Relaxation de AllDifferent avec préférences

Jean-Philippe Metivier, Patrice Boizumault, Samir Loudni

Relaxation de AllDifferent avec préférences

Jean-Philippe Métivier **Patrice Boizumault** **Samir Loudni**

GREYC (UMR 6072) — Université de Caen Basse Normandie
Boulevard du Maréchal Juin – BP 5186
14032 Caen Cedex
{jmetivie,patrice.boizumault}@info.unicaen.fr loudni@iutc3.unicaen.fr

Résumé

Dans cet article, nous proposons deux nouvelles sémantiques de violation pour la contrainte globale `AllDifferent` permettant de prendre en compte les préférences formulées par l'utilisateur. La première, dite *sémantique basée variable*, pour laquelle un poids est attribué à chaque variable et où la quantité de violation correspond à la somme des poids des variables à réinstancier. Pour cette sémantique, nous proposons un algorithme maintenant la consistance globale en temps polynomial. La seconde, dite *sémantique basée décomposition*, pour laquelle un poids est attribué à chaque contrainte binaire de différence et où la quantité de violation correspond à la somme des poids des contraintes binaires de différence insatisfaites. Pour cette sémantique, nous montrons que le test de consistance est un problème NP-Complet. Nous proposons alors de maintenir une consistance locale basée sur le calcul d'un minorant (en temps polynomial) et nous étudions expérimentalement la qualité du minorant et de la consistance ainsi maintenue.

1 Introduction

Beaucoup de `CSPs` modélisant des problèmes réels sont sur-contraints (il n'existe pas de solution satisfaisant toutes les contraintes). Dans cette situation, il est naturel de permettre à certaines contraintes, les contraintes molles, d'être relaxées. Plusieurs cadres ont été proposés pour traiter les problèmes sur-contraints (voir [5, 2]).

Plus récemment, des versions relaxées de certaines contraintes globales (`AllDifferent` [15, 19], `Gcc` [16, 20], `Regular` [10, 20],...) ont été proposées. L'idée sous-jacente de ces relaxations est de définir une sémantique de violation attachée à la contrainte de façon à obtenir un algorithme de filtrage exploitant à la fois la structure de la contrainte et la nature de la violation qui lui est attachée. Dans ces travaux, différentes sémantiques de violation mesurant le degré d'insatisfaction de chacune de ces contraintes globales ont été définies.

Dans cet article, nous proposons deux nouvelles sémantiques de violation pour la contrainte globale `AllDifferent` permettant de prendre en compte les préférences formulées par l'utilisateur : La première, dite *sémantique basée variable*, pour laquelle un poids est attribué à chaque variable et où la quantité de violation correspond à la somme des poids des variables à réinstancier. Pour cette sémantique, nous proposons un algorithme maintenant la consistance globale en temps polynomial. La seconde, dite *sémantique basée décomposition*, pour laquelle un poids est attribué à chaque contrainte binaire de différence et où la quantité de violation correspond à la somme des poids des contraintes binaires de différence insatisfaites. Pour cette sémantique, nous montrons que le test de consistance est un problème NP-Complet. Nous proposons alors de maintenir une consistance locale basée sur le calcul d'un minorant en temps polynomial.

Nous rappelons dans un premier temps comment la contrainte `AllDifferent` peut-être modélisée et résolue de deux façons différentes : couplages dans un graphe biparti et algorithmes de flots dans un réseau. La section 3 présente notre contribution concernant la sémantique basée variable. Dans la section 4, nous traitons la sémantique basée décomposition. Nous montrons que le test de consistance est un problème NP-Complet. En conséquence, nous proposons de maintenir une consistance locale grâce à un calcul de minorant utilisant la notion de *Conflict-Sets*. À la fin de cette section, nous étudions expérimentalement la qualité du minorant et de la consistance ainsi maintenue.

2 La contrainte globale AllDifferent

Dans cette section, nous rappelons la définition de la contrainte globale `AllDifferent` et nous présentons deux modèles permettant de mettre en œuvre le test de consistance globale et le filtrage associé :
- Modèle graphes bipartis et couplages [15],
- Modèle de flots [16].

2.1 Définition et exemple

La contrainte globale `AllDifferent` impose que les valeurs prises par un ensemble de n variables soient deux à deux différentes [15].

Définition 1 (`AllDifferent`). *La contrainte globale* `AllDifferent`$(X_1, \ldots, X_n)$ *est consistante ssi* $X_i \neq X_j$ *pour* $1 \leq i < j \leq n$.

***Exemple* 1.** *Considérons la contrainte* `AllDifferent`(X_1, X_2, X_3)*, avec* $D_1 = D_2 = \{1, 2\}$ *et* $D_3 = \{1, 2, 3\}$*. Cette contrainte admet les deux solutions suivantes :* $(X_1 = 1, X_2 = 2, X_3 = 3)$ *et* $(X_1 = 2, X_2 = 1, X_3 = 3)$.

2.2 Graphes bipartis et couplages

2.2.1 Modèle

Jean-Charles Régin a montré que la contrainte `AllDifferent`$(X_1, \ldots, X_n)$ peut être modélisée grâce aux couplages [15]. La relation entre un couplage et une contrainte `AllDifferent` se fait au travers d'un graphe biparti $\mathcal{G}=\{(X, D_X, A)\}$, avec :

$$\begin{array}{rcl} X & = & \{X_1, \ldots, X_n\} \\ D_X & = & \cup_X D_i \\ A & = & \{(X_i, d) \mid d \in D_i \text{ et } X_i \in X\}. \end{array}$$

Un couplage $\mathcal{M}$ est dit *couvrant* de X dans le graphe biparti $\mathcal{G}$ si tout sommet de X est l'extrémité d'une arête de $\mathcal{M}$. Un couplage couvrant dans $\mathcal{G}$ est un couplage *maximal* de cardinalité égale à $|X|$.

La Figure 1 illustre le graphe biparti associé à la contrainte `AllDifferent` de l'Exemple 1.

2.2.2 Test de Consistance

Grâce à l'utilisation de couplages, on peut caractériser la consistance d'une contrainte `AllDifferent`.

Théorème 1 (consistance de `AllDifferent` [15]). *La contrainte* `AllDifferent` *est consistante ssi il existe un couplage couvrant* $\mathcal{M}$ *de* X *dans* $\mathcal{G}$*. De plus, l'ensemble des solutions coïncide avec l'ensemble des couplages couvrants.*

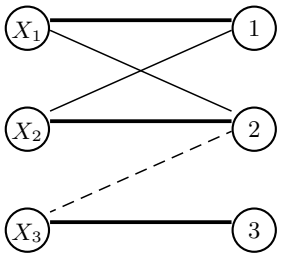

Fig. 1 – Graphe biparti associé à `AllDifferent`(X_1, X_2, X_3) avec D_1=D_2=$\{1,2\}$ et D_3=$\{2,3\}$. Les arêtes en gras représentent un couplage couvrant associé à la solution (1,2,3). En pointillés, les arêtes qui n'appartiennent à aucun couplage couvrant.

2.2.3 Filtrage

Une valeur d d'une variable X_i est viable si et seulement si elle figure dans au moins un couplage couvrant.

Grâce à une propriété énoncée par Claude Berge [1], une arête participe à au moins un couplage couvrant si et seulement si pour un couplage couvrant $\mathcal{M}$ donné, cette arête appartient soit à un cycle alterné pair (i.e., une suite d'arêtes telles qu'une appartient à $\mathcal{M}$ et la suivante non), soit à une chaîne alternée paire qui commence à un sommet libre (i.e., non couvert par $\mathcal{M}$). On a alors la propriété suivante :

Propriété 1 (viabilité d'une valeur pour `AllDifferent`). *Soit* $\mathcal{G}$ *le graphe biparti associé à la contrainte* `AllDifferent`$(X_1, \ldots, X_n)$ *et* $\mathcal{M}$ *un couplage couvrant de* $\mathcal{G}$*. Une valeur* d *du domaine de* X_i *est viable ssi elle appartient à* $\mathcal{M}$*, à un cycle alterné pair ou à une chaine alternée paire partant d'un sommet libre.*

Sur l'exemple de la Figure 1, l'arête $(X_3, 2)$ n'appartient à aucun couplage couvrant ; (X_3,2) est une valeur non viable.

2.2.4 Complexités

La complexité du test de consistance est en $\mathcal{O}(\sqrt{n}m)$ [6], avec n le nombre de variables et m la somme des cardinaux des domaines de X.

La détection de toutes les valeurs viables peut être réalisée par calcul des composantes fortement connexes et des chaînes alternées. Ainsi, le filtrage peut être effectué en $\mathcal{O}(m + n)$ [17].

2.3 Modèle de flots

2.3.1 Réseau associé

La seconde approche considère `AllDifferent` comme un cas particulier de la contrainte `Global Cardinality Constraint` (`gcc`) [16].

La contrainte `AllDifferent` peut être modélisée et résolue sous forme d'un problème *de flot* maximal dans un réseau $\mathcal{R} = \{\mathcal{V}, \mathcal{A}\}$. L'ensemble des sommets $\mathcal{V} = \mathcal{X} \cup D_X \cup \{s, t\}$ où s et t sont les sommets source et puits. L'ensemble des arcs $\mathcal{A} = \mathcal{A}_s \cup \mathcal{A}_X \cup \mathcal{A}_t$ où :

- $\mathcal{A}_s = \{(s, X_i) \mid X_i \in X\}$,
- $\mathcal{A}_X = \{(X_i, d) \mid X_i \in X, d \in D_i\}$,
- $\mathcal{A}_t = \{(d, t) \mid d \in D_X\}$.

À chaque arc $a \in \mathcal{A}$ est associée une *capacité* $c(a) = 1$ et une *demande* $d(a)$ définie comme suit :

$$d(a) = \begin{cases} 1 & \text{si } a \in \mathcal{A}_s \\ 0 & \text{si } a \in \mathcal{A}_X \\ 0 & \text{si } a \in \mathcal{A}_t \end{cases}$$

2.3.2 Test de Consistance & Filtrage

On a l'équivalence suivante [16] : la contrainte `AllDifferent` est consistante si et seulement si il existe un (s, t)-flot de valeur n dans le réseau associé (Ford & Fulkerson [4]).

Là encore, on peut caractériser simplement les valeurs viables : une valeur d d'une variable X_i est viable si et seulement si il existe un (s, t)-flot de valeur n passant par l'arc (X_i,d).

La complexité du test de consistance est en $O(n \times m)$ [4]. La détection de toutes les valeurs viables se fait par calcul de composantes fortement connexes dans le graphe résiduel. Ainsi, le filtrage associé peut être effectué en $O(m + n)$.

3 Sémantique de violation basée variable

Dans cette section, nous nous intéressons à la sémantique de violation μ_{var} [12] pour laquelle la quantité de violation d'une instanciation est le nombre de variables à réinstancier (dans celle-ci) pour que la contrainte soit vérifiée. Après avoir rappelé les travaux de Willem Jan van Hoeve sur la contrainte globale `soft-AllDifferent`, nous présenterons notre contribution, la contrainte globale Σ-`AllDifferent`, qui permet de prendre en compte des *préférences* sur les variables à réinstancier. Pour formuler les algorithmes de test de consistance et de filtrage, nous utiliserons systématiquement, dans cette section, le modèle graphes bipartis et couplages.

3.1 La contrainte soft-AllDifferent(μ_{var})

3.1.1 La sémantique de violation μ_{var}

Thierry Petit *et al.* [12] ont proposé une sémantique de violation, basée variable (μ_{var}), permettant de quantifier le degré de violation d'une contrainte globale. La mesure de violation μ_{var} est définie par :

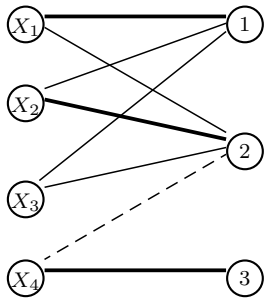

Fig. 2 – Graphe biparti associé à `soft-AllDifferent`$((X_1, \ldots, X_4),\mu_{var},z)$, avec D_1=D_2=D_3={1,2}, D_4={2,3} et D_z={0,1}. Les arêtes en gras, représentent un couplage maximal. En pointillés, la valeur non viable.

$$\mu_{var}(X_1, \ldots, X_n) = \sum_{d \in D_X} max(|\{i \mid X_i = d\}| - 1, 0)$$

Soit z une variable de coût de domaine D_z. La relaxation de la contrainte `AllDifferent` selon la sémantique de violation μ_{var} est définie comme suit :

Définition 2 (`soft-AllDifferent`(μ_{var}))**.** *La contrainte* ***soft-AllDifferent****$((X_1, \ldots, X_n),\mu_{var},z)$ est consistante ssi il existe une instanciation complète des variables de X telle que $\mu_{var}(X_1, \ldots, X_n) \leq max(D_z)$.*

Considérant l'exemple de la Figure 2. L'instanciation (1,2,1,3) oblige à réinstancier X_1 ou X_3, son coût est égal à 1. Comme $max(D_z) = 1$, l'instanciation (1,2,1,3) est une solution de la contrainte `soft-AllDifferent`$((X_1,X_2,X_3,X_4),\mu_{var},z)$. Par contre, l'instanciation (1,1,1,3) de coût 2 n'en est pas une.

3.1.2 Test de consistance

Dans [19], Willem Jan van Hoeve propose une caractérisation du test de consistance en termes de flots dans un réseau. Nous en donnons ici une caractérisation en termes de couplage dans un graphe biparti.

Théorème 2 (consistance de `soft-AllDifferent`(μ_{var}) [12])**.** *La contrainte* ***soft-AllDifferent****$((X_1, \ldots, X_n),\mu_{var},z)$ est consistante ssi il existe un couplage maximal $\mathcal{M}$ de $\mathcal{G}$ tel que $(n - |\mathcal{M}|) \leq max(D_z)$.*

Preuve. *Soit $\mathcal{M}$ un couplage maximal de $\mathcal{G}$. Le nombre de variables de X pouvant prendre des valeurs deux à deux différentes est égal à $|\mathcal{M}|$, car $\mathcal{M}$ est un couplage maximal. Le nombre de variables à réinstancier est donc de $(n - |\mathcal{M}|)$.* □

X	X_1	X_2	X_3	X_4	X_5	X_6
φ	6	10	2	7	7	4

TAB. 1 – Matrice des coûts associés à la contrainte Σ-`AllDifferent`$((X_1,\ldots,X_6),\mu^{\Sigma}_{var},z)$.

Sur l'exemple de la Figure 2, pour l'instanciation (1,2,1,3), il existe au moins un couplage maximal de cardinalité 3. Cette instanciation est donc une solution. Par contre, pour l'instanciation (1,1,1,3), tous les couplages maximaux sont de cardinalité 2. En conséquence, l'instanciation (1,1,1,3) n'est pas une solution.

3.1.3 Filtrage

Comme pour `AllDifferent`, on peut caractériser simplement les valeurs viables grâce à l'utilisation des couplages maximaux.

Corollaire 1 (viabilité d'une valeur pour `soft-AllDifferent`(μ_{var}))**.** *Une valeur d du domaine de X_i est viable ssi il existe au moins un couplage maximal $\mathcal{M}$ de $\mathcal{G}$ contenant l'arête (X_i,d) tel que $(n - |\mathcal{M}|) \leq max(D_z)$.*

Sur l'exemple de la Figure 2, on constate que l'arête $(X_4,2)$ ne peut apparaître que dans des couplages maximaux de cardinalité 2, donc de coût de 2. Comme $max(D_z) = 1$, $(X_4,2)$ est une valeur non viable.

3.2 Contributions : Σ-AllDifferent (μ^{Σ}_{var})

3.2.1 La sémantique de violation μ^{Σ}_{var}

Pour exprimer des préférences sur le choix des variables à réinstancier, on associe à chaque variable X_i un poids φ_i reflétant son importance par rapport aux autres. Ainsi, la quantité de violation d'une instanciation est égale à la somme des poids des variables à réinstancier. La mesure de violation μ^{Σ}_{var} est définie par :

$$\mu^{\Sigma}_{var}(X_1,\ldots,X_n) = \sum_{d\in D_X} \Big(\sum_{X_i=d, X_i\in X} \varphi_i - max_{X_i=d, X_i\in X}(\varphi_i) \Big)$$

Soit z une variable de coût de domaine D_z. La relaxation de la contrainte `AllDifferent` selon la sémantique de violation μ^{Σ}_{var} est définie comme suit :

Définition 3 (Σ-`AllDifferent`(μ^{Σ}_{var}))**.** *La contrainte Σ-`AllDifferent`$((X_1,\ldots,X_n), \mu^{\Sigma}_{var}, z)$ est consistante ssi il existe une instanciation complète des variables de X telle que $\mu^{\Sigma}_{var}(X_1,\ldots,X_n) \leq max(D_z)$.*

***Exemple* 2.** *Une usine souhaite affecter des équipes de nettoyage pour l'entretien de ses machines. À chaque ronde, une équipe ne peut nettoyer parfaitement qu'une seule machine mais peut, en plus de la première, nettoyer sommairement d'autres machines. Chaque machine ne peut être nettoyée que par certaines équipes car il faut que celles-ci soient qualifiées pour l'entretien de ce type de machine (démontage, remontage, ...). À chaque machine est associée une priorité de nettoyage définie par la nature des produits qu'elle manipule. Par exemple, une machine manipulant des produits alimentaires doit être nettoyée de manière prioritaire par rapport à une cartonneuse. Le but est de trouver une affectation pour chaque machine d'une équipe de façon à ce que la somme des priorités des machines sommairement nettoyées ne dépasse pas un seuil maximal donné.*

Ce problème peut être modélisé grâce à une contrainte Σ-`AllDifferent`(μ^{Σ}_{var}). On aura comme ensemble de variables les machines qui auront pour domaine la liste des équipes qualifiées pour leur nettoyage et pour poids leur priorité vis à vis du nettoyage.

Le tableau 1 illustre la matrice de coûts associés à une ronde où six machines doivent être nettoyées. L'équipe 1 *(resp.* 2 *et* 3*) est qualifiée pour le nettoyage des machines* 1*,* 2*,* 3 *et* 4 *(resp.* 1*,* 4 *et* 5 *pour l'équipe* 2 *et* 4*,* 5 *et* 6 *pour l'équipe* 3*).*

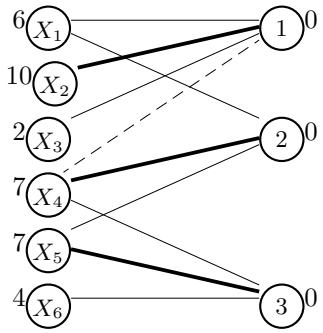

FIG. 3 – Graphe biparti associé à Σ-`AllDifferent` $((X_1,\ldots,X_6),\mu^{\Sigma}_{var},z)$, avec D_z=[0 ... 12]. En gras, les arêtes appartenant au couplage de poids maximal. En plein, les autes valeurs viables, et en pointillés les non viables.

Pour l'instanciation (1,1,1,2,2,3) de l'Exemple 2, $\mu^{\Sigma}_{var}(1,1,1,2,2,3)$=15.

3.2.2 Test de consistance

Comme pour `AllDifferent`, la contrainte Σ-`AllDifferent`(μ^{Σ}_{var}) est modélisée sous forme d'un graphe biparti $\mathcal{G}_{var}$=$\{(X, D_X, A)\}$, avec des poids sur les sommets, définis comme suit :

$$\begin{array}{rcl} w(d) & = & 0 \text{ si } d \in D_X \\ w(X_i) & = & \varphi_i \text{ si } X_i \in X \end{array}$$

La Figure 3 illustre le graphe biparti associé à la contrainte de l'Exemple 2.

Le test de consistance peut être mis en œuvre par la recherche d'un couplage de poids maximal $w(\mathcal{M})$ dans $\mathcal{G}_{var}$. Le poids d'un couplage $\mathcal{M}$ est défini comme suit :

$$w(\mathcal{M}) = \sum_{X_i \in \mathcal{M}} \varphi_i$$

Les variables participant à un couplage de poids maximal dans $\mathcal{G}_{var}$ correspondent aussi aux variables couvertes par un couplage maximal. Ce sont donc les variables hors du couplage maximal qui devront être réinstanciées. De telles variables peuvent être instanciées à n'importe quelle valeur de leur domaine, car elles entreront toujours en conflit avec une variable couverte par $\mathcal{M}$.

Théorème 3 (consistance de Σ-`AllDifferent`(μ^{Σ}_{var}))**.** *La contrainte* Σ-*`AllDifferent`*$((X_1, \ldots, X_n), \mu^{\Sigma}_{var}, z)$ *est consistante ssi il existe un couplage de poids maximal $w(\mathcal{M})$ de $\mathcal{G}_{var}$ tel que $\sum_{X_i \in X} \varphi_i - w(\mathcal{M}) \leq max(D_z)$.*

Preuve. *Soit $\mathcal{M}$ un couplage de poids maximal dans $\mathcal{G}_{var}$; $\mathcal{M}$ étant un couplage maximal, les seules variables à ré-instancier appartiennent à $X \backslash \mathcal{M}$, et leur poids est égal à :*

$$\sum_{X_i \in X \backslash \mathcal{M}} \varphi_i = \sum_{X_i \in X} \varphi_i - \sum_{X_i \in \mathcal{M}} \varphi_i = \sum_{X_i \in X} \varphi_i - w(\mathcal{M})$$

□

Sur l'exemple de la Figure 3, il existe au moins un couplage $\mathcal{M}$ de poids maximal, avec $\mathcal{M} = \{(X_2, 1), (X_4, 2), (X_5, 3)\}$. La somme totale des poids est égale à 36, et $w(\mathcal{M}) = 24$. L'instanciation (1,1,1,2,3,3) de coût 12 est une solution pour la contrainte Σ-`AllDifferent`$((X_1,\ldots,X_6), \mu^{\Sigma}_{var}, z)$ car $max(D_z) = 12$. Par contre, l'instanciation (2,1,1,3,3,3) de coût 13 n'en est pas une.

Le calcul d'un couplage de poids maximal est en $\mathcal{O}(n \times m)$ [3].

3.2.3 Filtrage

Corollaire 2 (viabilité d'une valeur pour Σ-`AllDifferent`(μ^{Σ}_{var}))**.** *Soit $\mathcal{M}\star$ l'ensemble de tous les couplages de poids maximal tels que $\sum_{X_i \in X} \varphi_i - w(\mathcal{M}) \leq max(D_z)$. La valeur (X_i, d) est viable ssi une des deux conditions suivantes est vérifiée :*

i) il existe un couplage $\mathcal{M} \in \mathcal{M}\star$ tel que $(X_i, d) \in \mathcal{M}$,

ii) $\varphi_i \leq \min_{\mathcal{M} \in \mathcal{M}\star}(\min_{X_j \in \mathcal{M}} \varphi_j)$

Algorithm 1: Filtrage pour Σ-`AllDifferent`(μ^{Σ}_{var})

début
- Soit $\mathcal{M}$ un couplage de poids maximal de $\mathcal{G}_{var}$ tel que $\sum_{X_i \in X} \varphi_i - w(\mathcal{M}) \leq max(D_z)$;
- Soit $CFCs$ l'ensemble des composantes fortement connexes de $\mathcal{G}_{var}$;
- Soit CAs l'ensemble des chaînes augmentantes partant d'un sommet libre de $\mathcal{G}_{var}$;
- Soit $Aretes = \mathcal{M} \cup CFCs \cup CAs$;
- Soit $Pmin = min_{(X_j, \_) \in Aretes} \varphi_j$;
- **pour chaque** (X_i, d) **faire**
  - **si** $((X_i, d) \notin Aretes)$ `et` $(\varphi_i > Pmin)$ **alors**
    - $D_i \leftarrow D_i \backslash \{d\}$

fin

Sur l'exemple de la Figure 3, aucun couplage de poids maximal de $\mathcal{M}\star$ ne contient l'arête $(X_4, 1)$. Comme le poids de la plus petite variable couverte parmi tous les couplages de $\mathcal{M}\star$ est égal à 6 et que $\varphi_4 = 7$, la valeur $(X_4, 1)$ est donc non viable.

La détection de toutes les valeurs viables peut être réalisée par calcul des composantes fortement connexes en $\mathcal{O}(n+m)$ et des chaînes alternées partant d'un sommet libre de plus petit poids en $\mathcal{O}(n+m)$.

4 Sémantique de violation basée décomposition

Dans cette section, nous nous intéressons à la sémantique de violation μ_{dec} [12] pour laquelle la quantité de violation d'une instanciation correspond au nombre de contraintes binaires insatisfaites. Après avoir rappelé les travaux de Willem Jan van Hoeve sur la contrainte globale `soft-AllDifferent`, nous présenterons notre contribution, la contrainte globale Σ-`AllDifferent`, qui permet de prendre en compte des *préférences* sur les contraintes binaires à relaxer.

À la différence de la sémantique précédente, nous montrons que le test de consistance pour Σ-`AllDifferent`(μ^{Σ}_{dec}) est un problème NP-Complet. En conséquence, nous proposons de maintenir une consistance locale grâce à un calcul de minorant utilisant la notion de *Conflict-Sets*. À la fin de cette section, nous étudions expérimentalement la qualité du minorant et de la consistance ainsi maintenue.

4.1 La contrainte soft-AllDifferent (μ_{dec})

4.1.1 La sémantique de violation μ_{dec}

Cette sémantique, issue des travaux de Thierry Petit *et al.* [12], permet, pour une contrainte globale pouvant être décomposée en un réseau de

contraintes élémentaires, de quantifier le degré de violation d'une instanciation par le nombre de contraintes élémentaires de différence insatisfaites. Dans le cas d'`AllDifferent`, la mesure de violation μ_{dec} est définie par :

$$\mu_{dec}(X_1,\ldots,X_n) = |\{(X_i,X_j) \mid 1 \leq i < j \leq n \text{ et } X_i = X_j\}|$$

Soit z une variable de coût de domaine D_z. La relaxation de la contrainte `AllDifferent` selon la sémantique de violation μ_{dec} est définie comme suit :

Définition 4 (`soft-AllDifferent`(μ_{dec})). *La contrainte* `soft-AllDifferent`*($(X_1,\ldots,X_n)$,μ_{dec},z) est consistante ssi il existe une instanciation complète des variables de X telle que $\mu_{dec}(X_1,\ldots,X_n) \leq max(D_z)$.*

4.1.2 Test de consistance

Willem Jan van Hoeve propose, dans [19], d'étendre le réseau $\mathcal{R}$ (cf. Section 2.3) en $\mathcal{R}_{dec}$, par l'ajout d'arcs de violation $\tilde{A}_t$, pour modéliser les violations des contraintes binaires de différence.

$\tilde{A}_t = \{(d,t) \mid d \in D_i, X_i \in X\}$, avec $c(a) = 1$ pour chaque arc $a \in \tilde{A}_t$. À chacun des arcs issus d'un même sommet d est associée une étiquette $j \in \{0,\ldots,(\delta(d)-1)\}$, avec $\delta(d)$ le nombre d'arcs entrants en d. À chaque arc $a \in \tilde{A}_t$ est associé un poids $w(a)=j$.

Ainsi, le premier arc issu d'une valeur d a un poids nul car lorsque que la valeur d est affectée une première fois, aucune contrainte binaire n'est insatisfaite. Le deuxième arc a un poids de 1, car lorsque que l'on affecte une deuxième fois la valeur d alors une contrainte binaire est insatisfaite. Le troisième arc aura un poids de 2, car si on affecte d une troisième fois deux contraintes binaires supplémentaires seront alors insatisfaites, et ainsi de suite. La Figure 4 illustre le réseau associé à la contrainte `soft-AllDifferent`(μ_{dec}).

Théorème 4 (consistance de `soft-AllDifferent`(μ_{dec}) [19]). *Une contrainte* `soft-AllDifferent`*($(X_1,\ldots,X_n)$,μ_{dec},z) est consistante ssi il existe dans $\mathcal{R}_{dec}$ un (s,t)-flot de valeur n de poids inférieur ou égal à $max(D_z)$.*

4.1.3 Filtrage

Comme pour `AllDifferent` (cf. Section 2.3.2), on peut caractériser la viabilité d'une valeur (X_i,d) grâce à l'existance d'un (s,t)-flot passant par l'arc (X_i,d).

Corollaire 3 (viabilité d'une valeur pour `soft-AllDifferent`(μ_{dec}) [19]). *Une valeur (X_i,d) est viable ssi il existe un (s,t)-flot f de valeur n dans $\mathcal{R}_{dec}$ passant par l'arc (X_i,d) tel que $w(f) \leq max(D_z)$.*

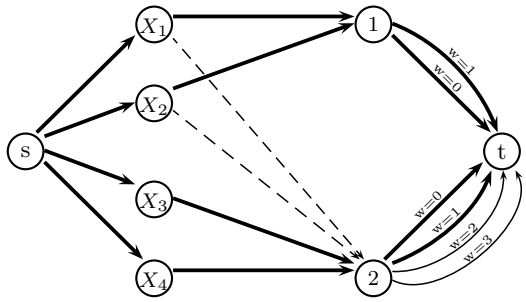

Fig. 4 – Réseau associé à la contrainte `soft-AllDifferent`(μ_{dec}). Les arcs en gras représentent un flot maximal de poids minimal associé à l'instanciation (1,1,2,2) de valuation optimale 2. Les arcs en pointillés réprésentent les valeurs non viables.

Le retrait de toutes les valeurs non viables peut être effectué en $\mathcal{O}(m+n)$, via la recherche de composantes fortement connexes et donc maintenir la consistance globale en $\mathcal{O}(n \times m)$.

4.1.4 Nombre minimal de contraintes binaires insatisfaites

On peut obtenir une instanciation de poids minimal $\mathcal{A}^\star$ grâce à la recherche d'un (s,t)-flot f de valeur n de poids minimal dans $\mathcal{R}_{dec}$, avec $\mu_{dec}(\mathcal{A}^\star) = w(f)$. On notera par $\mu^\star_{dec}$ le coût d'une telle instanciation ; $\mu^\star_{dec}$ correspond au nombre minimal de contraintes binaires de différence insatisfaites par une contrainte `soft-AllDifferent`(μ_{dec}).

4.1.5 Autres travaux

Claude-Guy Quimper *et al.* [14] et Alenxandro Zanarini *et al.* [21], proposent, pour `soft-Gcc` et donc pour `soft-AllDifferent`(μ_{dec}) (vu comme un cas particulier de `soft-Gcc`), de remplacer le test de consistance par le calcul de couplages maximaux, permettant ainsi de maintenir la consistance globale en $\mathcal{O}(\sqrt{n} \times m)$.

4.2 Contributions : Σ-AllDifferent (μ^Σ_{dec})

4.2.1 La sémantique de violation μ^Σ_{dec}

Pour exprimer des préférences sur le choix des contraintes à relaxer, on associe, à chaque contrainte binaire de différence $X_i \neq X_j$, un poids φ_{ij}, représentant l'importance du conflit entre X_i et X_j. Ainsi, la quantité de violation d'une instanciation est égale à la somme des poids des contraintes insatisfaites. La mesure de violation μ^Σ_{dec} est définie par :

$$\mu^\Sigma_{dec}(X_1,\ldots,X_n) = \sum_{\{(X_i,X_j) \mid 1 \leq i < j \leq n \text{ et } X_i = X_j\}} \varphi_{ij}$$

X	X_2	X_3	X_4	X_5
X_1	10	2	4	8
X_2		4	8	3
X_3			6	7
X_4				10

TAB. 2 – Matrice des coûts associés à la contrainte Σ-`AllDifferent`$((X_1,\ldots,X_5),\mu^{\Sigma}_{dec},z)$.

***Exemple* 3.** *Le responsable d'un complexe hôtelier souhaite partager les différentes salles de restauration entre différentes communautés. Chaque communauté dispose d'exigences concernant la liste des salles qu'elle peut utiliser pour déjeuner. Chaque communauté possède différents degrés d'affinité, par exemple la communauté des amoureux de la balade en forêt est très fortement en conflit avec la communauté des chasseurs alors que la communauté de la fanfare municipale possède des liens amicaux avec le groupe des chasseurs. Afin d'éviter des conflits entre différentes communautés, le responsable doit les répartir dans des salles en fonction de leurs affinités.*

Ce problème peut être modélisé grâce à une contrainte Σ-`AllDifferent`(μ^{Σ}_{dec}) avec comme ensemble de variables les différentes communautés, qui ont pour domaine les salles qu'elles peuvent accepter pour leur déjeuner. Le poids de chaque contrainte binaire de différence reflète alors l'importance du conflit entre deux communautés.

Le tableau 2 décrit la matrice des affinités des cinq communautés devant déjeuner dans le complexe. Le groupe 1 des chasseurs désire déjeuner dans la salle 1, le groupe 3 des amoureux de la balade en forêt dans la salle 2. Tous les autres groupes n'ont pas d'exigences.

Pour l'instanciation (1,1,1,2,2) de l'Exemple 3, $\mu^{\Sigma}_{dec}(1,1,1,2,2) = 26$.

4.2.2 NP-Complétude du test de consistance

Théorème 5. *Trouver une solution pour une contrainte Σ-`AllDifferent`(X,μ^{Σ}_{dec},z) est un problème NP-Complet.*

Preuve. *Considérons le problème de coloration des sommets d'un graphe $\mathcal{G}_{col}=(V,E)$. Si on transforme $\mathcal{G}_{col}$ en ajoutant l'ensemble des arcs $E'=\{(V_i,V_j), (V_i,V_j)\notin E\}$, avec pour tout arc $a \in E$ un poids $w(a)=(\mid E' \mid +1)$ et pour tout arc $a \in E'$ un poids $w(a)=1$ (cf. Figure 5), alors trouver une solution à Σ-`AllDifferent`(V,μ^{Σ}_{dec},z) avec $D_z=\{0,\ldots,\mid E' \mid\}$ est équivalent à trouver une solution au problème de coloration. Cette transformation s'effectue en temps polynomial car on ajoute au maximum un nombre quadratique d'arcs à $\mathcal{G}_{col}$.*

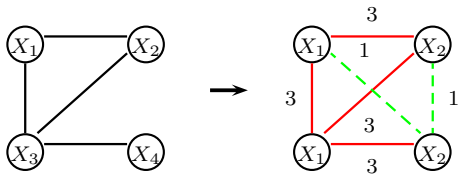

FIG. 5 – Transformation d'un problème de coloration pour une contrainte Σ-`AllDifferent`(μ^{Σ}_{dec}).

S'il existait un algorithme polynomial pour trouver une solution à Σ-`AllDifferent`(X,μ^{Σ}_{dec},z), alors grâce à la réduction décrite précédemment on pourrait trouver une solution au problème de coloration en temps polynomial, or le problème de coloration est un problème NP-Complet. □

4.2.3 Minorant et consistance locale

Nous proposons donc de maintenir une consistance partielle grâce à un calcul de minorant qui s'effectue en temps polynomial. Les principes de cet algorithme de filtrage sont présentés dans l'Algorithme 2.

Pour chaque valeur (X_i,d), on calcule un minorant du coût des solutions étendant l'affectation de d à X_i. Si le minorant trouvé excède la borne fixée (i.e., $\max D_z$), alors cette valeur est non viable. Puisque la consistance maintenue n'est que locale, il est nécessaire de propager les retraits, ce qui est assuré par la boucle `tant que` dans l'Algorithme 2.

Ainsi, on peut maintenir une consistance locale en $\mathcal{O}(m^2 \times LB)$ dans le pire cas, avec LB le coût de calcul d'un minorant.

4.2.4 Calcul d'un premier minorant

Pour calculer un premier minorant nous utilisons la détection de conflict-sets [11, 7].

Définition 5 (Conflict-set)**.** *Soit un CSP $P=(X,D,C)$, $K \subseteq C$ et X_K le sous-ensemble des variables défini sur K. K est un conflict-set ssi $P_K=(X_K,D_K,K)$ est inconsistant.*

Définition 6 (Conflict-set minimal)**.** *Soit K un conflict-set. K est minimal ssi $\nexists c \in K$ tel que $K \backslash \{c\}$ soit un conflict-set.*

Définition 7 (Conflict-sets disjoints)**.** *Soit K_1 et K_2 deux conflict-sets. K_1 et K_2 sont disjoints ssi $K_1 \cap K_2 = \emptyset$.*

Dans un conflict-set au moins une contrainte est insatisfaite. Pour obtenir un minorant, Thierry PETIT *et al.* proposent dans [11] de décomposer le réseau de contraintes en un ensemble de conflict-sets disjoints, puis de faire la somme des contraintes de plus petit poids de chacun de ces conflict-sets.

Algorithm 2: Filtrage par minorant

```
début
    retrait ← Vrai
    tant que retrait faire
        retrait ← Faux
        pour (X_i, d) t.q. d ∈ D_i, X_i ∈ X faire
            lb ← Calculer le minorant pour
                                (X_1, ..., X_{i-1}, d, ..., X_n)
            si lb > max(D_z) alors
                retrait ← Vrai
                D_i ← D_i\{d}
fin
```

Algorithm 3: Détection de conflict-sets disjoints minimaux

```
fonction trouver-un-CS(C)
CS ← ∅
tant que C ≠ ∅ et CS est consistant faire
    CS ← CS ∪ {c | c ∈ C}
    C ← C\{c}
si CS est inconsistant
alors retourner CS

fonction trouver-tous-CS(C)
ψ ← ∅
tant que C est inconsistant faire
    CS ← min-CS(trouver-un-CS(C))
    C ← C\CS
    ψ ← ψ ∪ {CS}
retourner ψ

fonction min-CS(CS)
C_util ← ∅
tant que C_util est consistant faire
    C ← C_util
    tant que C est consistant faire
        c_last ← dernière contrainte de CS
        C ← C ∪ {c_last}
    C_util ← C_util ∪ {c_last}
    CS ← C\C_util
retourner C_util
```

Théorème 6 (Minorant [11]). *Soit $\psi=\{\psi_1,\ldots,\psi_k\}$ un ensemble de conflict-sets disjoints, et $c^{min}_{\psi_i}$ la contrainte de plus petit poids du conflict-set ψ_i, alors on a*

$$lb(P) = \sum_{\psi_i \in \psi} \varphi_{c^{min}_{\psi_i}}.$$

La recherche d'un conflict-set dans le pire des cas est en $\mathcal{O}(e{\times}CC)$, avec e le nombre de contraintes et CC le temps pour réaliser un test de consistance. La minimisation d'un conflict-set est en $\mathcal{O}(e^2{\times}CC)$.

Pour Σ-`AllDifferent`(μ^{Σ}_{dec}), nous proposons de construire un premier minorant en calculant les conflict-sets disjoints minimaux sur l'ensemble des contraintes binaires de différence. Afin d'améliorer le minorant, la fonction `trouver-tous-CS` prend en entrée la liste des contraintes triées selon l'ordre décroissant de leur poids.

Nous utilisons comme test de consistance l'arc-consistance qui peut-être maintenue en $\mathcal{O}(e{\times}d)$ car une contrainte de différence est un contrainte anti-fonctionnelle [18].

4.2.5 Amélioration du minorant

Nous montrons dans cette section que le nombre de contraintes à relaxer pour une contrainte Σ-`AllDifferent`(μ^{Σ}_{dec}) (i.e., dans le cadre avec préférences) est toujours supérieur ou égal à $\mu^{\star}_{dec}$, le nombre minimal de contraintes binaires à relaxer pour `soft-Alldifferent`(μ_{dec}) (i.e., dans le cadre sans préférences).

Théorème 7. *Pour toute instanciation complète $\mathcal{A}$, soit $C_{unsat}(\mathcal{A})$ l'ensemble des contraintes binaires de différence insatisfaites par $\mathcal{A}$, on a alors :*

$$|\,C_{unsat}(\mathcal{A})\,| \geq \mu^{\star}_{dec}$$

Preuve. *Soient $\mathcal{A}$ une instanciation complète et $|C_{unsat}(\mathcal{A})|$ le nombre de contraintes binaires insatisfaites par $\mathcal{A}$. Ce nombre est indépendant de la mesure de violation. Soit $\mathcal{A}^{\star}$ une solution optimale pour μ_{dec}, alors $|\,C_{unsat}(\mathcal{A})\,| \geq \mu_{dec}(\mathcal{A}^{\star})$ d'où $|\,C_{unsat}(\mathcal{A})\,| \geq \mu^{\star}_{dec}$ (par définition de $\mu^{\star}_{dec}$).* □

On a bien une inégalité dans le théorème 7, car dans le cadre avec préférences, on va préférer relaxer trois contraintes de poids 2 plutôt que deux contraintes de poids 4.

Soit ψ une décomposition en conflict-sets et $\alpha{=}\mu^{\star}_{dec}{-}|\,\psi\,|$. Si $\alpha{>}0$ alors on peut améliorer la qualité du minorant en ajoutant les α contraintes de plus petit poids.

Théorème 8. *Soit C_{dec} l'ensemble des contraintes binaires de différence issues de la décomposition de `All-Different`. Soit lb_{ψ} un premier minorant pour une contrainte Σ-`AllDifferent`(X,μ^{Σ}_{dec},z), C^{min}_{ψ} l'ensemble des contraintes utilisées pour le calcul de lb_{ψ} et $\alpha{=}\mu^{\star}_{dec}{-}|\psi|$. Alors on peut améliorer le minorant de la manière suivante :*

$$lb = lb_{\psi} + \sum_{1 \leq i \leq \alpha} min_i\{C_{dec} \backslash C^{min}_{\psi}\}$$

Les deux propriétés suivantes décrivent des ensembles de contraintes dont on sait facilement prouver la satisfiabilité.

Propriété 2. *Une contrainte de différence est trivialement satisfaite si les domaines des variables impliquées dans la contrainte sont disjoints*

Propriété 3 (affaiblissement du Théorème de Hall)**.** *Si ω variables impliquées dans un* `AllDifferent`$(X_1, \ldots, X_n)$ *sont les seules variables à se partager un sous-ensemble de υ valeurs de telle façon à ce que la contrainte* `AllDifferent` *réduite à ces ω variables et υ valeurs est consistante, alors toutes les contraintes binaires de différence portant sur ces ω variables seront satisfaites.*

L'exemple suivant illustre la Propriété 3 : D_1={1,2,3}, D_2={1,2}, D_3={3} et D_4={3}. Les variables X_1 et X_2 peuvent être instanciées respectivement à 1 et à 2 sans engendrer de violation. Donc toutes les contraintes liées à X_1 ou X_2 seront satisfaites.

On peut améliorer la qualité du minorant en le complétant uniquement par des contraintes pouvant être insatisfaites (celles que l'on sait être satisfaites étant de poids nul).

Corollaire 4. *Soit lb_ψ un premier minorant pour une contrainte* Σ-`AllDifferent`$(X, \mu^{\Sigma}_{dec}, z)$*, C^{min}_ψ l'ensemble des contraintes utilisées pour le calcul de lb_ψ et C_{sat} l'ensemble des contraintes répondant aux propriétés 2 ou 3. Alors on peut améliorer le minorant de la manière suivante :*

$$lb = lb_\psi + \sum_{1 \leq i \leq \alpha} min_i\{C_{dec} \backslash (C^{min}_\psi \cup C_{sat})\}$$

4.2.6 Expérimentations

Pour tester la qualité de notre minorant et la force de la consistance maintenue, nous avons réalisé des expérimentations sur des instances générées aléatoirement avec un nombre de variables n et un nombre de valeurs dans D_X fixés. Nous avons mesuré deux critères :

- l'efficacité du filtrage (E_{tech}), c'est-à-dire le rapport entre le nombre de valeurs filtrées par une méthode et le nombre de valeurs contenu dans l'instance (i.e. m) ;
- la précision du minorant (P), c'est à dire le rapport entre le minorant et la valuation optimale (calculée séparément par une méthode de type recherche arborescente).

Les expérimentations sont faites avec D_z={$\mu^{\Sigma}_{dec}(\mathcal{A}^\star)$}, avec $\mathcal{A}^\star$ une solution de valuation optimale pour l'instance générée. Tous les résultats reportés dans la suite sont des moyennes sur 100 instances.

La Figure 6 compare l'efficacité des filtrages suivants :

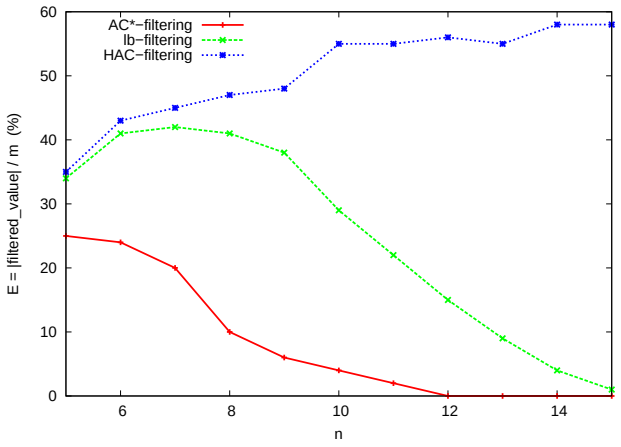

FIG. 6 – Efficacité du filtrage avec $|D_X|$=$n/2$.

$(n, \lvert D_X \rvert)$	(20,10)	(25,12)	(30,15)	(40,20)
tps(lb)	0.0040	0.0112	0.0204	0.0528
tps(opt)	5.8821	3170.5	—	—

TAB. 3 – Comparaison entre le temps de calcul de lb et de $\mu^{\Sigma\star}_{dec}$ (en secondes).

- un filtrage complet, qui pour chaque valeur, calcule en temps exponentiel la valuation optimale et retire la valeur du domaine de la variable si la valuation est supérieur à max(D_z) ;
- notre filtrage basée sur le calcul d'un minorant tel que présenté dans la section 4.2.5 ;
- le filtrage fournit par `AC3⋆` (pour plus de détails voir [8]) sur l'ensemble des contraintes binaires de la version décomposée de `AllDifferent`(μ_{dec}).

Le filtrage est effectué avec $|D_X|$ fixé à ($n/2$) de manière à ce que l'efficacité du filtrage par la méthode en temps exponentiel soit maximale, c'est-à-dire qu'il retire le plus grand nombre de valeurs.

On peut remarquer sur la Figure 6, que tant que le nombre de variables ne dépasse pas la dizaine alors notre filtrage est efficace et donc la consistance maintenue est forte. En revanche, au-delà l'efficacité décroît.

Il est à noter que notre filtrage est toujours au moins aussi efficace que le filtrage fourni par `AC3⋆`. Le Tableau 3 représente, pour différentes valeurs de (n,$|D_X|$), les temps de calcul nécessaires pour obtenir l'optimum et notre minorant.

La Figure 7 présente les variations de la précision du minorant en fonction de n et $|D_X|$. On peut remarquer que toutes les courbes ont le même comportement ;

- $|D_X|/n < 0.4$: Le problème est très sur-contraint, la détection des conflict-sets est très efficace et donc la complétion a une valeur faible, d'où un minorant de bonne qualité.

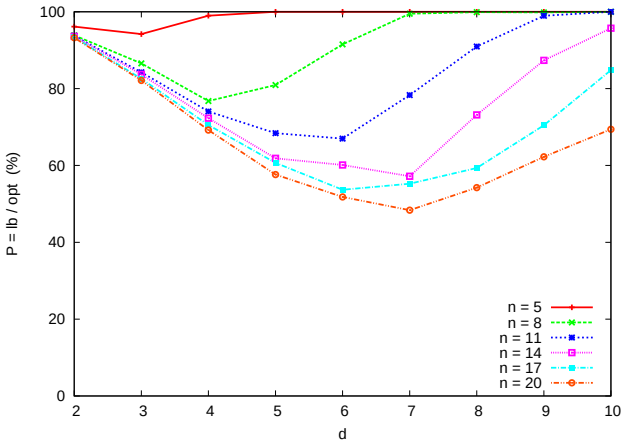

FIG. 7 – Précision du minorant.

– $0.5 < |D_X|/n < 0.6$: Dans cette zone, beaucoup de contraintes sont encore insatisfaites. Toutefois, peu de conflict-sets sont détectés, et la complétion a alors une valeur importante ce qui engendre beaucoup d'approximation et fait baisser la qualité du minorant.
– $|D_X|/n > 0.6$: Le problème est peu sur-contraint et donc peu de contraintes sont insatisfaites. Même si peu de conflict-sets sont détectés, la complétion est faible engendrant un minorant de bonne qualité.

5 Conclusions et travaux futurs

Dans cet article, nous avons présenté deux sémantiques de violation pour la relaxation de la contrainte `AllDifferent` dans le cadre avec préférences. Pour la sémantique basée variable, nous avons proposé un algorithme maintenant la consistance globale en $\mathcal{O}(n \times m)$. Pour la sémantique basée décomposition, nous avons montré que le test de consistance est un problème NP-Complet. C'est pourquoi nous avons proposé de maintenir une consistance locale grâce au calcul d'un minorant.

Dans de futurs travaux, nous souhaitons améliorer la qualité du filtrage et tester d'autres consistances pour améliorer la détection des conflict-sets (comme par exemple la consistance de singleton [13]). Nous souhaitons également étudier l'interaction entre les mécanismes de filtrage des `WCSP`s et la relaxation des contraintes globales (comme `AllDifferent`).

Remerciements

Nous remercions Thierry PETIT pour ses précieux conseils, Nicolas LEVASSEUR pour l'aide apportée pour la partie expérimentations ainsi que les rélecteurs anonymes pour leurs commentaires qui ont permis d'améliorer ce papier.

Références

[1] Claude Berge. *Graphes et Hypergraphes*. Dunod, 1970.
[2] S. Bistarelli, U. Montanari, F. Rossi, T. Schiex, G. Verfaillie, and H. Fargier. Semiring-based CSPs and Valued CSPs : Frameworks, Properties, and Comparaison. *Constraints*, 4(3) :199–240, 1999.
[3] Florin Dobrian, Mahantesh Halappanavar, Amit Kumar, and Alex Pothen. Vertex-weighted matching algorithms for computing column-space bases, CSC05, 2005. http ://www.cerfacs.fr/algor/CSC05/Abstracts/9_Dobrian.pdf.
[4] L.R. Ford and D.R. Fulkerson. Constructing Maximal Dynamic Flows From Static Flows. *Operations research*, 6(3) :419–433, 1958.
[5] E.C Freuder. Partial Constraint Satisfaction. *IJCAI-89*, 58(1–3) :21–70, 1989.
[6] John E. Hopcroft and Richard M. Karp. A $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. In *FOCS*, pages 122–125. IEEE, 1971.
[7] U Junker. Preferred Explanations and Relaxations for Over-Constrained Problems. *AAAI-2004*, 2004.
[8] J. Larrosa and T. Schiex. Solving weighted csp by maintaining arc consistency. *Artif. Intell.*, 159(1-2) :1–26, 2004.
[9] Jean-Philippe Métivier, Patrice Boizumault, and Samir Loudni. Σ-alldifferent : Softening alldifferent in weighted csps. In *ICTAI (1)*, pages 223–230. IEEE Computer Society, 2007.
[10] Gilles Pesant. A regular language membership constraint for finite sequences of variables. In Mark Wallace, editor, *CP*, volume 3258 of *Lecture Notes in Computer Science*, pages 482–495. Springer, 2004.
[11] T. Petit, C. Bessiere, and J.C. Régin. A General Conflict-Set Based Framework for Partial Constraint Satisfaction. *SOFT 2003*, 2003.
[12] Thierry Petit, Jean-Charles Régin, and Christian Bessière. Specific filtering algorithms for over-constrained problems. In Toby Walsh, editor, *CP*, volume 2239 of *Lecture Notes in Computer Science*, pages 451–463. Springer, 2001.
[13] Patrick Prosser, Kostas Stergiou, and Toby Walsh. Singleton consistencies. In Rina Dechter, editor, *CP*, volume 1894 of *Lecture Notes in Computer Science*, pages 353–368. Springer, 2000.
[14] Claude-Guy Quimper, Alejandro López-Ortiz, Peter van Beek, and Alexander Golynski. Improved algorithms for the global cardinality constraint. *CP 2004*, LNCS 3258 :542–556, 2004.
[15] J.C. Régin. A Filtering Algorithm for Constraints of Difference in CSPs. *AAAI-94*, pages 362–367, 1994.
[16] J.C. Régin. Generalized Arc Consistency for `Global Cardinality Constraint`. *AAAI-96*, pages 209–215, 1996.
[17] R Tarjan. Depth-First Search and Linear Graph Algorithms. *SIAM Journal on Computing*, pages 146–160, 1972.
[18] P. van Hentenryck, Y. Deville, and C.M. Teng. A Generic Arc Consistency Algorithm and its Specializations. *Artificial Intelligence*, 57(2–3) :91–321, 1992.
[19] W.J. van Hoeve. A Hyper-Arc Consistency Algorithm for the `Soft-AllDifferent` Constraint. *CP 2004*, LNCS 3258 :679–689, 2004.
[20] W.J. van Hoeve, G. Pesant, and L.M. Rousseau. On Global Warming : Flow-Based Soft Global Constraints. *Journal of Heuristics 12*, 4(5) :347–373, 2006.
[21] Alessandro Zanarini, Michela Milano, and Gilles Pesant. Improved algorithm for the soft global cardinality constraint. *CPAIOR 2006*, LNCS 3990 :288–299, 2006.