Réordonnancement de domaines dans les réseaux de contraintes

Christian Bessiere, Thierry Petit, Bruno Zanuttini

Réordonnancement de domaines dans les réseaux de contraintes*

Christian Bessiere **Thierry Petit** **Bruno Zanuttini**
LIRMM, CNRS, Université de Montpellier
LINA, École des Mines de Nantes
GREYC, Université de Caen Basse-Normandie
bessiere@lirmm.fr thierry.petit@emn.fr bruno.zanuttini@info.unicaen.fr

Résumé

Dans le but d'accélérer la résolution d'un CSP, nous nous intéressons au problème consistant à modifier l'ordre sur les valeurs qui est induit par la définition de chaque domaine. Nous discutons de l'intérêt de cette technique de reformulation et définissons les problèmes pertinents. Nous montrons qu'on peut trouver un ordre rendant un réseau de contraintes monotone (s'il en existe un) en temps polynomial, mais que le même problème est NP-difficile pour la classe des contraintes min-*closes*. Enfin, nous montrons en quoi nos résultats s'appliquent au maintien de diverses consistances aux bornes.

Abstract

So as to speed up the task of solving a CSP, we study the problem consisting of modifying the order on values which is induced by the definition of each domain. We discuss the usefulness of this reformulation technique and define the relevant problems. We show that one can find an order which makes a constraint network monotonic (if any) in polynomial time, but that the corresponding problem is NP-hard for the class of min-closed constraints. Finally, we show to what extent our results can be applied to maintaining various bound consistencies.

1 Introduction

De nombreuses classes traitables pour le problème de satisfaction de contraintes (CSP) dépendent d'un ordre total, sur les domaines des variables. En particulier, les contraintes monotones [10, 15], min-closes [9], convexes par rangée [13, 5]. De même, certaines techniques de *propagation*, comme la consistance aux bornes, sont définies en fonction de tels ordres.

L'ordre sur les domaines est considéré comme donné, mais il est souvent arbitraire. En effet, les solveurs travaillent très fréquemment sur des entiers. Les domaines sont donc implicitement ordonnés au moment où l'on renomme les valeurs du problème en valeurs entières. Par exemple, les valeurs *latin*, *maths* et *histoire* pour une variable x_i dans un problème d'emplois du temps finiront par s'appeler respectivement 0, 1 et 2 dans le domaine de x_i donné au solveur, sous-entendant l'ordre $latin < maths < histoire$.

Quand on résout une instance du CSP, il n'y a aucune raison de respecter un ordre donné sur les domaines. Si un autre ordre permet une résolution plus efficace, on a intérêt à en tirer parti. C'est l'approche que nous étudions dans cet article.

Réordonner les domaines peut transformer une contrainte quelconque en contrainte traitable facilement. Par exemple, dans notre problème d'emplois du temps vu plus haut, supposons que la variable x_i représente une option possible et qu'une variable x_j, de domaine *sciences*, *lettres* et *droit*, représente une filière. L"utilisateur pose une contrainte sur x_i et x_j imposant les options compatibles avec les filières. En lettres on peut prendre n'importe quelle option, en *droit*, on peut prendre *histoire* ou *maths*, et en *sciences* on peut prendre uniquement *maths*. Cette contrainte n'a a priori aucune sémantique remarquable. Pourtant, si les valeurs *maths*, *histoire*, *latin* sont renommées respectivement 0, 1 et 2 et *sciences*, *droit*, *lettres*, respectivement 0, 1 et 2, la contrainte posée devient la contrainte monotone $x_i \leq x_j$.

Réordonner les domaines peut rendre la propa-

*Ce travail a été réalisé dans le cadre du projet CANAR, financé par l'ANR (ANR-06-BLAN-0383-02).

gation aux bornes plus forte. Prenons la contrainte $alldiff(x_i, x_j, x_k)$ où $x_i \in \{b, c\}, x_j \in \{b, c\}, x_k \in \{a, b, c, d\}$, avec l'ordre $a < b < c < d$ sur tous les domaines. La consistance aux bornes ne supprimera aucune valeur pour cette contrainte. Or, si l'on avait eu l'ordre $b < c < a < d$, la consistance aux bornes aurait supprimé les valeurs b et c du domaine de x_k, c'est à dire exactement ce que l'arc consistance généralisée aurait supprimé. Or, les algorithmes de consistance aux bornes sont souvent moins coûteux en temps que les algorithmes d'arc consistance généralisée.

Nous nous intéressons donc au réordonnancement de valeurs dans des domaines finis pour les réseaux de contraintes. Nous définissons les problèmes algorithmiques liés, et les étudions pour les principaux langages de contraintes définis en fonction d'un ordre : les contraintes monotones, min-closes et convexes par rangée. Nous montrons que l'on peut trouver efficacement des ordres qui rendent un réseau de contraintes monotone (s'il en existe). Nous montrons en revanche que ce problème est NP-difficile pour la min-clôture. Notons que ce résultat a été montré parallèlement par Green et Cohen, avec des techniques totalement différentes [6].

Nous discutons aussi l'influence de l'ordre sur les domaines lors de la propagation de contraintes, en caractérisant les contraintes pour lesquelles la consistance aux bornes la plus faible est équivalente à l'arc consistance, quels que soient les domaines courants.

De façon générale, on peut donc voir les problèmes que nous traitons comme des problèmes de *reformulation* : étant donné un réseau de contraintes, il s'agit d'en modifier la syntaxe afin d'améliorer sa résolution. Nous insistons sur le fait que cette reformulation n'a par définition aucun impact sur l'ensemble de solutions du problème, ni *a priori* sur la façon dont elles sont présentées à l'utilisateur. D'un autre point de vue, on peut voir ces problèmes comme des problèmes d'*identification de structure*, telle que l'ont définie Dechter et Pearl [4] : il s'agit de découvrir, dans une relation ou un réseau de contraintes, une structure qui n'est pas donnée *a priori*, en l'occurrence parce que les domaines ne sont pas ordonnés convenablement.

2 Préliminaires

Contraintes Nous considérons des variables, notées généralement $x_1, x_2, \ldots$, avec des domaines finis notés $D_1, D_2 \ldots$ (ou $D_{x_1}, D_{x_2} \ldots$) et constitués de valeurs entières ($D_i \subseteq \mathbb{N}$). Une *relation* R d'arité k est un sous-ensemble de $\mathbb{N}^k$. Les tuples $(a_1, a_2, \ldots, a_k)$ seront aussi notés sous la forme $a_1 a_2 \ldots a_k$. Une *contrainte* d'arité k est un couple $C = (R, X)$, où R est une relation d'arité k et X est une séquence de k variables. La *transposée* d'une relation binaire $R = \{(a_i, a_j)\}$ est la relation $\{(a_j, a_i) \mid (a_i, a_j) \in R\}$. Le *complémentaire* C^c d'une contrainte (R, X) (relativement à des domaines $D_x, x \in X$) est la contrainte $(\Pi_{x \in X} D_x \setminus R, X)$.

Une *signature* est un couple $\Sigma = (X, D)$, où X est une séquence de variables et D une séquence de domaines, un par variable. Un *réseau de contraintes* CN sur $\Sigma = (X, D)$ est un ensemble de contraintes $C_i = (R_i, X_i)$ où pour tout i, on a $X_i \subseteq X$.

Une *affectation* t à un ensemble de variables Y (pour un ensemble de domaines D) est une fonction qui à toute variable $x \in Y$ associe une valeur $a \in D_x$. On note $t[x]$ la valeur affectée à x par t (ou $t[i]$ pour une variable x_i). Si t est une affectation à $X_j = (x_1, \ldots, x_k)$ (ou à un surensemble de X_j), on dit que t *satisfait* la contrainte $C_j = (R_j, X_j)$ si le tuple $t[1] \ldots t[k]$ est dans R_j. On note $t[x/b]$ l'affectation où $t[x]$ a été remplacé par b dans t. Si t_Y est une affectation à Y, $Y \subseteq X_j$, et $Z = X_j \setminus Y$, on note $R_j[t_Y]_Z$ la relation sur Z égale à $\{t_Z \mid t_Y t_Z \in R_j\}$, où $t_Y t_Z$ est l'affectation à X_j égale à t_Y sur Y et à t_Z sur Z. Lorsqu'il n'y a pas de confusion possible, on associe une affectation et un tuple de valeurs ; par exemple, le tuple $(a_1, \ldots, a_n)$ représentera l'affectation t avec $t[i] = a_i$ pour tout i. Une *solution* d'un réseau de contraintes CN sur $\Sigma = (X, D)$ est une affectation à X qui satisfait toutes les contraintes de CN.

Ordres sur les domaines Une *signature ordonnée* est un triplet $\Sigma = (X, D, <)$, où (X, D) est une signature avec $X = (x_1, \ldots, x_n)$, et $<$ est une séquence $(<_1, \ldots, <_n)$, où $<_i$ est un ordre total sur D_i. On considère que $(X, D, <)$ est aussi une signature ordonnée sur tout $X' \subseteq X$. Nous insistons sur le fait que $<_i$ n'est pas nécessairement l'ordre naturel de $\mathbb{N}$.

On note $\min_{<_i} D_i$ (resp. $\max_{<_i} D_i$) l'élément minimum (resp. maximum) de D_i selon $<_i$. Pour trois valeurs $a_i, b_i, c_i \in D_i$, on note $\mathrm{med}_{<_i}(a_i, b_i, c_i)$ la *valeur médiane* de a_i, b_i, c_i selon $<_i$:

$$\mathrm{med}(a_i, b_i, c_i) = \begin{cases} a_i & \text{si } c_i \leq_i a_i \leq_i b_i \text{ ou } b_i \leq_i a_i \leq_i b_i \\ b_i & \text{si } c_i \leq_i b_i \leq_i a_i \text{ ou } a_i \leq_i b_i \leq_i c_i \\ c_i & \text{si } a_i \leq_i c_i \leq_i b_i \text{ ou } b_i \leq_i c_i \leq_i a_i \end{cases}$$

Enfin, on note $[a_i..c_i]$ pour l'ensemble de toutes les valeurs entières b_i avec $a_i \leq_i b_i \leq_i c_i$ si $a_i \leq_i c_i$, ou $c_i \leq_i b_i \leq_i a_i$ si $c_i \leq_i a_i$. Par exemple, selon l'ordre $latin < maths < histoire$, on a $\min(histoire, latin) = latin$, $\mathrm{med}(latin, histoire, maths) = maths$, $\mathrm{med}(latin, histoire, histoire) = histoire$, et $[histoire..latin] = [latin..histoire] = \{latin, maths, histoire\}$.

Résolution Le *problème de satisfaction de contraintes (CSP)* consiste à produire une solu-

tion, s'il en existe une, pour un réseau de contraintes donné. Il est NP-complet en général, mais certaines classes traitables ont été identifiées. Certaines correspondent à un *langage* de relations $\mathcal{L}$: un algorithme polynomial existe pour résoudre le CSP si toutes les relations utilisées dans les contraintes sont dans $\mathcal{L}$. C'est le cas notamment des contraintes monotones [10, 15], min-closes [9], convexes par rangée [13] et convexes par rangée connexes [5], qui nous intéressent particulièrement dans cet article.

Les algorithmes classiques pour résoudre le CSP maintiennent un certain niveau de *consistance* au cours d'une recherche arborescente. Ils suppriment des valeurs inconsistantes, ce qui permet de réduire l'espace de recherche. Il est donc essentiel de distinguer les domaines *initiaux* des variables (donnés par la signature du réseau) des domaines *courants*.

Les niveaux de consistance qui nous intéressent dans cet article sont les suivants. Pour plus de détails nous renvoyons le lecteur à l'état de l'art récent [3], dont nous suivons la présentation.

Définition 1 (supports, consistances) *Soient $(X, D, <)$ une signature ordonnée, avec $X = (x_1, \ldots, x_k)$, et $C = (R, X)$ une contrainte. Un* support *pour $a_i \in D_i$ relativement à C et D est un tuple $t \in \Pi_{j=1}^{k} D_j \cap R$ avec $t[i] = a_i$; un* support aux bornes *pour $a_i \in D_i$ est un tuple $t \in \Pi_{j=1}^{k} [\min D_j .. \max D_j] \cap R$ avec $t[i] = a_i$. Le domaine D_i est :*

- arc consistant généralisé (GAC) *si toute valeur $a_i \in D_i$ a un support (a_i est aussi dite GAC),*
- range consistant (RC) *si toute valeur $a_i \in D_i$ a un support aux bornes (a_i est aussi dite RC),*
- bound(Z) consistant (BC(Z)) *si $\min_i D_i$ et $\max_i D_i$ ont un support aux bornes.*

Le problème algorithmique qui nous intéresse particulièrement est le suivant.

Problème 2 ($\mathcal{L}$-RENOMMABILITÉ) *Soit $\mathcal{L}$ une classe de réseaux de contraintes dépendante de la signature ordonnée du réseau. Alors le problème $\mathcal{L}$-*RENOMMABILITÉ *est défini comme suit.*

Entrée : *une signature $\Sigma = (X, D)$ et un réseau de contraintes CN sur Σ.*
Sortie : *s'il en existe une, une séquence $<$ d'ordres totaux sur les domaines de D telle que CN soit dans $\mathcal{L}$ pour la signature ordonnée $(X, D, <)$.*

Il est important de noter que nous nous intéressons au problème *non uniforme*, c'est-à-dire que nous étudions un problème (algorithmique) différent pour chaque classe $\mathcal{L}$.

Ainsi, nous considérons le problème d'*identification de structure* [4] pour une classe plus grande que $\mathcal{L}$, celle des réseaux pouvant être transformés en un réseau sur $\mathcal{L}$ en réordonnant les domaines. L'idée généralise donc la notion de *renommage* propositionnel [1]. Notons que les classes de réseaux de contraintes ainsi définies sont en quelque sorte « à moitié » des restrictions syntaxiques, et « à moitié » des restrictions topologiques.

3 Contraintes monotones

3.1 Définitions

Les contraintes monotones ont été introduites par Montanari [10] dans le cas binaire. Nous donnons ici une définition dans le cas n-aire. Intuitivement, une variable « croissante » est une variable x pour laquelle, étant donnée une solution d'un réseau, on a toujours une solution en augmentant la valeur de x. Une variable est alors monotone si elle est croissante ou décroissante. La définition classique, dans le cas binaire, requiert qu'une variable d'une contrainte soit croissante et l'autre décroissante (voir par exemple [14, 13]). Notre définition est plus générale.

Définition 3 (variable monotone) *Soient $\Sigma = (X, D, <)$ une signature ordonnée, et $C = (R, X)$ une contrainte. Alors $x \in X$ est dite* croissante (resp. décroissante) pour Σ dans C *si pour tout tuple $t \in R$ et toute valeur $a > t[x]$ (resp. $a < t[x]$), le tuple $t[x/a]$ est dans R. Si CN est un réseau de contraintes sur (X, D), $x \in X$ est dite* monotone pour Σ dans CN *si elle est croissante pour Σ dans toutes les contraintes de CN, ou décroissante pour Σ dans toutes les contraintes de CN.*

Définition 4 (monotone) *Soit $\Sigma = (X, D, <)$ une signature ordonnée. Un réseau de contraintes CN sur (X, D) est dit* monotone pour Σ *si toutes les variables de X sont monotones pour Σ dans CN.*

Il est important de noter qu'une variable doit être toujours décroissante ou toujours croissante. On peut alors décider efficacement si un réseau de contrainte monotone CN donné a une solution. En effet, par définition toute contrainte (R, X') non vide et monotone pour $\Sigma = (X, D, <)$ accepte le tuple t_m défini par $\forall x \in X, t_m[x] = \min D_x$ si x est décroissante dans CN et $t_m[x] = \max D_x$ si x est croissante dans CN.

En outre, même si un réseau n'est pas complètement monotone, reconnaître une contrainte $C = (R, X)$ telle que les variables de X soient monotones dans C permet une propagation efficace de C.

Exemple 5 *Dans le problème d'emplois du temps de l'introduction, x_i n'était pas monotone dans la contrainte avec x_j pour l'ordre latin $<$ maths $<$ histoire. Elle est monotone (décroissante) pour l'ordre maths $<$ histoire $<$ latin.*

3.2 Monotone-renommabilité

Nous nous intéressons maintenant au problème MONOTONE-RENOMMABILITÉ, défini comme le problème $\mathcal{L}$-RENOMMABILITÉ pour le langage $\mathcal{L}$ des contraintes monotones. Dans la suite, nous montrons que ce problème peut être résolu efficacement lorsque (i) les relations sont données en extension ou (ii) qu'elles sont toutes d'arité bornée k et données en intension, par un oracle capable de décider la consistance d'un tuple en temps $O(k)$.

Remarquons tout d'abord qu'un réseau peut être rendu monotone si et seulement s'il peut être rendu croissant (c'est-à-dire que toutes les variables y sont croissantes). L'idée principale est alors que l'on peut chercher un ordre convenable pour chaque variable indépendamment. Pour chaque variable x il faut trouver un ordre $<_x$ sur D_x selon lequel x soit croissante dans CN. Nous formalisons les contraintes à respecter par un graphe orienté, dont les sommets correspondent aux valeurs de D_x et dans lequel un arc (a, b) signifie que $a >_x b$ ne peut pas être vrai dans un ordre convenable.

Définition 6 *Soit CN un réseau de contraintes sur (X, D), et soit $x \in X$. Le* graphe des ordres sur x *relativement à CN, noté $\mathcal{G}_x(CN)$, est le graphe orienté (V, A) tel que :*
- *$V = D_x$,*
- *pour $a, b \in D_x$, on a $(a, b) \in A$ si et seulement s'il existe une contrainte (R_i, X_i) dans CN avec $x \in X_i$ et un tuple $t \in R_i$ avec $t[x] = b$ mais $t[x/a] \notin R_i$.*

La proposition suivante est immédiate. Nous en déduisons ensuite l'algorithme donné sur la figure 1 pour le cas où le réseau CN est donné en extension.

Proposition 7 *Soit CN un réseau de contraintes sur (X, D), et soit $x \in X$. Alors les ordres totaux sur D_x pour lesquels x est croissante dans CN sont exactement les ordres topologiques du graphe $\mathcal{G}_x(CN)$.*

Proposition 8 *Soit CN un réseau de contraintes donné en extension. L'algorithme 1 calcule un* monotone-*renommage de CN ou découvre qu'il n'y en a pas, en temps $O(emn^2d + nd^2)$, où n est le nombre de variables, e le nombre de contraintes, d la taille du plus grand domaine et m le nombre de tuples dans la plus grande relation.*

Algorithme 1 : Monotone-renommabilité pour les réseaux en extension

```
début
    renommage ← ∅ ;
    pour chaque variable x dans CN faire
        A_x ← ∅ ;
        G_x(CN) ← (D_x, A_x) ;
        pour (R_i, X_i) ∈ CN avec x ∈ X_i faire
            pour t ∈ R_i faire
                pour a_x ∈ D_x avec a_x ≠ t[x] faire
                    si t[x/a_x] ∉ R_i alors
                        A_x ← A_x ∪ {(a_x, t[x])}
        si G_x a un ordre topologique <_x alors
            renommage ← renommage ∪ {<_x}
        sinon
            retourner « aucun renommage »
    retourner renommage ;
fin
```

Preuve La correction de l'algorithme est assurée par la proposition 7. En ce qui concerne la complexité, pour chaque variable la construction du graphe requiert $O(emd)$ tests d'appartenance de tuples à une relation. Si on trie chaque relation préalablement en un arbre digital, en temps $O(mn)$, chaque test d'appartenance peut être effectué en temps $O(n)$. La construction du graphe requiert donc un temps $O(emdn + emn) = O(emdn)$ pour chaque variable. Le graphe obtenu contient $O(d)$ sommets et $O(d^2)$ arcs, on peut donc effectuer un tri topologique en temps $O(d^2)$, soit au final une complexité en $O(n \times (emdn + d^2))$. □

Dans le cas où toutes les relations de CN sont d'arité bornée par une constante k (et données par un oracle), on obtient également un algorithme polynomial en remplaçant l'instruction « pour $t \in R_i$ » par « pour $t \in D_1 \times \cdots \times D_\ell$ » (où $X = (x_1, \ldots, x_\ell)$), et le test « $t[x/a_x] \notin R_i$ » par un appel à l'oracle. On obtient alors la complexité $O(end^{k+1})$.

4 Contraintes min-closes

4.1 Définitions

Les contraintes min-closes ont été introduites par Jeavons et Cooper [9]. Elles généralisent les contraintes de Horn propositionnelles, et en conservent la traitabilité avec certaines hypothèses de représentation (en extension [9, corollaire 4.3] ou en CNF régulière de Horn [2, corollaire 14]). Notons que Jeavons et Cooper supposent un domaine commun à toutes les variables ; notre définition est donc un peu plus générale.

Définition 9 (min-close [9]) *Soit* $\Sigma = (X, D, <)$ *une signature ordonnée. Une contrainte* $C = (R, X)$ *est dite* min-close pour Σ *si pour toute paire de tuples* t, t' *dans* R*, le tuple* t_m *défini par* $\forall x \in X, t_m[x] = \min(t[x], t'[x])$ *est dans* R*. Un réseau de contraintes est dit* min-*clos pour* Σ *si toutes ses contraintes sont* min-*closes pour* Σ*.*

Exemple 10 *Soient* $D_1 = D_2 = D_3 = \{0, 1, 2, 3\}$*, et* $<$ *la séquence des ordres naturels sur chaque domaine. La contrainte* $C = (\{111, 221, 313\}, (x_1, x_2, x_3))$ *n'est pas* min-*close : les tuples* 221 *et* 313 *sont tels que* $t_m[x_1] = 2$*,* $t_m[x_2] = 1$ *et* $t_m[x_3] = 1$*, mais* $211 \notin \{111, 221, 313\}$*. Si l'ordre naturel est remplacé par l'ordre* $2 < 1 < 3$ *sur tous les domaines, alors* C *est* min-*close.*

4.2 Min-renommabilité

Nous nous intéressons au problème MIN-RENOMMABILITÉ, défini comme le problème $\mathcal{L}$-RENOMMABILITÉ pour la classe $\mathcal{L}$ des contraintes min-closes.

Dans le cas propositionnel, il est connu que ce problème peut être résolu en temps linéaire si chaque contrainte est donnée comme une seule clause [1, 7] ou en extension [16]. Nous montrons qu'à l'inverse, pour au moins trois valeurs dans les domaines, le problème devient NP-difficile. Ceci ferme le problème laissé ouvert par Jeavons et Cooper [9, p. 333].

Pour prouver la NP-difficulté, nous utilisons une réduction depuis le problème de satisfaisabilité (SAT) pour les formules de la forme

$$\bigwedge_{i \in I} R_{tr}(X_i, Y_i, Z_i) \wedge \bigwedge_{j \in J} X_j \neq X'_j$$

où

$$R_{tr}(X, Y, Z) \stackrel{\text{déf.}}{\equiv} ((X \wedge Y) \rightarrow Z) \wedge ((\neg X \wedge \neg Y) \rightarrow \neg Z)$$

Intuitivement, les variables X, Y, Z seront codées par $a < b$, $b < c$ et $a < c$, respectivement, et la contrainte $R_{tr}(X, Y, Z)$ imposera donc la transitivité d'un ordre sur $\{a, b, c\}$.

Le langage $\{R_{tr}, \neq\}$ n'est ni Horn, ni anti-Horn, ni affine, ni bijonctif, ni 0-valide, ni 1-valide. Le problème SAT correspondant est donc NP-complet en vertu du théorème de dichotomie de Schaefer [11].

Nous donnons tout d'abord deux lemmes qui exhibent des « gadgets » utilisés dans la réduction pour coder les contraintes propositionnelles. Les preuves sont omises par manque de place, mais une simple analyse des différents renommages candidats suffit.

Lemme 11 *Soient* ξ_1, ξ_2, ξ_3*,* $D_1 = \{\bot_1, \alpha_1, \beta_1\}$*,* $D_2 = \{\alpha_2, \beta_2\}$ *et* $D_3 = \{\alpha_3, \beta_3\}$*. Soit* $CN_\bot(\xi_1, \xi_2, \xi_3)$ *le réseau constitué des contraintes suivantes*[1] *:*

$$(\{\bot_1\ \alpha_2, \alpha_1\beta_2, \beta_1\beta_2\}, (\xi_1, \xi_2))$$
$$(\{\bot_1\ \alpha_3, \alpha_1\beta_3, \beta_1\beta_3\}, (\xi_1, \xi_3))$$
$$(\{\alpha_2\alpha_3, \alpha_2\beta_3, \beta_2\alpha_3\}, (\xi_2, \xi_3))$$

Alors il existe $\prec_2, \prec_3$ *pour lesquels* $CN_\bot(\xi_1, \xi_2, \xi_3)$ *est* min-*clos si et seulement si on a* $\bot_1 \prec_1 \alpha_1, \beta_1$*.*

Lemme 12 *Soient* x, ξ*,* $D_x = \{a, b, c\}$ *et* $D_\xi = \{\bot, \alpha, \beta\}$*. Soit* $C_{eq}[a, b, \alpha, \beta](x, \xi)$ *la contrainte :*

$$(\{a\alpha, b\beta, a\ \bot, b\ \bot, c\ \bot\}, (x, \xi))$$

Supposons $\bot \prec \alpha, \beta$*. Alors* $C_{eq}[a, b, \alpha, \beta](x, \xi)$ *est* min-*close si et seulement si on a soit* $a < b$ *et* $\alpha \prec \beta$*, soit* $a > b$ *et* $\alpha \succ \beta$*. Si* $D_x = \{a, b\}$*, la contrainte* $(\{a\alpha, b\beta, a\ \bot, b\ \bot\}, (x, \xi))$ *agit de même.*

Nous pouvons maintenant montrer que le problème est NP-difficile. Notons qu'il est trivialement dans NP si les contraintes sont données en extension. En revanche, si les contraintes sont données par un oracle, nous pouvons simplement dire qu'il est dans $\Sigma_2 P$ (un ordre candidat peut être vérifié avec un oracle coNP pour examiner toutes les paires de tuples dans chaque relation) ; rien ne garantit qu'il soit dans NP.

Proposition 13 *Le problème* MIN-RENOMMABILITÉ *est NP-difficile. Il le reste si toutes les relations sont binaires et données en extension, et si chaque domaine contient au plus trois valeurs.*[2]

Preuve Nous donnons une réduction à partir du problème SAT pour le langage $\{R_{tr}, \neq\}$. Soit φ une instance :

$$\varphi = \bigwedge_{i \in I} R_{tr}(X_i, Y_i, Z_i) \wedge \bigwedge_{j \in J} X_j \neq X'_j$$

Nous faisons tout d'abord en sorte que les ensembles de variables dans la portée des R_{tr}-contraintes soient disjoints deux à deux. Pour cela, si une variable X apparaît dans deux R_{tr}-contraintes, nous introduisons une nouvelle variable X', remplaçons l'une des deux occurrences de X par X', et ajoutons la contrainte $X = X'$ à φ. Clairement, la satisfaisabilité de φ est préservée par ces transformations.

Une fois la formule ainsi transformée, nous construisons un réseau de contraintes CN, dont les min-renommages coderont les modèles de φ.

[1] Pour simplifier les notations, nous omettons $\bot_1$ dans la notation du réseau, même si la définition de $CN_\bot$ dépend de la valeur $\bot_1$ distinguée dans le domaine de la première variable.

[2] Voir Green et Cohen [6] pour une autre preuve de ce résultat.

Pour chaque contrainte $R_{tr}(X_i, Y_i, Z_i)$ dans φ, nous introduisons dans *CN* une variable x_i de domaine $\{a_i, b_i, c_i\}$, et interprétons X_i comme $a_i < b_i$, Y_i comme $b_i < c_i$ et Z_i comme $a_i < c_i$. La contrainte initiale est ainsi codée implicitement par la transitivité de $<$ dans tout min-renommage de *CN*. Pour chaque variable X_j « restante », c'est-à-dire apparaissant dans φ dans des contraintes de différence ou d'égalité seulement, nous introduisons une nouvelle variable x_j de domaine $\{a_j, b_j\}$.

Nous codons maintenant les contraintes de différence $X_j \neq X'_j$, ainsi que les contraintes d'égalité $X = X'$ introduites lors du prétraitement. Soit une contrainte de la forme $X_j \neq X'_j$ où X_j est codée par $a_j < b_j$ pour la variable x_j de domaine $\{a_j, b_j, c_j\}$, et X'_j est codée par $a'_j < b'_j$ pour la variable x'_j de domaine $\{a'_j, b'_j, c'_j\}$ (tous les autres cas sont similaires). Nous introduisons une variable ξ_1 de domaine $\{\bot_1, \alpha_1, \beta_1\}$ et deux variables ξ_2, ξ_3 de domaines respectifs $\{\alpha_2, \beta_2\}$ et $\{\alpha_3, \beta_3\}$. Nous ajoutons alors à *CN* les contraintes du réseau $CN_\bot(\xi_1, \xi_2, \xi_3)$ du lemme 11, qui forcent $\bot_1 < \alpha_1, \beta_1$ dans tout min-renommage de *CN*. Enfin, nous ajoutons à *CN* les contraintes $C_{eq}[a_j, b_j, \alpha_1, \beta_1](x_j, \xi_1)$ et $C_{eq}[b'_j, a'_j, \alpha_1, \beta_1](x'_j, \xi_1)$ du lemme 12. La construction garantit alors que tout min-renommage de *CN* satisfait $a_j < b_j \Leftrightarrow b'_j < a'_j$, ce qui code $X_j \neq X'_j$.

Par construction, on a finalement que tout min-renommage de *CN* code un modèle de φ, et réciproquement, ce qui conclut la preuve puisque la construction est clairement réalisable en temps polynomial. □

Exemple 14 *Soit* $\varphi = R_{tr}(X, Y, Z) \wedge R_{tr}(X, Z, T) \wedge (X \neq T) \wedge (Z \neq U)$. *Nous rendons tout d'abord disjoints les ensembles de variables des* R_{tr}*-contraintes, ce qui donne :*

$$\varphi = \left\{ \begin{array}{ll} & R_{tr}(X, Y, Z) \wedge R_{tr}(X', Z', T) \\ \wedge & (X \neq T) \wedge (Z \neq U) \wedge (X = X') \wedge (Z = Z') \end{array} \right.$$

Nous introduisons alors deux variables x_1 *et* x_2*, de domaines* $D_1 = \{a_1, b_1, c_1\}$ *et* $D_2 = \{a_2, b_2, c_2\}$. *Donc* X*, par exemple, se lit* $a_1 < b_1$. *De plus, puisque* U *n'apparaît dans aucune* R_{tr}*-contrainte, nous introduisons* x_3 *de domaine* $\{a_3, b_3\}$ *;* U *se lit* $a_3 < b_3$.

Le réseau CN contient alors $CN_\bot(\xi_1, \xi_2, \xi_3)$, $C_{eq}[a_1, b_1, \alpha_1, \beta_1](x_1, \xi_1)$ *et* $C_{eq}[c_2, a_2, \alpha_1, \beta_1](x_2, \xi_1)$ *(pour coder* $X \neq T$*), ainsi que* $CN_\bot(\xi'_1, \xi'_2, \xi'_3)$, $C_{eq}[a_1, b_1, \alpha'_1, \beta'_1](x_1, \xi'_1)$ *et* $C_{eq}[a_2, b_2, \alpha'_1, \beta'_1](x_2, \xi'_1)$ *(pour coder* $X = X'$*), etc.*

On peut alors voir qu'il existe un min*-renommage de CN qui vérifie* $c_1 < a_1 < b_1$, $c_2 < a_2 < b_2$ *et* $a_3 < b_3$. *Ce renommage code donc* $\{\neg Z, X, \neg Y, \neg T, X', \neg Z', U\}$*, ce qui correspond bien à un modèle de* φ.

5 Réordonnancement de domaines et consistances aux bornes

5.1 Préliminaires sur les consistances

Nous nous intéressons maintenant à l'influence de la signature ordonnée sur les consistances aux bornes. En effet, tout comme pour les classes des contraintes monotones et min-closes, les consistances aux bornes sont influencées par l'ordre sur les domaines des variables. Nous montrons dans ce paragraphe dans quel cas cet ordre est optimal, puis dans le paragraphe 6 comment le calculer, s'il existe, ce qui reviendra à résoudre le problème $\mathcal{L}$-RENOMMABILITÉ pour une certaine classe $\mathcal{L}$ de réseaux.

Dans ce paragraphe, nous caractérisons donc les contraintes pour lesquelles quels que soient les domaines courants inclus dans les domaines initiaux, deux propagateurs donnés suppriment les mêmes valeurs des domaines. Notre travail diffère de celui de Schulte et Stuckey [12], qui étudient par exemple les combinaisons de propagateurs qui ne créent jamais de « trous » dans les domaines.

Nous appelons *propagateur* toute fonction qui, étant donné un domaine D, calcule un domaine $D' \subseteq D$. Par exemple, la range consistance pour une contrainte $C = (R, X)$ définit le propagateur qui, étant donné un domaine D, retire de chaque domaine les valeurs qui n'ont pas de support aux bornes dans D, et ce, itérativement jusqu'à stabilité. Le domaine résultant est l'image $D' \subseteq D$ de D par ce propagateur.

Définition 15 (équivalence de propagateurs) *Soit* (X, D) *une signature, et* $C = (R, X)$ *une contrainte sur* D. *Deux propagateurs* p_1, p_2 *sont dits* équivalents *sur* (X, D) *pour* C *si pour tout domaine* $D' \subseteq D$, p_1 *et* p_2 *suppriment de* D' *les mêmes valeurs.*

5.2 Convexité par rangée

Nos résultats portent sur une classe particulière de contraintes, celle des contraintes *convexes par rangée*. Ce langage a été introduit par van Beek et Dechter [13], puis Deville *et al.* [5] ont défini la sous-classe des contraintes *convexes par rangée connexes (CRC)*. Les définitions de base concernent les contraintes binaires.

Définition 16 (convexe par rangée binaire [13]) *Soit* $\Sigma = ((x_1, x_2), D, <)$ *une signature ordonnée binaire. Une contrainte* $C = (R, (x_1, x_2))$ *est* convexe par rangée pour Σ *si pour toute valeur* $a_1 \in D_1$, *l'ensemble des supports de* a_1 *dans* D_2 *pour* R *forme un intervalle de* D_2. *Un réseau de contraintes binaires est dit convexe par rangée pour* Σ *si toutes ses*

contraintes et toutes leurs transposées sont convexes par rangée pour Σ.[3]

Van Beek et Dechter montrent que si un réseau de contraintes binaires est convexe par rangée et chemin consistant, alors il est minimal et globalement consistant [13]. De plus, dans le cas binaire ils montrent qu'on peut décider efficacement s'il existe un ordre sur chacun des domaines tel qu'un réseau donné soit convexe par rangée ; en nos termes, le problème $\mathcal{L}$-RENOMMABILITÉ est traitable pour le langage $\mathcal{L}$ des contraintes convexes par rangée.

Le problème de la propriété de convexité par rangée est qu'elle n'est pas stable par application de la chemin consistance. Il s'ensuit que si un réseau est convexe par rangée mais non chemin consistant, on ne peut rien déduire en général sur l'existence d'une solution. La restriction aux contraintes *convexes par rangée connexes* (CRC) apporte une solution à ce problème.

Les contraintes CRC sont définies par Deville *et al.* comme les contraintes convexes par rangée satisfaisant de plus une certaine propriété de connexité.

Définition 17 (CRC [5]) *Soit* $\Sigma = ((x_1, x_2), D, <)$ *une signature ordonnée binaire, et soit* $C = (R, (x_1, x_2))$ *une contrainte convexe par rangée. Supposons en outre que toute valeur de* D_1, D_2 *a un support dans* D *pour* C*. Alors* C *est* connexe pour Σ *si pour toute paire de valeurs* a_1, b_1 *consécutives pour* Σ *dans* D_1*, leurs ensembles de supports respectifs* $[a_2..a'_2]$ *et* $[b_2..b'_2]$ *dans* D_2 $(a_2 < a'_2, b_2 < b'_2)$ *sont tels que* $b'_2 \geq prédécesseur(a_2)$ *et* $b_2 \leq successeur(a'_2)$*.*

Exemple 18 *Soient* $D_1 = D_2 = \{0, 1, 2, 3\}$ *avec l'ordre naturel sur chaque domaine. La contrainte* $C = (\{00, 13, 31, 32\}, (x_1, x_2))$ *n'est pas CRC puisque l'intervalle* $[0..0]$ *des supports de 0 pour* x_1 *dans* D_2 *ne « touche » pas l'intervalle* $[3..3]$ *des supports de 1 pour* x_1*. En revanche, si les domaines sont réordonnés de sorte que l'on ait* $0 <_1 3 <_1 1 <_1 2$ *et* $0 <_2 1 <_2 2 <_2 3$*, alors la contrainte est CRC.*

5.3 BC(Z) vs RC

Dans ce paragraphe, nous faisons le lien entre l'équivalence de BC(Z) et RC relativement à une contrainte C et le fait que C soit close par médiane. Cette propriété généralise la notion de contrainte CRC de Deville *et al.* [5], en ce sens que les contraintes med-*closes* binaires sont exactement les contraintes CRC [8, exemple 4.7].

[3] Si l'on n'impose pas cette condition sur les transposées, mais seulement sur les contraintes sur (x_i, x_j) pour $i < j$ selon un certain ordre, le réseau est dit convexe par rangée *directionnel* [13].

Définition 19 (med**-close [8])** *Soit* $\Sigma = (X, D, <)$ *une signature ordonnée. Une contrainte* n*-aire* $C = (R, X)$ *est* med-close pour Σ *si pour tout triplet de tuples* t, t', t'' *dans* R*, le tuple* t_m *défini par* $\forall x \in X, t_m[x] = \text{med}(t[x], t'[x], t''[x])$ *est dans* R*. Un réseau de contraintes est dit* med*-clos pour* Σ *si toutes ses contraintes sont* med*-closes pour* Σ*.*

Exemple 20 *Soient* $D_1 = D_2 = D_3 = \{0, 1, 2\}$ *avec l'ordre naturel sur chaque domaine. La contrainte* $C = (\{000, 010, 101, 222\}, (x_1, x_2, x_3))$ *n'est pas* med*-close, puisque le tuple médian de* $010, 101, 222$ *est le tuple interdit* 111*. Notons que si* D_2 *est réordonné de sorte que l'on ait* $1 <_2 0 <_2 2$ *(*D_1 *et* D_3 *inchangés), alors le tuple médian de tout triplet de tuples autorisés est également autorisé (en l'occurrence, c'est toujours* 000 *ou* 101*). Donc* C *est med-close pour ces ordres totaux.*

Le lemme suivant est donné dans le cas n-aire, tandis que nous n'avons pu prouver le lemme 22 (essentiellement, la réciproque du lemme 21) que dans le cas binaire.

Notons que l'hypothèse que toute valeur de D a un support pour C ne revient pas à dire que C est GAC. En effet, cette hypothèse se refère aux domaines initiaux D sur lesquels C est med-close, et non aux domaines courants D'.

Lemme 21 *Soit* $\Sigma = (X, D, <)$ *une signature ordonnée, et soit* $C = (R, X)$ *une contrainte* n*-aire* med*-close pour* Σ*. Supposons en outre que toute valeur de* D *a un support dans* D *pour* (R, X)*. Alors pour tout* $D' \subseteq D$*, si* C *est bound(Z)-consistante, alors elle est également range-consistante.*

Preuve Soient $a, b, c \in D'_x$ avec $a < b < c$, et notons t_a et t_c, respectivement, des supports aux bornes de a et c dans D'. On a donc $\forall y \neq x, t_a[y], t_c[y] \in [\min D'_y .. \max D'_y]$. Soit t_b un support de b dans D (qui existe par hypothèse), et soit t_m le tuple médian de t_a, t_b, t_c. On a alors $t_m[x] = b$ par hypothèse. En outre, par définition de la médiane, pour $y \neq x$ on a $t_m[y] \in [t_a[y]..t_c[y]]$. On a donc $t_m[y] \in [\min D'_y .. \max D'_y]$. t_a, t_b et t_c étant des tuples acceptés par C et C étant med-close sur D, il s'ensuit que t_m est accepté par C. C'est donc un support aux bornes de b dans D'. □

Lemme 22 *Soit* $\Sigma = ((x_1, x_2), D, <)$ *une signature ordonnée binaire, et soit* $C = (R, (x_1, x_2))$ *une contrainte binaire. Supposons que toute valeur de* D *a un support dans* D *pour* C*. Alors si* C *n'est pas* med*-close pour* Σ*, il existe un domaine* $D' \subseteq D$ *qui est bound(Z) consistant mais pas range consistant pour* C*.*

Preuve Soient sans perte de généralité $a_1, b_1, c_1 \in D_1$ avec $a_1 \leq b_1 \leq c_1$ et $a_2, b_2, c_2 \in D_2$ tels que

$a_1a_2, b_1b_2, c_1c_2 \in R$ et $t_m = \text{med}(a_1a_2, b_1b_2, c_1c_2) \notin R$. En particulier, on a $t_m \neq b_1b_2$, c'est-à-dire $b_2 \neq t_m[2] = \text{med}(a_2, b_2, c_2)$ puisque $t_m[1] = b_1$. Nous supposons sans perte de généralité $b_2 < a_2 < c_2$ (les autres cas se montrent de même).

Supposons tout d'abord que b_1 n'a pas de support dans $(D_1, [a_2..c_2])$, et considérons le domaine $D' = ([a_1..c_1], \{a_2, c_2\})$. Alors a_1, a_2, c_1, c_2 ont un support aux bornes dans D', et donc D' est bound(Z) consistant, mais b_1 n'a pas de support aux bornes dans D'.

Supposons maintenant que b_1 a un support $b_1b'_2$ dans $(D_1, [a_2..c_2])$. Soit $D' = (\{b_1\}, [b_2..b'_2])$. Par construction, on a $b_2 < a_2 \leq b'_2 \leq c_2$ et donc $b_1a_2 = t_m \notin R$. Donc b_1, b_2, b'_2 ont un support aux bornes dans D', et donc D' est bound(Z) consistant, mais a_2 n'a pas de support aux bornes dans D'. □

En utilisant les lemmes 21 et 22 et le fait que les contraintes binaires CRC et med-closes sont les mêmes, on obtient l'équivalence suivante.

Théorème 23 (BC(Z) vs RC) *Soit $\Sigma = (X, D, <)$ une signature ordonnée binaire, et $C = (R, X)$ une contrainte binaire. Supposons que toute valeur de D a un support dans D pour (R, X). Alors la bound(Z)-propagation et la range-propagation sont équivalentes sur (X, D) pour C si et seulement si C est CRC pour Σ.*

5.4 RC vs AC

Nous montrons maintenant que, pour une signature ordonnée $\Sigma = (X, D, <)$ fixée, la range consistance et l'arc consistance sont équivalentes si et seulement si le *complémentaire* C^c de C est convexe par rangée pour Σ. Nous donnons tout d'abord la définition de la convexité par rangée pour les contraintes n-aires.

Définition 24 (convexe par rangée n-aire [13]) *Soit $(X, D, <)$ une signature ordonnée. Une contrainte n-aire $C = (R, X)$ est dite* convexe par rangée pour Σ *si pour tout sous-ensemble Y de X avec $|Y| = |X| - 2$ et toute instanciation t_Y de Y, la contrainte binaire $(R[t_Y]_{X \setminus Y}, X \setminus Y)$ et sa transposée sont convexes par rangée pour Σ* [4].

Encore une fois, nous pouvons montrer une direction dans le cas n-aire, et l'autre dans le cas binaire seulement.

Lemme 25 *Soit $(X, D, <)$ une signature ordonnée, et soit $C = (R, X)$ une contrainte n-aire. Supposons que toute valeur de D a un support dans D pour (R, X). Si C^c est convexe par rangée, alors la range-propagation et la GAC-propagation sont équivalentes sur (X, D) pour C.*

Preuve Supposons que C^c est convexe par rangée. Par définition des consistances, si $D' \subseteq D$ est GAC pour Σ, alors il est range consistant pour Σ. Réciproquement, supposons que D' est range consistant. Soit $x_1 \in X$, $b_1 \in D'_1$, et soit t_b un support aux bornes de b_1 dans D'. Pour tout $x_i \in X$, on note $a_i = \min D_i, b_i = t_b[i], c_i = \max D_i$. On a $t_b \in R$, donc $t_b \notin R^c$. Soit $x_2 \neq x_1$. Puisque C^c est convexe par rangée pour Σ, $C^c[b_3 \dots b_n]_{\{1,2\}}$ est convexe par rangée. Il s'ensuit qu'on n'a pas $b_1a_2 \in R^c[b_3 \dots b_n]_{\{1,2\}}$ et $b_1c_2 \in R^c[b_3 \dots b_n]_{\{1,2\}}$, car sinon a_2, c_2 seraient des supports de b_1 dans $R^c[b_3 \dots b_n]_{\{1,2\}}$ alors que b_2 n'en est pas un (et $a_2 < b_2 < c_2$ par définition de la médiane). Donc l'un des deux tuples $b_1a_2b_3 \dots b_n$ ou $b_1c_2b_3 \dots b_n$ est dans R. En itérant le raisonnement à partir de ce nouveau tuple, on obtient ainsi qu'il existe un tuple $b_1d_2 \dots d_n \in R$ avec pour $i = 1, \dots, n$, $d_i = a_i$ ou $d_i = c_i$. Donc b_1 a un support dans D'. Finalement, D' est GAC. □

Théorème 26 (RC vs AC) *Soit $\Sigma = (X, D, <)$ une signature ordonnée binaire, et $C = (R, X)$ une contrainte binaire. Supposons que toute valeur de D a un support dans D pour C. Alors la range-propagation et l'arc-propagation sont équivalentes sur (X, D) pour C si et seulement si C^c est convexe par rangée pour Σ.*

Preuve Si C^c est convexe par rangée, le lemme 25 (cas binaire) montre que la range-propagation et l'arc-propagation sont équivalents. Réciproquement, supposons que C^c (binaire) n'est pas convexe par rangée, et soient sans perte de généralité $b_1 \in D_1, a_2, b_2, c_2 \in D_2$ avec $a_2 < b_2 < c_2$ et tels que $b_1a_2, b_1c_2 \in R^c$ et $b_1b_2 \notin R^c$. On a donc $b_1a_2, b_1c_2 \notin R$ et $b_1b_2 \in R$. Soient a_1 et c_1 des supports de a_2 et c_2 dans D, respectivement. Soit $D' = (\{a_1, b_1, c_1\}, \{a_2, c_2\})$. b_1 a un support aux bornes mais pas de support dans D', et donc D'_1 est range consistant mais pas arc consistant dans D'. □

6 Reconnaissance des contraintes CRC et de complémentaire convexe par rangée

Des résultats précédents, on obtient le corollaire suivant. L'équivalence vient du fait que l'arc consistance est plus forte que la range consistance, elle-même plus forte que la bound(Z) consistance.

Corollaire 27 (BC(Z) vs AC) *Soit $\Sigma = (X, D, <)$ une signature ordonnée binaire, et soit $C = (R, X)$ une contrainte binaire. Supposons en outre que toute*

[4] Il n'est pas clair dans l'article original si la condition « et sa transposée » est imposée. Nous l'imposons ici (c'est notamment nécessaire dans la preuve du lemme 25).

valeur de D a un support dans D pour (R, X). Alors la bound(Z)-propagation et l'arc-propagation sont équivalentes sur (X, D) pour C si et seulement si C est CRC et son complémentaire C^c est convexe par rangée pour Σ.

De fait, pour de telles contraintes, tester si toute valeur de tout domaine a un support dans le domaine courant revient à tester si les bornes de chaque domaine ont un support dans l'enveloppe convexe de l'autre domaine.

D'un point de vue de leur représentation sous forme matricielle, les contraintes CRC de complémentaire convexe par rangées sont celles pour lesquelles les 0 sont concentrés dans deux coins opposés et n'occupant aucune ligne ni colonne en commun. Les frontières 1/0 sont quelconques du moment que les 0 restent concentrés dans les coins de la matrice. Graphiquement (avec les minimums des domaines en bas et à gauche, chaque bloc étant présent ou non) :

$$\begin{pmatrix} 1 & 1 & \mathbf{1} & 0 & 0 \\ 1 & 1 & \mathbf{1} & 1 & 0 \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ 0 & 1 & \mathbf{1} & 1 & 1 \\ 0 & 0 & \mathbf{1} & 1 & 1 \end{pmatrix}$$

Cette vision nous permet de donner un algorithme polynomial pour reconnaître les contraintes pouvant être rendues CRC et de complémentaire convexe par rangée en réordonnant les domaines.

Nous prouvons tout d'abord que ces contraintes sont exactement celles pouvant être représentées comme ci-dessus. Pour des raisons de simplicité, et par symétrie, nous supposons dorénavant que le tuple $(\min D_1, \min D_2)$ ne satisfait pas R (autrement dit, par symétrie nous fixons un 0 en bas à gauche de la matrice). Intuitivement, sur le schéma ci-dessus D_1^- est le bloc de lignes de gauche, $D_1^=$ le bloc de lignes en gras, etc.

Définition 28 (CRC+^{c}RC-partition) *Soit $\Sigma = ((x_1, x_2), D, <)$ une signature ordonnée binaire, et soit $C = (R, X)$ une contrainte binaire sur Σ. Une* CRC+^{c}RC-partition *de D pour C (s'il en existe une) est une séquence de domaines $(D_1^-, D_1^=, D_1^+, D_2^-, D_2^=, D_2^+)$ telle que :*

1. *$D_1^= = \{a_1 \in D_1 \mid \forall a_2 \in D_2, a_1a_2 \in R\}$ et $D_2^= = \{a_2 \in D_2 \mid \forall a_1 \in D_1, a_1a_2 \in R\}$,*
2. *$\{D_i^-, D_i^=, D_i^+\}$ forme une partition de D_i,*
3. *$\forall i$, $\forall a_i \in D_i^-, b_i \in D_i^=, a_i < b_i$ et $\forall a_i \in D_i^-, c_i \in D_i^+, a_i < c_i$ et $\forall b_i \in D_i^=, c_i \in D_i^+, b_i < c_i$,*
4. *$R[D_1^+ \times D_2^-] = D_1^+ \times D_2^-$ et $R[D_1^- \times D_2^+] = D_1^- \times D_2^+$.*

La preuve de la proposition suivante est omise par manque de place, mais suit l'intuition géométrique.

Proposition 29 *Soit $\Sigma = (X, D, <)$ une signature ordonnée binaire, et $C = (R, X)$ une contrainte binaire sur (X, D). Supposons que toute valeur de D a un support dans D pour C. Alors $C = (R, X)$ est CRC et C^c convexe par rangée sur Σ si et seulement s'il existe une CRC+^{c}RC-partition $(D_1^-, D_1^=, D_1^+, D_2^-, D_2^=, D_2^+)$ de D telle que $R[D_1^- \times D_2^-]$ soit croissante, et $R[D_1^+ \times D_2^+]$ décroissante, pour Σ.*

Afin de trouver une partition convenable, nous construisons un graphe sur les tuples manquants dans R. La preuve du lemme 31 est omise.

Définition 30 *Soit (X, D) une signature binaire avec $X = (x_1, x_2)$, et soit C une contrainte binaire sur (X, D). Le* graphe CRC+^{c}RC *de R est défini comme le graphe non orienté (V, A) avec $V = (D_1 \times D_2) \setminus R$ et pour tous $a_1a_2, b_1b_2 \in R^c$, $\{a_1a_2, b_1b_2\} \in A$ si et seulement si $a_1 = b_1$ ou $a_2 = b_2$.*

Lemme 31 *Soit $\Sigma = (X, D, <)$ une signature ordonnée binaire, et soit $C = (R, X)$ une contrainte binaire sur (X, D). Supposons en outre que toute valeur de D a un support dans D pour C. Alors les deux points suivants sont équivalents :*

1. *il existe une CRC+^{c}RC-partition $(D_1^-, D_1^=, D_1^+, D_2^-, D_2^=, D_2^+)$ de D telle que $R[D_1^- \times D_2^-]$ soit croissante, et $R[D_1^+ \times D_2^+]$ décroissante, pour $(X, D, <)$,*
2. *le graphe CRC+^{c}RC de R, G, a deux composantes connexes C^- et $C+$ avec $D_1^- = \{a_1 \mid a_1a_2 \in C^-\}$ et de même pour D_1^+, D_2^-, D_2^+, $R[D_1^- \times D_2^-]$ est croissante, et $R[D_1^+ \times D_2^+]$ décroissante, pour Σ.*

Corollaire 32 *Soit (X, D) une signature binaire, et $C = (R, X)$ une contrainte binaire sur (X, D). Supposons en outre que toute valeur de D a un support dans D pour C. Alors on peut décider s'il existe un couple d'ordres $<$ telle que C soit CRC et de complémentaire convexe par rangée sur $(X, D, <)$ en temps $O(d^4)$, où d est la taille du plus grand domaine dans D.*

Preuve L'algorithme procède en construisant d'abord le graphe CRC+^{c}RC de C, G, dont la définition est indépendante de l'ordre sur les domaines, en temps $O(d^4)$ puisque les sommets en sont des couples de valeurs. Il calcule ensuite en temps linéaire $O(d^4)$ les composantes connexes de G. S'il y en a au moins trois, alors d'après la proposition 29 et le lemme 31 il n'existe pas d'ordre convenable.

Sinon, l'algorithme calcule D_i^+ et D_i^- pour $i = 1, 2$, et cherche un couple d'ordres $<^-$ tel que $R[D_1^- \times$

$D_2^-]$ soit croissante sur $(X, D, <^-)$, et un couple d'ordres $<^+$ tel que $R[D_1^+ \times D_2^+]$ soit décroissante sur $(X, D, <^+)$. S'il n'en trouve pas, alors d'après la proposition 29 et le lemme 31 à nouveau, il n'existe pas d'ordre convenable. Sinon, tout ordre compatible avec $<^-$ sur (D_1^-, D_2^-) et $<^+$ sur (D_1^+, D_2^+), et plaçant les premiers comme éléments minimaux et les seconds comme éléments maximaux, est convenable. En vertu des résultats du paragraphe 3, la recherche des ordres demande un temps $O(d^3)$, ce qui conclut. □

Pour terminer, notons qu'en utilisant les mêmes idées que dans le corollaire 32, on peut déterminer efficacement si un *réseau* de contraintes peut être rendu CRC et de complémentaire convexe par rangées, autrement dit résoudre efficacement le problème $\mathcal{L}$-RENOMMABILITÉ pour ce langage. Néanmoins, il n'est pas évident que l'équivalence entre l'arc consistance généralisée et la bound(Z) consistance (généralisée) reste vraie pour des contraintes définies par des *conjonctions* de contraintes CRC et de complémentaire convexe par rangée, ce qui limite les applications de ce résultat.

7 Conclusion

Nous avons étudié la technique du réordonnancement des domaines dans le but d'améliorer la résolution des CSP. Nous avons donné un algorithme polynomial qui décide si les domaines d'un réseau de contraintes peuvent être réordonnés pour obtenir un réseau de contraintes monotones, et donc facile à résoudre et à propager. Nous avons montré que savoir s'il existe un ordre sur les domaines tel que toutes les contraintes soient min-closes est NP-difficile. Ce dernier cas clôt une question laissée ouverte dans [9]. Finalement, nous avons caractérisé la classe des contraintes binaires pour lesquelles la BC(Z) consistance est équivalente à l'arc consistance, et donné un algorithme polynomial pour les reconnaître.

Références

[1] Bengt Aspvall. Recognizing disguised NR(1) instances of the satisfiability problem. *Journal of Algorithms*, 1 :97–103, 1980.

[2] Bernhard Beckert, Reiner Hähnle, and Felip Manyà. Transformations between signed and classical clause logic. In E. Dubrova, M. Miller, and T. Hanyu, editors, *Proc. 29th IEEE International Symposium on Multiple-Valued Logic (ISMVL'99)*, pages 248–255. IEEE Computer Society Press, 1999.

[3] Christian Bessiere. Constraint propagation. In F. Rossi, P. van Beek, and T. Walsh, editors, *Handbook of Constraint Programming*, Foundations of Artificial Intelligence. Elsevier, 2006.

[4] Rina Dechter and Judea Pearl. Structure identification in relational data. *Artificial Intelligence*, 58(1–3) :237–270, 1992.

[5] Yves Deville, Olivier Barette, and Pascal Van Hentenryck. Constraint satisfaction over connected row convex constraints. *Artificial Intelligence*, 109 :243–271, 1999.

[6] Martin J. Green and David A. Cohen. Domain permutation reduction for constraint satisfaction problems. *Artificial Intelligence*, 172 :1094–1118, 2008.

[7] Jean-Jacques Hébrard. A linear algorithm for renaming a set of clauses as a Horn set. *Theoretical Computer Science*, 124 :343–350, 1994.

[8] Peter Jeavons, David Cohen, and Martin C. Cooper. Constraints, consistency and closure. *Artificial Intelligence*, 101 :251–265, 1998.

[9] Peter G. Jeavons and Martin C. Cooper. Tractable constraints on ordered domains. *Artificial Intelligence*, 79 :327–339, 1995.

[10] Ugo Montanari. Networks of constraints : Fundamental properties and applications to picture processing. *Information Science*, 7 :95–132, 1974.

[11] Thomas J. Schaefer. The complexity of satisfiability problems. In *Proc. 10th Annual ACM Symposium on Theory Of Computing (STOC'78)*, pages 216–226. ACM Press, 1978.

[12] Christian Schulte and Peter J. Stuckey. When do bounds and domain propagation lead to the same search space. In H. Søndergaard, editor, *Proc. 3rd International Conference on Principles and Practice of Declarative Programming (PPDP'01)*, pages 115–126. ACM Press, 2001.

[13] Peter van Beek and Rina Dechter. On the minimality and global consistency of row-convex constraint networks. *Journal of the ACM*, 42 :543–561, 1995.

[14] Pascal Van Hentenryck, Yves Deville, and Choh-Man Teng. A generic arc-consistency algorithm and its specializations. *Artificial Intelligence*, 57 :291–321, 1992.

[15] Gerhard J. Woeginger. An efficient algorithm for a class of constraint satisfaction problems. *Operations Research Letters*, 30 :9–16, 2002.

[16] Bruno Zanuttini and Jean-Jacques Hébrard. A unified framework for structure identification. *Information Processing Letters*, 81(6) :335–339, 2002.