints $X, Y \subsetneq S(C_\ell)$ tel que $R(C_\ell)$ est la jointure de $R(C_\ell)[X]$ et $R(C_\ell)[Y]$.

**Exemple 1** *Cet exemple illustre le cas d'une contrainte $C_\ell$ d'arité quatre et sa relation associée. En considérant $V_I = \{V_i, V_m\}$, $V_J = \{V_j\}$ et $V_K = \{V_k\}$, $C_\ell$ satisfait MvD, donc elle est décomposable en deux contraintes $C'_\ell$ et $C''_\ell$. Les tables ci-dessous représentent les relations $R(C_\ell)$, $R(C'_\ell)$ et $R(C''_\ell)$.*

| $R(C_\ell)$ | | | |
|---|---|---|---|
| $V_i$ | $V_m$ | $V_j$ | $V_k$ |
| $v_i$ | $v_m$ | $v_j$ | $v'_k$ |
| $v_i$ | $v_m$ | $v'_j$ | $v_k$ |
| $v_i$ | $v_m$ | $v'_j$ | $v'_k$ |
| $v_i$ | $v_m$ | $v_j$ | $v_k$ |
| $v'_i$ | $v'_m$ | $v_j$ | $v_k$ |

| $R(C'_\ell)$ | $R(C''_\ell)$ |
|---|---|
| $V_i\ V_m\ V_j$ | $V_i\ V_m\ V_k$ |
| $v_i\ v_m\ v_j$ | $v_i\ v_m\ v_k$ |
| $v_i\ v_m\ v'_j$ | $v_i\ v_m\ v'_k$ |
| $v'_i\ v'_m\ v_j$ | $v'_i\ v'_m\ v_k$ |

Pour le cas ternaire, satisfaire la dépendance multivaluée garantie la décomposition de la contrainte en deux contraintes binaires sans perte de satisfiabilité.

D'une façon générale, être MvD permet la décomposition d'une contrainte d'arité quelconque en deux contraintes d'arité inférieure à celle de la contrainte originelle et qui ne sont pas forcément toutes les deux binaires. Pour cela, nous définissons une nouvelle forme récursive de la dépendance multivaluée qui permettra de décomposer les contraintes d'arité quelconque en un ensemble de contraintes binaires équivalentes.

**Définition 3 (dépendance multivaluée décrémentale)** *Une contrainte $C_\ell$ d'arité $a_\ell \geq 3$ satisfait la dépendance multivaluée décrémentale (DMvD pour decremental multivalued dependency) si*

1. *il existe trois sous-ensembles disjoints de variables $V_I$, $V_J$ et $V_K$ tel que $S(C_\ell) = V_I \cup V_J \cup V_K$ et $C_\ell$ satisfait MvD*
2. *$a_I + a_J \geq 3$ alors $C_\ell[V_I \cup V_J]$ satisfait DMvD.*
3. *$a_I + a_K \geq 3$ alors $C_\ell[V_I \cup V_K]$ satisfait DMvD.*

*Une instance non-binaire $I = (V, D, C)$ satisfait la dépendance multivaluée décrémentale si chaque contrainte non-binaire $C_\ell \in C$ est DMvD.*

Nous devons maintenant montrer que satisfaire DMvD permet à la contrainte d'être remplacée par un ensemble de contraintes binaires équivalentes. Intuitivement, et d'après la définition précédente, si une contrainte $C_\ell$ satisfait DMvD, elle sera récursivement décomposée selon la proposition 1 en commençant par décomposer $C_\ell$ en deux contraintes équivalentes d'arité inférieure à celle de $C_\ell$ et nous continuons avec les nouvelles contraintes jusqu'à l'obtention de contraintes binaires. Le théorème suivant montre que cette règle de décomposition préserve la satisfiabilité.

**Théorème 1** *Si une contrainte $C_\ell$ est DMvD, alors elle est décomposable en un ensemble de contraintes binaires sans perte de satisfiabilité.*

**Preuve :** (par induction) Il est clair que DMvD est définie récursivement en respectant MvD. Donc, nous allons prouver par induction que quelles que soient la contrainte $C_\ell$ et son arité $a_\ell$, si $C_\ell$ satisfait DMvD alors elle sera décomposée en un ensemble de contraintes binaires sans perte de satisfiabilité.

- **cas de base :** pour $a_\ell = 3$, nous pouvons constater d'après le théorème 1 que si une contrainte ternaire satisfait MvD, alors elle est décomposable en deux contraintes d'arité inférieure (obligatoirement 2) sans perte de satisfiabilité.
- **hypothèse d'induction :** nous supposons que les contraintes DMvD avec une arité $a_\ell \leq p - 1$ sont décomposables en un ensemble de contraintes binaires sans perte de satisfiabilité.
- **cas inductif :** nous prouvons que c'est le cas aussi quand $a_\ell = p$. Comme $C_\ell$ est DMvD, elle est donc décomposable en deux contraintes $C'_\ell$ et $C''_\ell$ sans perte de satisfiabilité, chacune a une arité inférieure à $p$. Nous avons aussi supposé que lorsqu'une contrainte satisfait DMvD et son arité est inférieure à $p$, elle est décomposable en un ensemble de contraintes binaires en préservant la satisfiabilité.

DMvD préserve la satisfiabilité quelle que soit l'arité de la contrainte. □

Nous étudions maintenant la complexité de vérification de cette propriété et nous montrons qu'elle est

polynomiale seulement quand les trois sous-ensembles disjoints $V_I, V_J$ et $V_K$ sont donnés à l'avance.

**Proposition 2** *Étant donnés une contrainte $C_\ell$ et trois sous-ensembles disjoints de variables $V_I$, $V_J$ et $V_K$ tel que $S(C_\ell) = V_I \cup V_J \cup V_K$, vérifier si $C_\ell$ satisfait DMvD par rapport aux trois sous-ensembles disjoints $V_I$, $V_J$ et $V_K$ peut être réalisé en temps polynomial.*

**Preuve :** Pour trois sous-ensembles disjoints de variables $V_I, V_J$ et $V_K$ donnés tel que $S(C_\ell) = V_I \cup V_J \cup V_K$, nous aurons besoin de $O(a_\ell.r^2.log(r))$ pour vérifier si la contrainte $C_\ell$ satisfait DMvD.

- $r^2$ pour énumérer les tuples,
- $log(r)$ pour tester l'appartenance des tuples à la relation associée à la contrainte,
- $a_\ell$ ou plus précisément $a_\ell - 2$ pour appliquer récursivement le test sur les nouvelles contraintes.

La complexité de la décomposition est $O(r.a_\ell^2)$ :

- $r$ pour parcourir les tuples de la contrainte $C_\ell$,
- $a_\ell^2$ ou plus précisément $a_\ell(a_\ell-1)/2$ qui correspond au nombre maximal de contraintes binaires.

Par conséquent, vérifier si une contrainte satisfait DMvD et par la suite la décomposer est réalisable en temps polynomial. □

La contrainte $C_\ell$ de l'exemple 1 satisfait DMvD parce que $C'_\ell$ et $C''_\ell$ sont d'arité trois et sont MvD.

Pour le cas ternaire, cette propriété peut évidemment être vérifiée en temps polynomial et il existe un algorithme (Algorithme 1) en $O(er^2.log(r))$ pour vérifier si une instance ternaire $I$ pourrait être transformée en une instance binaire équivalente en respectant la propriété MvD. La décomposition d'une contrainte MVD nécessite un temps linéaire $O(r)$ ($O(er)$ pour décomposer toutes les contraintes).

**Algorithm 1:** Tester si les contraintes d'une instance ternaire sont décomposables

```
function TESTER_DECOMPOSITION(I = (V, D, C) : CSP) : Booléen
    foreach C_ℓ ∈ C avec S(C_ℓ) = {V_i, V_j, V_k} do
        if (not TESTER_CONTRAINTE(C_ℓ,V_i, V_j, V_k)) et
    (not TESTER_CONTRAINTE(C_ℓ,V_j, V_i, V_k)) et (not
    TESTER_CONTRAINTE(C_ℓ,V_k, V_j, V_i)) then
            return faux
    return vrai
end function
function TESTER_CONTRAINTE( C_ℓ : Contrainte, V_i, V_j, V_k :
Variable) : Booléen
for t_ℓ, t'_ℓ ∈ R(C_ℓ) avec t_ℓ[{V_i}] = t'_ℓ[{V_i}] do
        if ((t_ℓ[{V_j}] ≠ t'_ℓ[{V_j}]) et (t_ℓ[{V_k}] ≠ t'_ℓ[{V_k}]))
    then
            if (((t_ℓ[{V_i}], t'_ℓ[{V_j}], t_ℓ[{V_k}]) ∉ R(C_ℓ)) ou
        ((t_ℓ[{V_i}], t_ℓ[{V_j}], t'_ℓ[{V_k}]) ∉ R(C_ℓ))) then
                return faux
    return vrai
end function
```

Nous pouvons immédiatement constater qu'une nouvelle classe polynomiale pourrait être définie pour les instances ternaires bivalentes [1].

**Proposition 3** *La classe des instances ternaires bivalentes qui satisfont MvD est polynomiale.*

**Preuve :** Dans [15, 4], une classe polynomiale pour CSP est un ensemble (fini ou infini) d'instances pour lesquelles il existe deux algorithmes de complexité polynomiale, un premier pour la reconnaissance d'instances et un second pour les résoudre. Comme nous l'avons mentionné auparavant, $O(er^2.log(r))$ est suffisant pour vérifier si une instance ternaire satisfait la MvD. Pour la résolution, nous commençons par convertir l'instance en instance binaire, ce qui nécessite $O(er)$ en temps. Ensuite, nous savons que les instances binaires bivalentes définissent une classe polynomiale [7] dont la cohérence de chemin est une procédure de décision. Donc, il faut $O(n^3d^3)$ pour appliquer PC-4 [17] afin d'avoir une instance globalement cohérente [7]). □

Nous avons effectué une étude expérimentale sur 701 instances ternaires de la la compétition CP [2] pour tester l'applicabilité de notre approche en pratique. Finalement, nous avons obtenu un total de résultats pour 581 instances, nous précisons que nous avons fixé une durée d'une heure pour chaque benchmark et que notre bibliothèque ne couvre pas les instances avec des contraintes globales. 72 benchmarks ont été détectés comme étant MvD parmi lesquels nous pouvons citer les familles `pseudo/primeDimacs`, `pseudo/par`, `pseudo/garden` et `pseudo/aim`. Nous notons que tous ces benchmarks appartiennent à la classe polynomiale définie dans la proposition 3. De plus, la version transformées de ces benchmarks est mieux résolue par MAC[22] que celui de la version ternaire (y compris le temps de décomposition).

La table 1 montre quelques autres résultats des benchmarks donts les contraintes ne respectent pas toutes MvD. Les notations suivantes signifient :

- Family : nom de famille de benchmarks
- N : nombre d'instances ternaires testées de cette famille
- n : nombre de variables
- e/e' : nombre de contraintes / nombre de contraintes ternaires
- E : nombre de contraintes décomposables.

Pour le cas général, si nous ne connaissons pas la répartition de variables en trois sous-ensembles disjoints, la vérification de cette propriété semble être plus complexe.

1. Une instance CSP est dite bivalente si la taille de tous les domaines de variables est inférieure ou égale à deux.

2. voir http ://www.cril.univ-artois.fr/CPAI08 pour plus de détails.

| Family | N | n | e/e' | E |
|---|---|---|---|---|
| ssa | 1 | 757 | 847/479 | 460 |
| aim-100 | 24 | 100 | 263/261 | 136 |
| primes-* | 16 | 100 | 46/23 | 16 |
| travellingSalesman-* | 30 | 69 | 290/23 | 1 |

TABLE 1 – Résultats expérimentaux de certains benchmarks ternaires montrant le nombre de contraintes MvD.

**Théorème 2** *Étant donnée une contrainte $C_\ell$, vérifier si $C_\ell$ satisfait DMvD est NP-complet.*

**Preuve :** Non détaillée par manque d'espace. Nous spécifions que la preuve s'appuie sur une réduction polynomiale du problème 3-SAT connu pour être NP-complet. □

Nous pouvons aussi définir un niveau plus élevé de MvD pour les contraintes non-binaires.

**Définition 4 (dépendance multivaluée forte)** *Étant donnée une contrainte $C_\ell$ avec $a_\ell \geq 3$, nous disons que $C_\ell$ satisfait la dépendance multivaluée forte (SMvD pour strong multivalued dependency) si quels que soient les trois sous-ensembles disjoints de variables $V_I$, $V_J$ et $V_K$ tel que $S(C_\ell) = V_I \cup V_J \cup V_K$, nous avons $C_\ell$ satisfait DMvD. Une instance non-binaire $I = (V, D, C)$ satisfait la dépendance multivaluée forte si chaque contrainte non-binaire $C_\ell \in C$ est SMvD.*

Revenons à l'exemple 1, la contrainte $C_\ell$ n'est pas SMvD car si on considère $V_I = \{V_j, V_k\}$, $V_J = \{V_i\}$ et $V_K = \{V_m\}$, $C_\ell$ ne satisfait pas DMvD.

Avant de conclure cette section, nous devons préciser les points suivants. Pour une contrainte $C_\ell$ avec $S(C_\ell) = V_I \cup V_J \cup V_K$, si nous n'avons pas $V_I \twoheadrightarrow V_J, V_K$ nous ne pouvons rien infirmer à propos de $V_J \twoheadrightarrow V_I, V_K$ et $V_K \twoheadrightarrow V_I, V_J$. Nous signalons également que si $V_J \twoheadrightarrow V_I, V_K$ et $V_K \twoheadrightarrow V_I, V_J$ nous ne pouvons rien confirmer pour $V_I \twoheadrightarrow V_J, V_K$.

## 4 Décomposition basée sur l'interdépendance

Dans cette section, nous présentons une nouvelle règle, que nous appelons *interdépendance*, permettant la décomposition de contraintes non-binaires sans perte de satisfiabilité. L'interdépendance est définie comme suit :

**Définition 5 (interdépendance)** *Une contrainte $C_\ell$ d'arité $a_\ell \geq 3$ satisfait l'interdépendance (ID pour interdependency) s'il existe trois sous-ensembles disjoints de variables $V_I$, $V_J$ et $V_K$ tel que $\forall v_I \in R(C_\ell)[V_I], \forall v_J, v'_J \in R(C_\ell)[V_J]$ et $\forall v_K, v'_K \in R(C_\ell)[V_K]$ si*
- $(v_I, v_J, v_K) \in R(C_\ell)$
- $(v_I, v'_J, v'_K) \in R(C_\ell)$

*alors il n'existe aucun $v'_I$ tel que*
- $(v'_I, v'_J, v_K) \in R(C_\ell)$ *ou*
- $(v'_I, v_J, v'_K) \in R(C_\ell)$

*Dans ce cas, nous disons aussi que $V_I$, $V_J$ et $V_K$ sont interdépendants (ID). Une instance $I = (V, D, C)$ satisfait l'interdépendance si chaque contrainte non-binaire $C_\ell \in C$ satisfait ID.*

Comme pour la dépendance multivaluée, une contrainte $C_\ell$ dont la portée peut être partitionnée en trois sous-ensembles disjoints de variables interdépendants est décomposable en trois contraintes d'arité inférieure à celle de la contrainte originelle.

**Théorème 3** *Si une contrainte d'arité quelconque $C_\ell$ satisfait ID, alors elle est décomposable en trois contraintes $C_\ell[V_I]$, $C_\ell[V_J]$ et $C_\ell[V_K]$ sans perte de satisfiabilité.*

**Preuve :** (par l'absurde) Soit $C_\ell$ une contrainte d'arité quelconque qui satisfait ID, nous prouvons par l'absurde que la décomposition de contraintes ID préserve la satisfiabilité. Donc, nous supposons qu'il existe une affectation $\mathcal{A}$ qui ne viole aucune nouvelles contraintes (obtenues après décomposition) sans satisfaire la contrainte originelle $C_\ell$. Donc, il existe trois sous-ensembles disjoints de variables $V_I$, $V_J$ et $V_K$ tel que $S(C_\ell) = V_I \cup V_J \cup V_K$ et $C_\ell$ satisfait ID. Dans ce cas, nous pouvons exprimer $\mathcal{A}$ comme $(v_I, v_J, v_K)$. Comme $\mathcal{A}$ viole $C_\ell$, il existe trois tuples $t_1$, $t_2$ et $t_3$ appartenant à $R(C_\ell)$ tel que
- $t_1[V_I] = v_I$, $t_1[V_J] = v_J$ et $t_1[V_K] = v'_K$ (1),
- $t_2[V_I] = v_I$, $t_2[V_J] = v'_J$ et $t_2[V_K] = v_K$ (2),
- $t_3[V_I] = v'_I$, $t_3[V_J] = v_J$ et $t_3[V_K] = v_K$ (3).

Nous précisons que $v_I$ doit être différent de $v'_I$ (pareillement pour $v_J$, $v_K$ et $v'_J$, $v'_K$). Autrement, $\mathcal{A}$ ne viole pas $C_\ell$. Dans ce cas, nous pouvons écrire
- $(v_I, v_J, v'_K) \in R(C_\ell)$
- $(v_I, v'_J, v_K) \in R(C_\ell)$

et par définition de ID, il n'existe aucun $v'_I \neq v_I$ tel que
- $(v'_I, v_J, v_K) \in R(C_\ell)$ (a)

ce qui se contredit avec (3) et donc la décomposition de contraintes ID préserve la satisfiabilité. □

**Exemple 2** *Ce deuxième exemple illustre le cas d'une contrainte $C_\ell$ d'arité quatre et sa relation associée. En considérant $V_I = \{V_i\}$, $V_J = \{V_m, V_j\}$ et $V_K = \{V_k\}$, $C_\ell$ satisfait ID, donc elle est décomposable en trois contraintes $C'_\ell$, $C''_\ell$ et $C'''_\ell$. Les tables ci-dessous représentent les quatre relations $R(C_\ell)$, $R(C'_\ell)$, $R(C''_\ell)$ et $R(C'''_\ell)$.*

| $R(C_\ell)$ | | | |
|---|---|---|---|
| $V_i$ | $V_m$ | $V_j$ | $V_k$ |
| $v_i$ | $v'_m$ | $v'_j$ | $v_k$ |
| $v_i$ | $v_m$ | $v_j$ | $v'_k$ |
| $v'_i$ | $v_m$ | $v'_j$ | $v'_k$ |

| $R(C'_\ell)$ | $R(C''_\ell)$ | $R(C'''_\ell)$ |
|---|---|---|
| $V_i\ V_m\ V_j$ | $V_m\ V_j\ V_k$ | $V_i\ V_k$ |
| $v_i\ v'_m\ v'_j$ | $v'_m\ v'_j\ v_k$ | $v_i\ v_k$ |
| $v_i\ v_m\ v_j$ | $v_m\ v_j\ v'_k$ | $v_i\ v'_k$ |
| $v'_i\ v_m\ v'_j$ | $v_m\ v'_j\ v'_k$ | $v'_i\ v'_k$ |

Malheureusement, l'interdépendance ne garantie pas la décomposition de contraintes non-binaires en contraintes binaires équivalentes. Pour cela, nous introduisons l'interdépendance décrémentale.

**Définition 6 (interdépendance décrémentale)** *Étant donnée une contrainte $C_\ell$ avec $a_\ell \geq 3$, nous disons que $C_\ell$ satisfait l'interdépendance décrémentale (DID pour decremental interdependency) si*

1. *il existe une partition $V_I$, $V_J$ et $V_K$ tel que $S(C_\ell) = V_I \cup V_J \cup V_K$ et $C_\ell$ satisfait ID*
2. *$a_I + a_J \geq 3$ alors $C_\ell[V_I \cup V_J]$ satisfait DID.*
3. *$a_I + a_K \geq 3$ alors $C_\ell[V_I \cup V_K]$ satisfait DID.*
4. *$a_J + a_K \geq 3$ alors $C_\ell[V_J \cup V_K]$ satisfait DID.*

*Une instance non-binaire $I = (V, D, C)$ satisfait l'interdépendance décrémentale si chaque contrainte non-binaire $C_\ell \in C$ est DID.*

Nous pouvons maintenant déduire que DID est une condition suffisante pour décomposer une contrainte non-binaire en un ensemble de contraintes binaires équivalentes. Comme nous l'avons mentionné dans la définition précédente, la contrainte $C_\ell$ sera récursivement décomposée en respectant la proposition 3.

**Théorème 4** *Si une contrainte $C_\ell$ satisfait DID, alors elle sera décomposée en un ensemble de contraintes binaires sans perte de satisfiabilité.*

**Preuve :** (par induction) Il est clair que DID est récursivement définie et basée sur ID. Nous prouvons par induction que quelle que soit l'arité $a_\ell$ de la contrainte $C_\ell$ la décomposition de contraintes qui satisfont DID préserve la satisfiabilité.

- **cas de base :** pour $a_\ell = 3$, nous avons montré dans le théorème 3 que la décomposition de contraintes ID préserve la satisfiabilité.
- **hypothèse d'induction :** nous supposons que la satisfiabilité est préservée pour $a_\ell \leq p - 1$.
- **cas inductif :** nous prouvons qu'elle restera préservée quand $a_\ell = p$. Comme $C_\ell$ satisfait DID, elle est donc décomposable en trois contraintes binaires $C_\ell[V_I \cup V_J]$, $C_\ell[V_J \cup V_K]$ et $C_\ell[V_I \cup V_K]$ sans perte de satisfiabilité, chacune de ses nouvelles contraintes a une arité inférieure à $p$. Nous avons supposé que lorsqu'une contrainte satisfait DID et son arité est inférieure à $p$, elle est récursivement décomposable en un ensemble de contraintes binaires sans perte de satisfiabilité.

La décomposition de contraintes DID préserve la satisfiabilité quelle que soit l'arité de la contrainte. □

Nous pouvons constater que la contrainte $C_\ell$ de l'exemple 2 est DID car les deux contraintes d'arité trois $C'_\ell$ et $C''_\ell$ satisfont évidemment ID.

Étant donnés une contrainte $C_\ell$ et trois sous-ensembles disjoints de variables $V_I$, $V_J$ et $V_K$ tel que $S(C_\ell) = V_I \cup V_J \cup V_K$, vérifier si $C_\ell$ satisfait DID par rapport à $V_I$, $V_J$ et $V_K$ peut se réaliser en temps polynomial. Par contre, dans le cas général, déterminer s'il existe trois sous-ensembles disjoints $V_I$, $V_J$ et $V_K$ tel que $S(C_\ell) = V_I \cup V_J \cup V_K$ est NP-complet.

**Proposition 4** *Étant donnés une contrainte $C_\ell$ et trois sous-ensembles disjoints de variables $V_I$, $V_J$ et $V_K$ tel que $S(C_\ell) = V_I \cup V_J \cup V_K$, vérifier si $C_\ell$ satisfait DID par rapport à $V_I$, $V_J$ et $V_K$ peut se réaliser en temps polynomial.*

**Preuve :** Similaire à la preuve de la proposition 2 □.

**Théorème 5** *Étant donnée une contrainte $C_\ell$, tester s'il existe trois sous-ensembles disjoints de variables $V_I, V_J$ et $V_K$ tel que $S(C_\ell) = V_I \cup V_J \cup V_K$ et $V_I$, $V_J$ et $V_K$ sont DID est NP-complet.*

**Preuve :** Supprimée pour manque d'espace. comme pour le théorème 2, nous devons encore spécifier que cette preuve repose sur une réduction polynomiale du problème 3-SAT bien connu pour être NP-complet. □

Contrairement à la dépendance multivaluée, l'interdépendance est symétrique, c'est-à-dire si une contrainte $C_\ell$ avec $S(C_\ell) = \{V_i, V_j, V_k\}$ est ID par rapport à la variable $V_i$, elle le sera aussi par rapport aux variables $V_j$ et $V_k$. Par conséquent, nous n'avons pas à tester la propriété pour les trois variables, il suffit d'avoir un résultat d'une variable de $S(C_\ell)$.

Pour le cas ternaire, cette propriété peut être vérifiée, pour une contrainte donnée, en temps polynomial ($O(d.r^2 log(r))$) alors que la décomposition d'une telle contrainte peut se réaliser en temps linéaire ($O(r)$). Ceci nous conduit à la proposition suivante.

**Proposition 5** *La classe d'instances ternaires bivalentes qui satisfont l'interdépendance est polynomiale.*

**Preuve :** Similaire à la preuve de la proposition 3. □

D'un point de vue expérimental, tous les benchmarks qui satisfont MvD sont aussi ID en plus de certaines autres instances de la famille `primes-20` et `primes-30`. Toutes ces instances satisfont soit la classe polynomiale décrite dans la proposition 5, soit BTP [3]

3. Une instance binaire $I$ satisfait **Broken Triangle Property (BTP)** par rapport à un ordre sur les variables $<$ si, pour tout triplet de variables $(V_i, V_j, V_k)$ tel que $V_i < V_j < V_k$, si $(v_i, v_j) \in R(C_{ij})$, $(v_i, v'_k) \in R(C_{ik})$ et $(v_j, v''_k) \in R(C_{jk})$, alors soit $(v_i, v''_k) \in R(C_{ik})$, soit $(v_j, v'_k) \in R(C_{jk})$.

| Family | N | n | e/e' | E |
|---|---|---|---|---|
| ssa | 1 | 757 | 847/479 | 273 |
| pret | 8 | 105 | 70/70 | 35 |
| dubois | 13 | 98 | 65/65 | 64 |
| travellingSalesman-* | 30 | 69 | 290/23 | 1 |

TABLE 2 – Résultats expérimentaux sur quelques benchmarks ternaires montrant le nombre de contraintes ID.

[5] ou soit DBTP [4] [10]. De plus, les instances binaires obtenues sont mieux résolues par MAC que les instances ternaires originelles.

Pour quelques instances appartenant à `dubois`, `pret` et `primes`, toutes les contraintes ternaires sauf une ont été remplacées par des contraintes binaires. Globalement, nous avons réussi à convertir 86 benchmarks de 581 considérés (ce qui représente environ 14% au total).

Maintenant, nous définissons l'interdépendance forte.

**Définition 7 (interdépendance forte)** *Étant donnée une contrainte $C_\ell$ avec $a_\ell \geq 3$, nous disons que $C_\ell$ satisfait l'interdépendance forte (SID pour strong interdependency) si pour chaque trois sous-ensembles disjoints de variables $V_I$, $V_J$ et $V_K$ tel que $S(C_\ell) = V_I \cup V_J \cup V_K$ , nous avons $C_\ell$ satisfait DID. Une instance non-binaire $I = (V, D, C)$ satisfait l'interdépendance forte si chaque contrainte non-binaire $C_\ell \in C$ satisfait SID.*

Pour l'exemple 2, la contrainte $C_\ell$ ne satisfait pas l'interdépendance forte en considérant $V_I = \{V_m, V_k\}$, $V_J = \{V_j\}$ et $V_K = \{V_i\}$.

Pour finir, nous définissons comment une contrainte pourrait être parfaitement décomposable.

**Définition 8 (décomposition parfaite)** *Une contrainte d'arité quelconque $C_\ell$ est* parfaitement décomposable *si elle satisfait l'interdépendance pour chaque trois sous-ensembles disjoints $V_I$, $V_J$ et $V_K$ tel que $S(C_\ell) = V_I \cup V_J \cup V_K$.*

Nous montrons que les contraintes parfaitement décomposables possèdent une propriété intéressante qui sera utilisée dans la suite.

**Lemme 1** *Une contrainte d'arité quelconque $C_\ell$ satisfait ID pour chaque triplet de sous-ensembles disjoints de variables $V_I$, $V_J$ et $V_K$ tel que $S(C_\ell) = V_I \cup V_J \cup V_K$, si et seulement si, $C_\ell[\{V_i, V_j, V_k\}]$ satisfait ID pour chaque triplet de variables $V_i$, $V_j$ et $V_k \in S(C_\ell)$.*

4. DBTP est une extension de BTP aux instances d'arité quelconque en utilisant le codage dual.

**Preuve :** ($\Rightarrow$) Pour une contrainte $C_\ell$, nous supposons qu'elle satisfait ID tous trois sous-ensembles disjoints de variables $V_I$, $V_J$ et $V_K$ tel que $S(C_\ell) = V_I \cup V_J \cup V_K$ alors qu'il existe $V_i$, $V_j$ et $V_k \in S(C_\ell)$ tel que $C_\ell[\{V_i, V_j, V_k\}]$ ne satisfait pas ID. Donc, il existe $v_i, v'_i \in D_i$, $v_j, v'_j \in D_j$ et $v_k, v'_k \in D_k$ tel que
- $(v_i, v_j, v_k) \in R(C_\ell)[\{V_i, V_j, V_k\}]$
- $(v_i, v'_j, v'_k) \in R(C_\ell)[\{V_i, V_j, V_k\}]$ et
- $(v'_i, v'_j, v_k) \in R(C_\ell)[\{V_i, V_j, V_k\}]$ (ou $(v'_i, v_j, v'_k) \in R(C_\ell)[\{V_i, V_j, V_k\}]$).

Dans ce cas, il est obligatoire que $v_i$ du premier tuple $(v_i, v_j, v_k)$ appartienne à $v_I$ et $v_i$ du second tuple $(v_i, v'_j, v'_k)$ appartienne à $v'_I$ qui est différent de $v_I$, autrement $V_I$, $V_J$ et $V_K$ ne sont pas ID. Pour cela, il existe $v_m, v'_m \in D_m$ ($V_m \in V_I$) tel que $v_I[\{V_i, V_m\}] = (v_i, v_m)$ et $v'_I[\{V_i, V_m\}] = (v'_i, v_m)$. Si nous considérons une autre répartition, c'est-à-dire $V_I = V_I - \{V_m\}$ et $V_J = V_J \cup \{V_m\}$, $V_I$, $V_J$ et $V_K$ ne sont pas ID ce qui est impossible car cela contredit notre hypothèse.
($\Leftarrow$) Supposons pour une contrainte donnée $C_\ell$ qu'il existe trois sous-ensembles disjoints de variables $V_I$, $V_J$ et $V_K$ (avec $S(C_\ell) = V_I \cup V_J \cup V_K$) qui ne sont pas ID bien que $V_i$, $V_j$ et $V_k$ sont ID pour tout $V_i, V_j, V_k \in S(C_\ell)$. Alors, $\exists v_I, v'_I \in R(C_\ell)[V_I]$, $v_J, v'_J \in R(C_\ell)[V_J]$, $v_K$, $v'_K \in R(C_\ell)[V_K]$ de telle façon que
- $(v_I, v_J, v_K) \in R(C_\ell)$,
- $(v_I, v'_J, v'_K) \in R(C_\ell)$ et
- $(v'_I, v'_J, v_K) \in R(C_\ell)$ (ou $(v'_I, v_J, v'_K) \in R(C_\ell)$)

$v_I \neq v'_I \Longrightarrow \exists V_i \in V_I$ tel que $v_I[\{V_i\}] \neq v'_I[\{V_i\}]$, nous notons $v_i = v_I[\{V_i\}]$ et $v'_i = v'_I[\{V_i\}]$. D'une façon similaire, nous obtenons $v_j = v_J[\{V_j\}]$, $v'_j = v'_J[\{V_j\}]$, $v_k = v_K[\{V_k\}]$ et $v'_k = v'_K[\{V_k\}]$. Ceci implique que $(v_i, v_j, v_k) \in R(C_\ell)[\{V_i, V_j, V_k\}]$, $(v_i, v'_j, v'_k) \in R(C_\ell)[\{V_i, V_j, V_k\}]$ et $(v'_i, v'_j, v_k) \in R(C_\ell)[\{V_i, V_j, V_k\}]$. Par conséquent, $V_i$, $V_j$ et $V_k$ ne sont pas ID ce qui contredit notre hypothèse. □

Nous établissons maintenant le lien entre l'interdépendance forte et la décomposition parfaite.

**Corollaire 1** *Si une contrainte $C_\ell$ satisfait l'interdépendance forte, alors elle est parfaitement décomposable.*

## 5 Comparaison entre les deux règles

Ici, nous prouvons que les deux règles de décomposition sont différentes et qu'il n'existe aucun lien entre les deux approches.

**Proposition 6** *L'interdépendance et la dépendance multivaluée sont incomparables.*

**Preuve :** Nous devons juste noter que certaines contraintes de la famille de benchmarks `dubois` sont décomposables en considérant l'interdépendance mais

pas en considérant la dépendance multivaluée. Certains autres (comme `ssa`) ont des contraintes qui sont décomposables en utilisant la dépendance multivaluée mais pas en utilisant l'interdépendance. Finalement, nous avons certains benchmarks qui sont décomposables en considérant les deux règles (comme `primes-Dimacs`). □

Nous pouvons finalement signaler que même lorsqu'une contrainte $C_\ell$ avec $S(C_\ell) = V_I \cup V_J \cup V_K$ est décomposable en considérant MvD et par rapport à $V_I$, $V_J$ et $V_K$, ceci n'impliquera rien sur la décomposition en considérant ID.

## 6 Travaux connexes

D'une part, nous pouvons constater que la notion de décomposition traitée dans ce papier pourrait être comparée à la notion de transformation introduite dans [12] sauf que cette dernière, telle qu'elle est définie, n'autorise pas la décomposition de contraintes. Certains autres travaux [16, 18] ont étudiés auparavant la décomposabilité de contraintes mais en se basant sur des règles différentes inspirées dans certains cas de la théorie des bases de données relationnelles. Nous devons aussi signaler que les propositions 3 et 5 sont forcément des cas particuliers des classes de Schaefer [23], modulo renommage des valeurs.

D'autre part, plusieurs autres travaux ont étudié la décomposition et en particulier pour les contraintes globales. Pour des raisons d'espace, nous nous limitons ici aux contraintes *All-Diff* [20]. Pour cela, nous supposons que toutes les variables possèdent le même domaine (et par conséquent $v_i = v_j = v_k$ et $v'_i = v'_j = v'_k$, etc.). Nous montrons que les contraintes All-Diff sont décomposables en considérant l'interdépendance et pas la dépendance multivaluée.

**Théorème 6** *Toute contrainte non-binaire All-Diff est parfaitement décomposable.*

**Preuve :** Pour le cas ternaire, toute contrainte $C_\ell$ avec $S(C_\ell) = \{V_i, V_j, V_k\}$ est parfaitement décomposable car si nous avons $(v_i, v'_j, v''_k) \in R(C_\ell)$ et $(v_i, v''_j, v'_k) \in R(C_\ell)$ alors il est impossible que $v'''_i \in D_i$ existe tel que $(v'''_i, v'_j, v'_k) \in R(C_\ell)$ et $(v'''_i, v''_j, v''_k) \in R(C_\ell)$ vu que $v'_j$ et $v'_k$ sont égaux et ils ne peuvent pas apparaître dans un même tuple (par définition de contrainte All-Diff). Pour le cas général, supposons qu'il est possible d'avoir une contrainte All-Diff non-binaire qui n'est pas parfaitement décomposable. D'après le lemme 1, il existe $V_i$, $V_j$, et $V_k \in S(C_\ell)$ tel que $V_i$, $V_j$ et $V_k$ ne sont pas ID. Puisque $R(C_\ell)[\{V_i, V_j, V_k\}]$ est aussi All-Diff, il en résulte que cette contrainte ternaire All-Diff est parfaitement décomposable, ce qui est impossible. □

**Théorème 7** *Les contraintes All-Diff ne sont pas décomposable en considérant la dépendance multivaluée.*

**Preuve :** Étant donnée une contrainte $C_\ell$ avec $S(C_\ell) = \{V_i, V_j, V_k\}$, si nous avons $(v_i, v'_j, v''_k) \in R(C_\ell)$ et $(v_i, v''_j, v'_k) \in R(C_\ell)$, nous devons avoir $(v_i, v'_j, v'_k) \in R(C_\ell)$ et $(v_i, v''_j, v''_k) \in R(C_\ell)$ pour que $C_\ell$ soit décomposable. Mais nous savons que ceci est impossible à cause de l'affectation d'une même valeur à des variables différentes dans un tuple (par définition des contraintes All-Diff). □

Enfin, nous signalons que certaines contraintes globales paramétrées comme AtMost et AtLeast ne satisfont ni la dépendance multivaluée ni l'interdépendance.

## 7 Conclusion

Dans ce papier, nous avons introduit, dans un premier temps, deux nouvelles règles pour tester si une contrainte non-binaire est décomposable en un ensemble de contraintes binaires (ou non-binaires d'arité inférieure à celle de la contrainte originelle) sans perte de satisfiabilité. La première règle est basée sur une notion qui a été précédemment introduite dans la théorie des bases de données relationnelles. La deuxième s'est appuyée sur le concept d'interdépendance entre les valeurs de variables appartenant à la porté de la contrainte. Notre étude est accompagnée d'une expérimentation qui montre l'applicabilité de ces deux règles sur les instances ternaires surtout qu'elles appartiennent toutes à des classes polynomiales connues. Dans un second temps, nous avons montré que la dépendance multivaluée et l'interdépendance sont différentes. Ensuite, nous les avons testées sur quelques contraintes globales avant de les comparer à certains travaux précédents. Dans le futur, plusieurs pistes méritent d'être étudiées. Tout d'abord, il faut tester si notre approche peut être appliquée sur certaines contraintes globales de l'état de l'art autre que les contraintes All-Diff. Ensuite, une deuxième piste consiste à savoir si les instances transformées en considérant une de nos deux approches sont incluses dans une classe polynomiale connue ou si elles définissent une nouvelle classe traitable.

## 8 Remerciements

Je tiens à remercier Philippe Jégou, Cyril Terrioux et les relecteurs anonymes de cet article qui ont contribué à l'amélioration de la rédaction et qui ont proposé des perspectives intéressantes pour la poursuite de ce travail.

## Références


[1] Christian Bessiere, George Katsirelos, Nina Narodytska, Claude-Guy Quimper, and Toby Walsh. Decomposition of the nvalue constraint. In *Principles and Practice of Constraint Programming - CP 2010 - 16th International Conference, CP 2010*, pages 114–128, 2010.

[2] Edgar Frank Codd. Further normalization of the data base relational model. *IBM Research Report, San Jose, California*, RJ909, 1971.

[3] David A. Cohen. A New Classs of Binary CSPs for which Arc-Constistency Is a Decision Procedure. In *Principles and Practice of Constraint Programming - CP 2003, 9th International Conference, CP 2003, Proceedings*, pages 807–811, 2003.

[4] David A. Cohen, Peter Jeavons, and Marc Gyssens. A unified theory of structural tractability for constraint satisfaction problems. *J. Comput. Syst. Sci.*, 74(5) :721–743, 2008.

[5] M. Cooper, Peter Jeavons, and Andras Salamon. Generalizing constraint satisfaction on trees : hybrid tractability and variable elimination. *Artificial Intelligence*, 174 :570–584, 2010.

[6] Martin C. Cooper, Achref El Mouelhi, Cyril Terrioux, and Bruno Zanuttini. On broken triangles. In *Principles and Practice of Constraint Programming - 20th International Conference, CP 2014*, pages 9–24, 2014.

[7] R. Dechter. Constraint Networks. In *Encyclopedia of Artificial Intelligence*, volume 1, pages 276–285. John Wiley & Sons, Inc., second edition, 1992.

[8] R. Dechter and J. Pearl. Network-based heuristics for constraint satisfaction problems. *Artificial Intelligence*, 34 :1–38, 1987.

[9] Rina Dechter. On the expressiveness of networks with hidden variables. In *Proceedings of the 8th National Conference on Artificial Intelligence. Boston, Massachusetts, 2 Volumes.*, pages 556–562, 1990.

[10] Achref El Mouelhi, Philippe Jégou, and Cyril Terrioux. A Hybrid Tractable Class for Non-Binary CSPs. In *2013 IEEE 25th International Conference on Tools with Artificial Intelligence, 2013*, pages 947–954, 2013.

[11] Achref El Mouelhi, Philippe Jégou, and Cyril Terrioux. Microstructures for csps with constraints of arbitrary arity. In *Proceedings of the Tenth Symposium on Abstraction, Reformulation, and Approximation, SARA 2013*, 2013.

[12] Achref El Mouelhi, Philippe Jégou, and Cyril Terrioux. Hidden Tractable Classes. In *26th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2014*, 2014.

[13] Ronald Fagin. Multivalued dependencies and a new normal form for relational databases. *ACM Trans. Database Syst.*, 2(3) :262–278, 1977.

[14] Eugene C. Freuder. Eliminating interchangeable values in constraint satisfaction problems. In *AAAI*, pages 227–233, 1991.

[15] G. Gottlob, N. Leone, and F. Scarcello. A Comparison of Structural CSP Decomposition Methods. *Artificial Intelligence*, 124 :343–282, 2000.

[16] M. Gyssens, P. Jeavons, and D. Cohen. Decomposing constraint satisfaction problems using database techniques. *Artificial Intelligence*, 66 :57–89, 1994.

[17] C. Han and C. Lee. Comments on Mohr and Henderson's path consistency algorithm. *Artificial Intelligence*, 36 :125–130, 1988.

[18] Peter Jeavons, David A. Cohen, and Martin C. Cooper. Constraints, consistency and closure. *Artificial Intelligence*, 101(1-2) :251–265, 1998.

[19] Philippe Jégou. Decomposition of Domains Based on the Micro-Structure of Finite Constraint Satisfaction Problems. In *Proceedings of the 11th National Conference on Artificial Intelligence, AAAI*, pages 731–736, 1993.

[20] Jean-Louis Laurière. A language and a program for stating and solving combinatorial problems. *Artificial Intelligence.*, 10(1) :29–127, 1978.

[21] U. Montanari. Networks of Constraints : Fundamental Properties and Applications to Picture Processing. *Artificial Intelligence*, 7 :95–132, 1974.

[22] D. Sabin and E. Freuder. Contradicting Conventional Wisdom in Constraint Satisfaction. In *Proc. of ECAI*, pages 125–129, 1994.

[23] Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, STOC '78, pages 216–226. ACM, 1978.

[24] K. Stergiou and T. Walsh. Encodings of Non-Binary Constraint Satisfaction Problems. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and Eleventh Conference on Innovative Applications of Artificial Intelligence, AAAI*, pages 163–168, 1999.

[25] Yuanlin Zhang and Roland H. C. Yap. Arc consistency on $n$-ary monotonic and linear constraints. In *Principles and Practice of Constraint Programming - CP 2000, 6th International Conference*, pages 470–483, 2000.