Application des techniques CSP au raisonnement sur les intervalles

Olivier Lhomme

Dassault Electronique[1] *55, Quai Marcel Dassault 92214 Saint-Cloud*

Michel Rueher

Université de Nice – Sophia Antipolis, I3S-CNRS, Route des Colles, B.P. 145, 06903 Sophia Antipolis Cedex

RÉSUMÉ. *Dans cet article nous montrons comment les techniques de réduction de problème développées pour les CSP peuvent être adaptées pour approximer les domaines de variation des variables dans des systèmes de contraintes non-linéaires définis sur les réels. Nous introduisons un cadre uniforme pour la représentation des domaines continus et des domaines discrets ordonnés. Différentes consistances partielles qui portent uniquement sur les bornes des intervalles associés à ces domaines sont définies et des algorithmes de filtrage pour leur mise en oeuvre sont proposés. Les relations entres ces nouvelles consistances partielles, la consistance d'arc et la k-consistance sont exposées. Les apports de ces techniques sont illustrées par une application qui a été réalisée avec le langage Interlog où ces différents algorithmes ont été implémentés. Enfin, nous montrons les liens entre notre démarche et différentes approches récentes proposées pour la résolution de systèmes de contraintes non-linéaires sur les réels.*

ABSTRACT. *This paper shows how CSP reduction techniques can be adapted for approximating the range of values of variables in non-linear constraints systems over the reals. A general framework for representing both continuous domains and finite ordered domains is introduced. We define several partial consistencies —as well as the corresponding filtering algorithms— which work only on the bounds of the intervals associated with such domains. The relationship between these new consistencies, arc consistency and k–consistency is investigated. The benefits of our approach are illustrated on an application which has been developed in Interlog, a Prolog-like interval narrowing language based upon these new consistencies. Finally, we outline the links between our framework and different approaches which have recently been proposed for solving non-linear constraints over the reals.*

MOTS-CLÉS : *Contraintes sur les domaines continus, réduction d'intervalles sur les réels, consistances sur les bornes*

KEY WORDS: *Constraints over continuous domains, interval narrowing, bound–consistency*

.

1. Adresse actuelle: École des Mines de Nantes 4 rue Alfred Kastler, La Chantrerie, 44070 Nantes Cedex 03.

1 Introduction

Les problèmes provenant du monde physique font fréquemment intervenir des contraintes numériques non linéaires, des données imprécises ou des paramètres mal définis. Ces problèmes sont généralement sous-contraints, *i.e.,* il existe un très grand nombre de solutions, voire un nombre infini si les domaines sont continus. Ce type de problème peut s'exprimer en termes de CSP [Wal72, Mon74, Mac77]: étant donné un ensemble de variables $\mathcal{V} = \{x_1, \ldots, x_n\}$, un ensemble de domaines associés $\mathcal{D} = \{D_{x_1}, \ldots, D_{x_n}\}$, un ensemble de contraintes numériques $\mathcal{C} = \{C_1, \ldots, C_m\}$, il s'agit de trouver les domaines de variation de chacune des variables, c'est-à-dire les projections sur chacune des variables de l'ensemble des solutions du CSP. Ces domaines de variation présentent souvent plus d'intérêt que des solutions ponctuelles.

> **Exemple:**
> Soient trois variables réelles liées par une contrainte $U = R \times I$. Si I varie entre 1 et 2 et que R varie entre 10 et 11, il est plus intéressant de connaître le domaine de variation de U (*i.e.,* l'intervalle $[10, 22]$) qu'une solution (e.g., $R = 10.53, I = 1, U = 10.53$).

Malheureusement, lorsque les contraintes sont non-linéaires, il n'existe pas de méthode générale pour calculer les domaines de variation des variables. Il est cependant possible de déterminer des approximations des domaines de variation. En particulier, les techniques de réduction de problème issues des CSPs [Mac77, Tsa93] peuvent être adaptées à cette fin. Une variante de AC-3 [Mac77], appelée ***algorithme de propagation d'intervalles***, est la technique de consistance la plus utilisée sur les domaines numériques, qu'ils soient discrets (et finis) comme dans `CHIP` [DvS+88] ou qu'ils soient continus [Dav87] comme dans `BNR-Prolog` [OV90], `Interlog` [BT93, Lho94], `Prolog IV` [Col94, BT95] ou `Newton` [BMV94].

L'algorithme de propagation d'intervalles utilise les propriétés des domaines numériques; en particulier il exploite l'existence de fonctions qui permettent de calculer directement la projection d'une contrainte sur une variable sans énumérer les valeurs de son domaine. L'arithmétique des intervalles [Moo66] est au coeur de l'implémentation de ces fonctions:

> **Exemple:**
> Si D_x, D_y et D_z sont des domaines numériques représentés par des unions d'intervalles, la projection sur la variable x de la contrainte $x + y = z$ est définie par $D_x \cap (D_z \ominus D_y)$ où l'opérateur $\ominus$ représente la soustraction sur des unions d'intervalles.

Toutefois, l'utilisation de cet algorithme pour des contraintes définies sur les réels pose quelques problèmes liées à la spécificité des nombres réels.

1.1 Représentation des nombres réels

Dans le cas des CSP sur les domaines finis, l'algorithme AC-3 converge vers un point fixe unique représentant la fermeture par consistance d'arc du CSP [JV88].
Sur les domaines continus, [GH88] ont montré qu'un traitement "équitable" [2] des contraintes est suffisant pour que l'algorithme de propagation d'intervalles converge vers un point fixe unique. Cependant, pour les domaines continus, la convergence vers un point fixe unique $PF_{\mathcal{R}}$ ne garantit pas que celui-ci puisse être atteint en un nombre fini d'itérations.

Exemple:
Soit $\begin{cases} y = x + 1 & \text{(a)} \\ y = 2 \times x & \text{(b)} \\ D_x = [0, 10] & D_y =]-\infty, +\infty[\end{cases}$
Les propagations suivantes vont être effectuées:
$D_x = [0, 10] \xrightarrow{\text{(a)}} D_y = [1, 11]$, $\quad D_y = [1, 11] \xrightarrow{\text{(b)}} D_x = [0.5, 5.5]$
$D_x = [0.5, 5.5] \xrightarrow{\text{(a)}} D_y = [1.5, 6.5]$, $\quad D_y = [1.5, 6.5] \xrightarrow{\text{(b)}} D_x = [0.75, 3.25]$
$\cdots$

Sur cet exemple un phénomène d'itération et de convergence asymptotique vers la solution se produit. La propagation d'intervalles se comporte ici comme un algorithme dichotomique, et la solution $\{x = 1, y = 2\}$ —toujours comprise dans les domaines D_x et D_y— n'est jamais atteinte [3].

L'implémentation des nombres réels reposant généralement sur une représentation en virgule flottante, et l'ensemble des nombres flottants étant fini, un point fixe $PF_{\mathcal{F}}$ est atteint en un nombre fini d'itérations. Toutefois, la représentation en virgule flottante conduit à de nouvelles questions:

- Le point fixe atteint dans une représentation en virgule flottante est-il unique?
- Quels sont les rapports entre $PF_{\mathcal{R}}$, le point fixe réel, et $PF_{\mathcal{F}}$, le point fixe atteint dans une représentation en virgule flottante, ou, en d'autres termes, quels sont les rapports entre les relations sur les réels et leur implémentation sous forme de fonctions sur des nombres flottants?

L'arithmétique des intervalles —utilisée dans l'algorithme de propagation d'intervalles pour approximer la projection d'une fonction— a initialement été développée pour contrôler rigoureusement la propagation des erreurs d'arrondis dues aux calculs en virgule flottante. Elle permet de répondre simplement aux questions précédentes: $PF_{\mathcal{F}}$ est unique si une incertitude $(-\varepsilon_1, +\varepsilon_2)$ est ajoutée au résultat R des fonctions flottantes de telle sorte que le résultat réel appar-

2. Un traitement "équitable" implique que toute réduction qui peut être opérée le sera effectivement; la gestion à l'aide d'une file FIFO des contraintes susceptibles de réduire les domaines est un cas particulier de traitement "équitable".

3. Une étude détaillé du problème des convergences asymptotiques peut être trouvée dans [LGR96].

tienne à l'intervalle $[R-\varepsilon_1, R+\varepsilon_2]$. De plus, $PF_{\mathcal{R}}$ est inclus dans $PF_{\mathcal{F}}$.
Pour formaliser précisément et de façon homogène les rapports entre les nombres réels et les nombres flottants, nous introduisons dans la section 2.1 une notion de ***valeur numérique abstraite*** qui capture aussi bien l'aspect continu des nombre réels que l'aspect discret des nombres flottants.

1.2 Consistances partielles sur les bornes

Dans un CSP numérique, un domaine D est ordonné, fini, mais sa cardinalité est en $O(n)$ où n est le nombre de flottants compris entre le minorant et le majorant de D. La convexité des domaines est de ce fait nécessaire pour autoriser une représentation en compréhension par les bornes de l'intervalle. Or le calcul de la projection d'une contrainte sur une variable peut détruire la convexité des domaines. Le nombre d'intervalles nécessaires pour définir ces domaines peut devenir très important.

Exemple:

Soit	$D_x = [0, 31416]$	$D_y = [0, 31416]$	$D_z =]-\infty, +\infty[$
	$\sin(x) = 0.2$	$\cos(y) = 0.1$	$z = x \times y$

Un filtrage par consistance d'arc entraînerait pour D_x et D_y l'union de 10^4 intervalles (10^4 est le nombre de parties monotones de `sinus` et `cosinus` dans [0,31416]) et pour D_z l'union de 10^8 intervalles.

L'explosion combinatoire, due à la représentation des domaines, peut être évitée en préservant la convexité des domaines. L'approche la plus répandue (e.g., [BO92, Lho93, BMV94, Ben95]) consiste ainsi à approximer le domaine d'une variable par l'intervalle qui englobe ce domaine. Elle se justifie principalement par un souci d'efficacité, mais aussi parce que pour les systèmes physiques, la zone de fonctionnement est souvent continue. Cette approche repose sur l'idée qu'il est souvent possible de réduire les unions d'intervalles sans avoir à les calculer explicitement. Elle peut aussi être considérée comme un pré-traitement au terme duquel les unions d'intervalles seront exploitées.
Pour caractériser le point fixe atteint dans cette approche nous introduisons la ***2–B–consistance*** qui peut être vue informellement comme une consistance d'arc limitée aux bornes des domaines.

L'implémentation des nombres réels par des nombres en virgule flottante garantit que le point fixe sera atteint après un nombre fini d'itérations. Toutefois, d'un point de vue opérationnel, il est nécessaire d'interrompre l'algorithme lorsque des convergences asymptotiques apparaissent ; ces phénomènes étant difficiles à identifier, l'algorithme est en général interrompu lorsque le temps imparti est épuisé, ou lorsqu'un domaine est restreint d'une quantité inférieure à un certain seuil. Pour caractériser le résultat effectivement fourni lorsque la propagation est interrompue avant que le point fixe ne soit atteint, nous introduisons la notion de ***2–B(w)–consistance***.

Pour les CSP sur des domaines finis, lorsque les relations sont exprimées

en extension, il existe des consistances plus fortes que la consistance d'arc: les k–consistances [Fre78]. Dans les CSP numériques, les relations —exprimées en compréhension— ne peuvent être résolues symboliquement, et les algorithmes de k–consistance ne sont pas applicables directement. Pourtant, ils permettraient d'obtenir de meilleures approximations des domaines de variation des variables. Afin de prendre en compte ce besoin de consistances plus fortes, nous proposons dans cet article une nouvelle famille de consistances partielles: les ***k–B–consistances***.

1.3 Plan de l'article

La section 2 est consacrée à la définition des valeurs numériques abstraites. Nous introduisons aussi dans cette section les notions de convexité d'un ensemble de valeurs numériques abstraites, de contraintes disjonctives et de NCSP (problème de satisfaction de contraintes numériques). La section 3 présente tout d'abord les notions de 2–B–consistance et 2–B(w)–consistance. Ces consistances partielles sont ensuite généralisées aux consistances partielles de degré k. Pour illustrer les apports des ces différentes techniques de consistance, la modélisation d'un échangeur de chaleur est présentée dans la section 4. Enfin, dans la section 5 nous comparons ces techniques de consistance avec la "box consistency" récemment introduite dans [BMV94] et nous abordons quelques prolongements possibles de ces travaux.

2 Approximation des réels et des contraintes sur les réels

2.1 Définition des valeurs numériques abstraites

La plupart des calculs numériques reposent sur une abstraction des nombres réels par des nombres flottants munis de mécanismes d'arrondis qui garantissent la correction des opérations. Nous introduisons ici la notion de valeur numérique abstraite (notée *vna* dans la suite du texte) qui permet de rester indépendant de la représentation des nombres manipulés et de l'implémentation en machine des fonctions mathématiques. Les *vna* fournissent également un modèle pour évaluer la complexité de l'algorithme de réduction d'intervalles sur des domaines continus. Enfin, elles offrent un cadre uniforme pour représenter à la fois les domaines discrets ordonnés et les domaines continus.

Définition 1 (valeur numérique abstraite). Soit $\mathcal{F} = \{f_0, f_1, \ldots, f_n\}$ un ensemble fini ordonné qui permet de représenter les nombres réels, une *valeur numérique abstraite* est un élément de l'ensemble $\mathcal{A}$:

$$\mathcal{A} = \{]-\infty, f_0[, f_0,]f_0, f_1[, f_1,]f_1, f_2[, f_2, \ldots, f_n,]f_n, +\infty[\}$$

En d'autres termes, une valeur numérique abstraite v est un objet qui représente un ensemble E de valeurs. Cet ensemble a l'une des formes suivantes:

- $E = \{f\}$ où f est un élément de $\mathcal{F}$,
- $E = \{x \mid x \in]f, f^+[\}$ où f et f^+ sont deux éléments consécutifs de $\mathcal{F}$,
- $E = \{x \mid x \in]-\infty, f_0[\}$
- $E = \{x \mid x \in]f_n, +\infty[\}$

La *vna* dénotée par l'intervalle $]f, f^+[$ représente un ensemble infini de réels mais chacune des *vna* ne constitue qu'un seul élément de $\mathcal{A}$. Les *vna* dénotées par $]-\infty, f_0[$ et $]f_n, +\infty[$ représentent les infinis mathématiques et, respectivement, les nombres plus petit que f_0 et ceux plus grands que f_n. La fonction $\alpha : \mathcal{R} \to \mathcal{A}$ est une fonction d'approximation qui associe à tout réel une *vna*. Nous pourrons ainsi considérer le domaine associé à une variable comme un ensemble fini tout en préservant la continuité attachée au domaine initial.

On notera qu'un ensemble fini discret $\mathcal{D}$ peut être représenté par lui même, *i.e.,* $\mathcal{F} = \mathcal{D}$. L'ensemble $\mathcal{A}$ des *vna* peut alors être défini par les éléments du domaine initial[4] car les ensembles $\{x \mid x \in]f, f^+[\}$ sont vides. Dans ce cas la fonction d'abstraction $\alpha : \mathcal{D} \to \mathcal{A}$ est la fonction idendité $\mathbb{I}$ [Ben95].

Définition 2 (intervalle). Soit v_1 et v_2 deux *vna*, un ***intervalle*** $[v_1, v_2]$ représente l'ensemble des *vna* comprises entre v_1 et v_2 :

$$[v_1, v_2] = \{x \in \mathcal{R} \mid v_1 \leq \alpha(x) \leq v_2\}$$

$\mathcal{I}(\mathcal{A})$ représente l'ensemble des intervalles définis sur $\mathcal{A}$ et $\mathcal{U}(\mathcal{A})$ représente l'ensemble des unions d'intervalles définis sur $\mathcal{A}$.

Notations:

Nous utiliserons les notations suivantes, éventuellement indicées :

- x, y et z représentent des variables réelles ;
- v désigne une *vna* alors que a et b désignent des nombres réels. v^{+k} (resp. v^{-k}) désigne la $k^{ième}$ *vna* plus grande (resp. plus petite) que v si elle existe ; si elle n'existe pas, v^{+k} (resp. v^{-k}) sera la *vna* maximale (resp. minimale) ;
- C et $C(x_1, \ldots, x_n)$ désignent une contrainte, $var(C)$ désigne l'ensemble des variables de C et $\rho(C)$ désigne le sous-ensemble des valeurs de $\mathcal{R}^n$ qui satisfont la contrainte.

4. Pour que la complexité des opérations élémentaires reste indépendante de la valeur des nombres manipulés, il faut toutefois limiter le nombre de chiffres disponibles pour représenter une *vna*.

2.2 Définition des fonctions et relations sur les *vna* et les intervalles

Une *vna* représentant un ensemble de valeurs réelles, le résultat d'une fonction appliquée à cet ensemble ne peut pas toujours être approximé par une seule *vna*. Les fonctions appliquées aux *vna* devront de ce fait être définies sur $\mathcal{P}(\mathcal{A})$.

Définition 3 (extension aux *vna*). Une fonction $g_{\mathcal{A}} : \mathcal{P}(\mathcal{A})^n \rightarrow \mathcal{P}(\mathcal{A})$ est une extension aux *vna* de $g_{\mathcal{R}} : \mathcal{R}^n \rightarrow \mathcal{R}$ ssi :

$$\forall x_1 \dots x_n \in \mathcal{R},\ \alpha(g_{\mathcal{R}}(x_1, \dots, x_n)) \in g_{\mathcal{A}}(\{\alpha(x_1)\}, \dots, \{\alpha(x_n)\})$$

Une relation $r_{\mathcal{A}} \subseteq \mathcal{A}^n$ est une extension aux *vna* d'une relation $r_{\mathcal{R}} \subseteq \mathcal{R}^n$ ssi:

$$(a_1, \dots, a_n) \in r_{\mathcal{R}} \Rightarrow (\alpha(a_1), \dots, \alpha(a_n)) \in r_{\mathcal{A}}$$

Soit C une contrainte, on note $\tilde{\rho}(C)$, la plus petite (au sens de l'inclusion) extension aux *vna* de la relation $\rho(C)$ [5].

Le traitement des deux *vna* particulières $]-\infty, f_0[$ et $]f_n, +\infty[$ nécessite uniquement l'ajout d'une fonction de manipulation symbolique des infinis aux méthodes de calcul de l'arithmétique des intervalles.

Définition 4 (extension aux intervalles). Une fonction $G_{\mathcal{A}} : \mathcal{U}(\mathcal{A})^n \rightarrow \mathcal{U}(\mathcal{A})$ est une extension aux intervalles de $g_{\mathcal{A}} : \mathcal{P}(\mathcal{A})^n \rightarrow \mathcal{P}(\mathcal{A})$ ssi:

$\forall v_1 \ \dots \ v_n \in \mathcal{A},$

$$[(v_1 \in I_1 \ \wedge \ \dots \ \wedge \ v_n \in I_n) \Rightarrow g_{\mathcal{A}}(\{v_1\}, \dots, \{v_n\}) \subseteq G_{\mathcal{A}}(I_1, \dots, I_n)]$$

Une relation $R_{\mathcal{A}} \subseteq \mathcal{A}^n$ est une extension aux intervalles d'une relation $r_{\mathcal{A}} \subseteq \mathcal{A}^n$ ssi:

$\forall v_1 \ \dots \ v_n \in \mathcal{A},$

$$[(v_1, \dots, v_n) \in r_{\mathcal{A}} \wedge v_1 \in I_1 \wedge \dots \wedge v_n \in I_n \Rightarrow (I_1, \dots, I_n) \in R_{\mathcal{A}}]$$

En résumé, la notion de valeur numérique abstraite fournit une abstraction qui approxime le domaine de calcul. $\alpha : \mathcal{R} \rightarrow A$ peut ainsi être vu comme une fonction d'abstraction qui vérifie les propriétés suivantes:

(1) $\forall x \in \mathcal{R}, x \in \alpha(x)$

(2) $\forall r_{\mathcal{R}} \subseteq \mathcal{R}^n, \alpha(r_{\mathcal{R}}) \subseteq \mathcal{A}^n$

(3) $\forall r_{\mathcal{R}} \subseteq \mathcal{R}^n, \alpha(r_{\mathcal{R}}) \supseteq \{< \alpha(x_1), \dots, \alpha(x_n) > \mid < x_1, \dots, x_n > \in \ r_{\mathcal{R}}\}$

5. Le calcul de $\tilde{\rho}(C)$ soulève quelques questions délicates. Si $\mathcal{F}$ est l'ensemble des nombres flottants, il est nécessaire pour calculer $\tilde{\rho}(C)$ que les fonctions flottantes rendent un résultat correct "à un flottant près". Or, si c'est le cas pour l'addition, il n'en est pas de même pour des fonctions comme $\sin x$ où l'erreur de calcul des fonctions fournies par les bibliothèques standards peut être importante. Certes, il est toujours possible d'obtenir un résultat à une *vna* près, mais ceci exige l'utilisation de flottants de tailles variables et modifie significativement le coût des opérations élémentaires. Une solution moins coûteuse — et souvent satisfaisante d'un point de vue opérationnel— consiste à répercuter l'erreur de calcul sur les résultats fournis par les différentes fonctions. On peut ainsi associer à chaque fonction une primitive spécifique qui détermine l'erreur associée au résultat v en fonction des données et "élargit" en conséquence ce dernier à l'intervalle $[v^{-k_1}, v^{+k_2}]$ où k_1 et k_2 dépendent de la précision des calculs. Pour que le point fixe en virgule flottante $PF_{\mathcal{F}}$ encadre au plus près le point fixe réel $PF_{\mathcal{R}}$ il est nécessaire de choisir k_1 et k_2 les plus petits possible. Dans cette seconde approche $\tilde{\rho}(C)$ ne correspond plus à la plus petite extension aux *vna* de la relation $\rho(C)$ mais à une approximation de cette extension.

Des rapports très étroits existent entre la physique qualitative [FCBM89, DT96] et l'abstraction fournie par les *vna*. De fortes similarités existent aussi avec les fonctions d'abstraction et de concrétisation utilisées en interprétation abstraite [BS88].

2.3 Convexité d'un ensemble de *vna* — Contraintes disjonctives

Définition 5 (domaine convexe). Nous dirons que D_x, le domaine d'une variable x, est convexe ssi il contient toutes les *vna* comprises entre $\texttt{min}(D_x)$ et $\texttt{max}(D_x)$, *i.e.,* un ensemble D de *vna*, est convexe sur $\mathcal{A}$ ssi:

$$\forall v \in \mathcal{A}, \texttt{min}(D) \leq v \leq \texttt{max}(D) \Rightarrow v \in D$$

> **Exemple:**
> L'ensemble $D = \{1, 2, 3, 4\}$ est convexe sur $\mathcal{A}$ si $\mathcal{A}$ correspond au sous ensemble des entiers inférieurs à 100, mais D est non convexe si $A = \{0, 0.5, 1, 1.5, \ldots, 99.5, 100\}$.

Si $D_{x_1}, \ldots, D_{x_k}$ sont des domaines convexes, alors $E = D_{x_1} \times \cdots \times D_{x_k}$ sera dénommé un ***bloc***.
Soit $C(x_1, \ldots, x_k)$ une contrainte et $E = D_{x_1} \times \cdots \times D_{x_k}$ un bloc défini sur les domaines des variables correspondantes. Pour tout i dans $[1..k]$, la projection sur x_i de la contrainte $C(x_1, \ldots, x_k)$, notée $\Pi_{x_i}(C)$, est définie par:
$\Pi_{x_i}(C) = \{v_i \in D_i \mid \exists\ (v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_k) \in D_{x_1} \times \cdots \times D_{i-1} \times D_{i+1} \times \cdots \times D_{x_k} \wedge\ < v_1, .., v_k > \in \tilde{\rho}(C)\}$

> **Exemple:**
> Soit $\{y = x^2,\ D_x = [-2, +2],\ D_y = [1, 10]\}$, $\Pi_y(y = x^2) = [1, 4]$ et $\Pi_x(y = x^2) = [-2, -1] \cup [1, 2]$.

Définition 6 (contrainte disjonctive). Une contrainte $C(x_1, \ldots, x_k)$ est disjonctive sur le bloc $E = D_{x_1} \times \cdots \times D_{x_k}$ ssi:
$\exists i \in [1..k]$ tel que $\Pi_{x_i}(C(x_1, \ldots, x_k))$ soit un ensemble non convexe de *vna*.

> **Exemple:**
> Sur $D_x \times D_y$, la contrainte $x^2 = y$ est:
> - non disjonctive si $D_x = [-2, 2]$ et $D_y = [0, 4]$
> - disjonctive si $D_x = [-2, 2]$ et $D_y = [1, 4]$,
> - non disjonctive si $D_x = [1, 2]$ et $D_y = [1, 4]$.

Définition 7 (contrainte monotone). On appelle contrainte monotone, une contrainte non disjonctive dont toutes les fonctions de projection sont des fonctions monotones.

On notera dans l'exemple précédent que la contrainte $x^2 = y$ n'est pas monotone pour $\{D_x = [-2, 2],\ D_y = [0, 4]\}$ mais qu'elle est monotone pour $\{D_x = [1, 2],\ D_y = [1, 4]\}$.

2.4 Définition des contraintes basiques

Définition 8 (contrainte basique). Une contrainte $C(x_1, \ldots, x_k)$ est dite basique si pour tout i entre 1 et k, il est possible d'exhiber deux fonctions $\texttt{Min}_{x_i}(C)$ et $\texttt{Max}_{x_i}(C)$ telles que pour tout bloc $E = D_{x_1} \times \cdots \times D_{x_k}$:

- $\texttt{Min}_{x_i}(C) = \texttt{min}(\Pi_{x_i}(C))$
- $\texttt{Max}_{x_i}(C) = \texttt{max}(\Pi_{x_i}(C))$.

Les fonctions $\texttt{Min}_{x_i}(C)$ et $\texttt{Max}_{x_i}(C)$ seront dénommées les fonctions extremum solutions de la contrainte C. L'ensemble des contraintes basiques est infini, mais les contraintes numériques usuelles peuvent être implantées avec un sous-ensemble réduit de contraintes basiques. Nous désignerons par Γ ce sous-ensemble et nous supposerons par la suite que toutes les contraintes appartiennent à Γ.

> **Exemple:**
> Dans l'implantation actuelle de `Interlog`, Γ contient les contraintes qui correspondent aux opérateurs arithmétiques, trigonométriques et logarithmiques usuels, *i.e.*, $\Gamma \supseteq \{x = y, x \leq y, x < y, x \neq y, z = x + y, z = x \times y, y = -x, y = \sin x, y = \cos x, y = e^x, y = \text{abs}(x), z = x^y, z = \min(x, y), z = \max(x, y)\}$.
> De plus Γ contient les contraintes ensemblistes de la forme $x \in E$ avec $E \in \mathcal{U}(\mathcal{A})$. Pour ces contraintes l'arithmétique des intervalles fournit les fonctions extremum solutions.

On notera que l'introduction de nouvelles variables permet toujours de réécrire une contrainte non basique, construite à partir des opérateurs mathématiques élémentaires, en une conjonction de contraintes basiques.

> **Exemple:**
> $e^x = \sin(x + y)$ se réécrit en $(e^x = x'_1) \wedge (x'_1 = \sin x'_2) \wedge (x'_2 = x + y)$.

Il est toutefois important de souligner que si dans une telle transformation, la consistance globale [Hyv92] est invariante, une propriété locale comme la consistance d'arc n'est pas nécessairement invariante.

2.5 Définition d'un NCSP

Nous appellerons NCSP un problème de satisfaction de contraintes numériques. Les définitions correspondantes sont directement dérivées de celles des CSPs [Mac77]; la seule différence réside dans le fait que les contraintes ne peuvent pas être données en extension.

Définition 9 (NCSP). Un NCSP $P = (\mathcal{V}, \mathcal{D}, \mathcal{C})$ sur un ensemble de *vna* $\mathcal{A}$ est défini par

(1) un ensemble de variables $\mathcal{V} = \{x_1, \ldots, x_n\}$

(2) un ensemble de domaines $\mathcal{D} = \{D_{x_1}, \ldots, D_{x_n}\}$ où $D_{x_i} \in \mathcal{I}(\mathcal{A})$ pour tout i dans $1..n$ (D_{x_i} est le domaine associée à la variable x_i);

(3) un ensemble de contraintes $\mathcal{C} = \{C_1, \ldots, C_m\}$

On notera que les domaines D_{x_i} sont ici des intervalles. Si le domaine initial d'une variable x n'est pas convexe, alors une contrainte de la forme $x \in E$ avec $E \in \mathcal{U}(\mathcal{A})$ est ajoutée à $\mathcal{C}$.

Exemple:
La figure 1 représente les contraintes de trois NCSPs, P_1, P_2 et P_3, qui ne diffèrent que par leurs domaines.

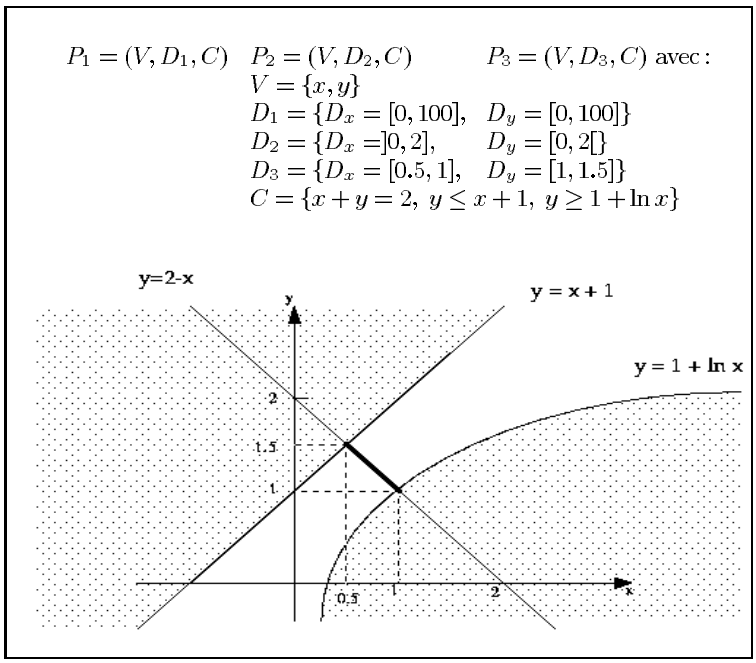

FIG. 1 – *Contraintes de P_1, P_2 et P_3*

Une solution d'un NCSP $P = (\mathcal{V}, \mathcal{D}, \mathcal{C})$ est une fonction $\Theta : \mathcal{V}^n \to \mathcal{A}^n$: $\Theta(x_1, \ldots, x_n) = (v_1, \ldots, v_n)$ tel que:
$\forall\, C_i(x_{j_1}, \ldots, x_{j_k}) \in \mathcal{C}, < v_{j_1}, \ldots, v_{j_k} > \in \tilde{\rho}(C_i) \cap (D_{x_{j_1}} \times \cdots \times D_{x_{j_k}})$
Ainsi, $(1, 1)$ est une solution de P_1 (figure 1).

L'ensemble des solutions de P sera dénoté $Sol(P)$. La projection de l'ensemble des solutions de P sur la variable x, notée $Sol_x(P)$, est définie par:

$$Sol_x(P) = \{v \mid \exists\, \Theta \in Sol(P) \wedge \Pi_x(\Theta) = v\}$$

P est globalement consistant ssi $\forall x \in V, D_x = Sol_x(P)$. La consistance globale traduit le fait que pour toute variable x du problème, le domaine D_x est l'extension aux *vna* du domaine de variation de x. Contrairement à la définition

de la consistance globale de Freuder [Fre78] sur les CSPs, cette définition permet de traiter des contraintes données en compréhension.

P est vide ssi il existe un domaine D_x de $\mathcal{D}$ qui soit vide. Si P est un NCSP vide, alors $Sol(P) = \varnothing$. On notera $P_\varnothing$ la classe des NCSP dont un au moins des domaines est vide.

Deux NCSPs P et P' sont équivalents ($P \equiv P'$) ssi $Sol(P) = Sol(P')$. Par exemple, $P_1 \equiv P_2 \equiv P_3$.

On utilisera un pré-ordre $\preceq$ entre $P = (\mathcal{V}, \mathcal{D}, \mathcal{C})$ et $P' = (\mathcal{V}, \mathcal{D}', \mathcal{C})$ et on notera $P \preceq P'$ si $P \equiv P_\varnothing$, ou, si $P \not\equiv P_\varnothing$ et les domaines de P sont inclus dans ceux de P'.

Nous utiliserons la terminologie inspirée de [Jég91] pour parler des techniques de consistances. Une consistance partielle λ (e.g., λ = consistance d'arc ou consistance de chemin) est une propriété d'un NCSP. Le filtrage par λ d'un NCSP $P = (\mathcal{V}, \mathcal{D}, \mathcal{C})$, notée $\Phi_\lambda(P)$ est un NCSP $P' = (\mathcal{V}, \mathcal{D}', \mathcal{C})$ qui vérifie au moins les conditions suivantes:

(1) $P' \equiv P$

(2) P' est λ–consistant

(3) $P' \preceq P$

Un algorithme de filtrage par λ calcule $\Phi_\lambda(P)$ et est habituellement dénommé technique de réduction ou de consistance.

3 Réductions des intervalles

Rappelons que notre objectif n'est pas d'énumérer toutes les solutions d'un NCSP, ni de résoudre formellement son système de contraintes, mais de calculer les domaines de variation des différentes variables. Toutefois, la détermination d'un NCSP globalement consistant, équivalent au NCSP initial est un problème NP-difficile [6]. Nous limiterons donc notre ambition à la recherche d'une approximation majorante du domaine de variation des variables. L'algorithme de [Dav87], inspirée de AC3, fournit une telle approximation. Cette section introduit d'abord deux consistances partielles:

(1) la 2–B–consistance qui formalise le point fixe vers lequel converge cet algorithme,

(2) la 2–B(w)–consistance qui caractérise le résultat fournit lorsque cet algorithme est interrompu prématurément.

Nous généralisons ensuite ces deux consistances partielles à la 3–B–consistance (resp. 3–B(w)–consistance) et à la k–B–consistance (resp. k–B(w)–consistance)

6. C'est un problème indécidable sur le corps des réels (à cause des fonctions transcendantes) mais en précision finie le problème devient NP-difficile (une énumeration est possible).

et nous décrivons les algorithmes correspondant qui permettent de calculer une meilleure approximation des domaines de variation des variables.

3.1 2–B–consistance

La 2–B–consistance est une consistance partielle qui garantit les propriétés suivantes:

Définition 10 (2–B–consistance). Soient $P = (\mathcal{V}, \mathcal{D}, \mathcal{C})$ un NCSP, x une variable de $\mathcal{V}$ dont le domaine D_x est $[a, b]$. D_x est 2–B–consistant ssi:

$$\forall C(x, x_1, \ldots, x_k) \in \mathcal{C}, \quad \exists v_1, \ldots, v_k \in D_{x_1} \times \cdots \times D_{x_k} | < a, v_1, \ldots, v_k > \in \tilde{\rho}(C)$$
$$\wedge \exists v'_1, \ldots, v'_k \in D_{x_1} \times \cdots \times D_{x_k} | < b, v'_1, \ldots, v'_k > \in \tilde{\rho}(C)$$

Un NCSP est 2–B–consistant ssi tous ses domaines sont 2–B–consistants.

> **Exemple:**
> Considérons le NCSP $P_2 = (\{x, y\}, \{D_x =]0, 2], \ Dy = [0, 2[\}, \{x + y = 2, \ y \leq x + 1, \ y \geq 1 + \ln x\})$. P_2 est *2–B–consistant.*
> Pour le montrer, il faut d'abord montrer que D_x est 2–B–consistant. Comme $D_x =]0, 2]$, il faut donc que pour $x =]0, 0^+[$ et pour $x = 2$ les trois contraintes de P_2 soient vérifiables. Si $x =]0, 0^+[$, la *vna* $]2^-, 2[$ de D_y vérifie la contrainte $\{x + y = 2\}$ car il existe au moins un couple de réels (r_1, r_2) dans ces deux *vna* tels que $r_1 + r_2 = 2$, la valeurs 0 de D_y vérifie la contrainte $\{y \leq x + 1\}$, la valeur 1 de D_y vérifie la contrainte $\{y \geq 1 + \ln x\}$ car $\ln x$ sur $]0, 0^+[< 0$. Pour $x = 2$, on peut suivre la même démarche. Il en résulte que D_x est 2–B–consistant. Une démonstration analogue permet de montrer que D_y est aussi 2–B–consistant.

Les définitions de consistance d'arc et de 2–B–consistance sont comparables: la consistance d'arc impose une condition sur tous les éléments des domaines alors que la 2–B–consistance impose cette même condition aux seules bornes de l'intervalle qui contient les domaines. La 2–B–consistance englobe dans un seul intervalle le domaine d'une variable, et certaines valeurs de ce domaine pourront donc être localement inconsistantes.

> **Exemple:**
> Soit P défini par
> $D_x = [1, 4], \ D_y = [-2, +2]$
> $x = y^2$
> P est 2–B–consistant, mais pas consistant d'arc car la valeur 0 de D_y n'a pas de support dans D_x qui puisse satisfaire la contrainte.

La 2–B–consistance et la consistance d'arc sont très liées: si P ne contient pas de contraintes disjonctives sur les domaines considérés, alors P est 2–B–consistant ssi P est consistant d'arc [Lho94].

Si toute les contraintes sont des contraintes basiques, la 2–B–consistance est identique à la "hull-consistency" introduite dans [BMV94] et à la "interval consistency" définie par [VDT92] pour les domaines finis entiers.

Définition 11 (filtrage par 2–B–consistance). Un filtrage par 2–B–consistance d'un NCSP $P = (\mathcal{V}, \mathcal{D}, \mathcal{C})$, noté $\Phi_{2BC}(\mathrm{P})$, est un NCSP $P' = (\mathcal{V}, \mathcal{D}', \mathcal{C})$ tel que:

(1) $P' \equiv P$,

(2) P' est 2–B–consistant,

(3) $P' \preceq P$,

(4) $\forall \ P'' = (\mathcal{V}, \mathcal{D}'', \mathcal{C}), \{P'' \equiv P \ \wedge \ (P'' \text{ est 2–B–consistant}) \ \wedge \ P'' \preceq P \ \wedge \ P' \preceq P''\} \Rightarrow D'' = D'$.

Exemple:
$P_2 = \Phi_{2BC}(P_1)$.

Le point 4 de cette définition traduit le fait que le filtrage par 2–B–consistance est le plus grand NCSP vérifiant les 3 premières conditions. Il en découle que le filtrage par 2–B–consistance est unique. La figure 2 présente un algorithme incrémental de filtrage par 2–B–consistance directement dérivé de l'algorithme de Davis [Dav87]. Le domaine d'une variable étant un ensemble de *vna*, il est possible d'analyser la complexité de cet algorithme. Soit d le nombre de *vna* du plus grand domaine de $\mathcal{D}$, soit m_n le nombre de contraintes de `New_constraints`, et soit m le nombre total de contraintes de P. [Lho94] montre que la borne inférieure de la complexité en temps de 2–B–filtering est en $O(m_n)$ et la borne supérieure est en $O(d \times m)$. Si toutes les contraintes sont passées dans `New_constraints` (*i.e.*, $m_n = m$) alors la borne inférieure devient $O(m)$. On remarquera que cette borne inférieure de la complexité de 2–B–filtering est indépendante de d.

3.2 2–B(w)–consistance

Lorsque l'ensemble des *vna* $\mathcal{A}$ approxime un domaine continu, 2–B–filtering peut conduire à des problèmes de convergence asymptotique. Ainsi, l'algorithme est en général interrompu avant la terminaison normale. Ces interruptions prématurées empêchent de caractériser précisément les domaines calculés: le NCSP résultant n'est pas 2–B–consistant. La notion de *2–B(w)–consistance* que nous introduisons ici permet de remédier à ce problème. La 2–B(w)–consistance est une consistance partielle qui généralise la 2–B–consistance: w est un paramètre qui caractérise l'imprécision admise sur les bornes (w peut être vu comme une "largeur de borne", cf. figure 3). L'idée sous-jacente est de garantir qu'il existe, pour chaque contrainte, des valeurs proches des bornes des domaines qui vérifient la contrainte, tout en laissant ces valeurs non spécifiées.

Définition 12 (2–B(w)–consistance). Soient un NCSP $P = (\mathcal{V}, \mathcal{D}, \mathcal{C})$, x une variable de $\mathcal{V}$ dont le domaine D_x est $[a, b]$, w un entier positif. D_x est 2–B(w)–consistant ssi:

$$\forall C(x, x_1, \ldots, x_k) \in C, \exists v \in [a, a^{+w}], \ \exists v_1, \ldots, v_k \in D_{x_1} \times \cdots \times D_{x_k} \mid < v, v_1, \ldots, v_k > \in \tilde{\rho}(C)$$
$$\wedge \exists v' \in [b^{-w}, b], \ \exists v'_1, \ldots, v'_k \in D_{x_1} \times \cdots \times D_{x_k}$$

function 2–B–filtering(P_0, New_constraints)
Pré-condition : $P_0 = (\mathcal{V}, \mathcal{D}, \mathcal{C})$ est un NCSP 2–B–consistant.
Post-condition : Le résultat est un NCSP P
et $P = \Phi_{2BC}(P_0 \cup \text{New\_constraints}) \vee P = P_\varnothing$

```
1   P = (V, D, C ∪ New_constraints)
2   Queue ← {< C, x > | C ∈ New_constraints ∧ x ∈ var(C)}
3   while Queue ≠ ∅
4       pop < C, x > from Queue
5       Result ← Narrowing(< C, x >)
6       if Result = fail then return P_∅
7       if Result ≠ unchanged then D_x ← Result
8           Queue ← append(Queue, {< C', x' > | C' ∈ C
9                   ∧ C' ≠ C ∧ x ∈ var(C') ∧ x' ∈ var(C')})
10  return P
```

function Narrowing$(< C, x >)$ % $\mathcal{D} = \{D_{x_1}, \ldots, D_{x_n}\}$

```
1   if Min_x(C) = ∅ ∨ Max_x(C) = ∅ then Result ← fail
2   else    m ← Min_x(C)
3           M ← Max_x(C)
4           if [m, M] ≠ D_x then Result ← [m, M]
5           else Result ← unchanged
6   return Result
```

FIG. 2 – *Algorithme de filtrage par 2–B–consistance*

$| < v', v'_1, \ldots, v'_k > \in \tilde{\rho}(C)$

Un NCSP est 2–B(w)–consistant ssi tous ses domaines sont 2–B(w)–consistants.

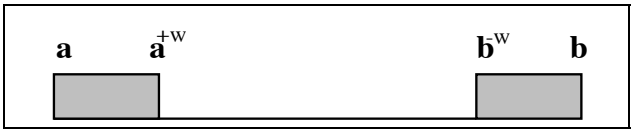

FIG. 3 – *Illustration de la notion de largeur de bornes*

La définition de la 2–B(w)–consistance subsume la définition de la 2–B–consistance: pour $w = 0$, la 2–B(w)–consistance est équivalente à la 2–B–consistance. On remarquera que si P est 2–B(w)–consistant, et $w' \geq w$, alors P est 2–B(w')–consistant.

$P' = (\mathcal{V}, \mathcal{D}', \mathcal{C})$ sera appelé filtrage par 2-B(w)–consistance de $P = (\mathcal{V}, \mathcal{D}, \mathcal{C})$ s'il vérifie:

(1) $\Phi_{2BC}(P) \preceq P' \preceq P$,

(2) P' est 2-B(w)–consistant.

On notera $P' = \Phi_{2BC(w)}(P)$.

Contrairement au filtrage par 2–B–consistance, le filtrage par 2–B(w)–consistance dépend de l'ordre d'évaluation des contraintes et n'est donc pas unique (la propriété 4 de la définition 11 n'est pas vérifiée). En particulier, lorsque le filtrage par 2–B–consistance est le NCSP vide, il peut arriver que deux ordres d'évaluation des contraintes dans l'algorithme de filtrage par 2–B(w)–consistance conduisent l'un à un NCSP vide, l'autre à un NCSP non vide.

Un algorithme qui calcule un filtrage par 2–B(w)–consistant peut être dérivé de l'algorithme 2–B–filtering en imposant dans la procédure Narrowing que la réduction du domaine ne soit effectuée que lorsque celle-ci est supérieure à w. Si d est la taille maximale des domaines et m le nombre de contraintes, une analyse de complexité de cet algorithme montre que la borne inférieure de la complexité en temps est en $O(m)$ et la borne supérieure est en $O(\frac{m \times d}{w+1})$ [Lho94].

La 2–B–consistance —comme la consistance d'arc— n'est qu'une consistance partielle et n'est pas toujours suffisante pour calculer le domaine de variation des variables:

> **Exemple:**
> P_2 (cf. figure 1) est 2–B–consistant (et même consistant d'arc), mais les domaines des variables sont seulement une approximation des domaines de variation (*i.e.*, $D_x = [0.5, 1]$, $D_y = [1, 1.5]$).

La démarche usuelle consiste à utiliser un algorithme d'énumération de type "domain–splitting" [Cle87]. Cet algorithme —proche des méthodes numériques de "bi–section"[Bli92] et de "tightening" [HS94]— pose de multiples points de choix et peut conduire à une explosion combinatoire.

Les limites de la consistance d'arc sur les domaines finis sont à l'origine des travaux sur les consistances partielles plus fortes. La consistance de chemin [Mon74, Mac77] a ainsi été introduite, puis a été généralisée par la k-consistance [Fre78]. De manière similaire, nous introduisons ici les notions de ***3*–B–consistance** et de ***k*–B–consistance.** Ces consistances partielles fournissent de meilleures approximations du domaine de variation des variables que celles fournies par la 2–B–consistance. Nous verrons dans la section suivante que les algorithmes de filtrage associés à ces consistances sont tout à fait utilisables en pratique.

3.3 La 3–B–consistance et la 3-B(w_1, w_2)–consistance

Comme la 2–B–consistance, la 3–B–consistance n'impose des conditions que sur les bornes des domaines afin d'éviter toute explosion combinatoire. Informellement, la 3–B–consistance peut être considérée comme une forme de 3-consistance forte [Fre78] restreinte aux bornes des domaines.

Soit un NCSP $P = (\mathcal{V}, \mathcal{D}, \mathcal{C})$, on notera $P \uplus \{D_{x_i} = [a,a]\}$ le NCSP $P_1 = (\mathcal{V}, \{D_{x_1}, \ldots, D_{x_{i-1}}, [a,a], D_{x_{i+1}}, \ldots, D_{x_n}\}, \mathcal{C})$.

Définition 13 (3–B–consistance). Soient $P = (\mathcal{V}, \mathcal{D}, \mathcal{C})$ un NCSP, x une variable de $\mathcal{V}$ dont le domaine D_x est $[a,b]$. D_x est 3–B–consistant ssi:

(1) $\Phi_{2BC}(P \uplus \{D_x = [a,a]\}) \neq \varnothing$

(2) $\Phi_{2BC}(P \uplus \{D_x = [b,b]\}) \neq \varnothing$

Un NCSP est 3–B–consistant ssi tous ses domaines sont 3–B–consistants.

Il découle de cette définition que si $P = (\mathcal{V}, \mathcal{D}, \mathcal{C})$ est 3–B–consistant, alors P est 2–B–consistant. Il en résulte aussi que si P est fortement 3–consistant [Tsa93], alors P est 3–B–consistant.

Définition 14 (filtrage par 3–B–consistance). Un filtrage par 3–B–consistance de P est un NCSP $P' = (\mathcal{V}, \mathcal{D}', \mathcal{C})$ qui vérifie :

(1) $P' \equiv P$,

(2) P' est 3–B–consistant,

(3) $P' \preceq P$,

(4) $\forall\ P'' = (\mathcal{V}, \mathcal{D}'', \mathcal{C}), \{P'' \equiv P \wedge (P'' \text{ est 3–B–consistant}) \wedge P'' \preceq P \wedge P' \preceq P''\} \Rightarrow D'' = D'$.

Le filtrage par 3–B–consistance est unique [Lho94] et sera noté Φ_{3BC}(P). $\Phi_{3BC}(P) \preceq \Phi_{2BC}(P)$ et un filtrage par 3–B–consistance fournit donc une approximation plus précise des domaines de variation qu'un filtrage par 2–B–consistance.

> **Exemple:**
> P_1 et P_2 (cf. figure 1) ne sont pas 3–B–consistants mais P_3 est 3–B–consistant. De plus $P_3 = \Phi_{3BC}(P_1) = \Phi_{3BC}(P_2)$.

De manière analogue à la 2–B–consistance, la notion de 3–B–consistance se généralise à la 3–B(w_1, w_2)–consistance qui permet de donner une largeur aux bornes. Du fait de la non-unicité du filtrage par 2–B(w)–consistance, la définition de la ***3*-B(w_1, w_2)–consistance** nécessite l'utilisation de quantificateurs existentiels.

Définition 15 (3-B(w_1, w_2)–consistance). Soient un NCSP $P = (\mathcal{V}, \mathcal{D}, \mathcal{C})$, x une variable de $\mathcal{V}$ dont le domaine D_x est $[a,b]$, deux entiers positifs w_1 et w_2. D_x est 3-B(w_1, w_2)–consistant ssi:

(1) $\exists w_{11} \mid (0 \leq w_{11} \leq w_1)$ et il existe $\Phi_{2BC(w_2)}$ $(P \uplus \{D_x = [a, a^{+w_{11}}]\}) \neq \varnothing$

(2) $\exists w_{12} \mid (0 \leq w_{12} \leq w_1)$ et il existe $\Phi_{2BC(w_2)}$ $(P \uplus \{D_x = [b^{-w_{12}}, b]\}) \neq \varnothing$

Un NCSP est 3-B(w_1, w_2)–consistant ssi tous ses domaines sont 3-B(w_1, w_2)–consistants.

On remarquera que la 3–B–consistance est équivalente à la 3-B(0,0)–consis-

tance. Si P est 3-B(w_1, w_2)–consistant, $w_{1'} \geq w_1, w_{2'} \geq w_2$ alors P est 3-B$(w_{1'}, w_{2'})$–consistant.
$P' = (\mathcal{V}, \mathcal{D}', \mathcal{C})$ sera appelé filtrage par 3-B(w_1, w_2)–consistance de $P = (\mathcal{V}, \mathcal{D}, \mathcal{C})$ s'il vérifie:

(1) $\Phi_{3BC}(P) \preceq P' \preceq P$,

(2) P' est 3-B(w_1, w_2)–consistant.

On notera $P' = \Phi_{3BC(w_1,w_2)}(P)$. Un filtrage par 3-B$(w_1, w_2)$–consistance n'est pas unique.

Avant de donner un algorithme de filtrage par 3–B–consistance et par 3–B(w_1, w_2)–consistance nous généralisons ces notions à la k–B–consistance.

3.4 La k–B–consistance et la k–B$(w_1, w_2, \ldots, w_{k-1})$–consistance

La k–B–consistance va généraliser les consistances partielles précédentes. La k–B–consistance continue de garantir des conditions uniquement sur les bornes des domaines. Informellement elle peut être considérée comme une forme de k-consistance forte [Fre78] restreinte aux bornes des domaines.

La définition de la k–B–consistance est récursive. Dans la définition ci-dessous $\Phi_k(P)$ dénote le filtrage par k–B–consistance de P (*i.e.,* $\Phi_2(P)$ dénote $\Phi_{2BC}(P)$).

Définition 16 (k–B–consistance). Soient un NCSP $P = (\mathcal{V}, \mathcal{D}, \mathcal{C})$, x une variable de $\mathcal{V}$ dont le domaine D_x est $[a, b]$. D_x est k–B–consistant (avec $k > 2$) ssi:

(1) $\Phi_{k-1}(P \uplus \{D_x = [a, a]\}) \neq \varnothing$

(2) $\Phi_{k-1}(P \uplus \{D_x = [b, b]\}) \neq \varnothing$

Un NCSP est k–B–consistant ssi tous ses domaines sont k–B–consistants.

Un filtrage par k–B–consistance de $P = (\mathcal{V}, \mathcal{D}, \mathcal{C})$ est un NCSP $P' = (\mathcal{V}, \mathcal{D}', \mathcal{C})$ qui vérifie:

(1) $P' \equiv P$,

(2) P' est k–B–consistant,

(3) $P' \preceq P$,

(4) $\forall\ P' = (\mathcal{V}, \mathcal{D}', \mathcal{C}), \{P'' \equiv P \wedge (P''$ est k–B–consistant$) \wedge P'' \preceq P \wedge P' \preceq P''\} \Rightarrow D'' = D'$.

Le filtrage par k–B–consistance est unique. Si P est k–B–consistant, alors P est h–B–consistant pour $2 \leq h \leq k$. De plus, $\Phi_k(P) \preceq \Phi_h(P)$.

Définition 17 (B–consistance globale). Soit $P = (\mathcal{V}, \mathcal{D}, \mathcal{C})$, x une variable de $\mathcal{V}$ dont le domaine D_x est $[a, b]$. D_x est globalement B-consistant ssi $a \in Sol_x(P)$ et $b \in Sol_x(P)$. La B-consistance globale garantit que le domaine de

chaque variable est le plus petit intervalle contenant son domaine de variation (les bornes du domaine sont donc des solutions). P est globalement B-consistant ssi tous ses domaines sont globalement B-consistants.

Un algorithme de filtrage par B-consistance globale permettrait de décider de la satisfiabilité d'un système de contraintes numériques quelconque. Dans le cadre théorique fourni par les *vna* la satisfiabilité d'un tel système se traduit par un problème NP-difficile (une énumération est toujours possible). Il est néanmoins intéressant de mettre en évidence les relations existant entre k–B–consistance et B-consistance globale: si $P = (\mathcal{V}, \mathcal{D}, \mathcal{C})$ et $|\mathcal{V}| = n - 1$, et P est n-B-consistant, alors P est globalement B-consistant.

La notion de k–B–consistance est maintenant généralisée par la k–B(w_1, w_2, ..., w_{k-1})–consistance de façon à autoriser une imprécision sur les bornes. On notera $\Phi_{k(w_1,\ldots,w_{k-1})}(P)$ un filtrage par k–B($w_1, \ldots, w_{k-1}$)–consistance de P ; par exemple $\Phi_{2(w)}(P) = \Phi_{2BC(w)}(P)$.

Définition 18 (k–B($w_1, w_2, \ldots, w_{k-1}$)–consistance). Soient $P = (\mathcal{V}, \mathcal{D}, \mathcal{C})$, x une variable de P dont le domaine D_x est $[a, b]$, et $w_1, \ldots w_{k-1}$ des entiers positifs. D_x est k–B($w_1, w_2, \ldots, w_{k-1}$)–consistant ssi:

(1) $\exists w_{11} \mid (0 \leq w_{11} \leq w_1)$ et il existe $\Phi_{k(w_2,\ldots,w_{k-1})}(P \uplus \{D_x = [a, a^{+w_{11}}]\}) \neq \varnothing$

(2) $\exists w_{12} \mid (0 \leq w_{12} \leq w_1)$ et il existe $\Phi_{k(w_2,\ldots,w_{k-1})}(P \uplus \{D_x = [b^{-w_{12}}, b]\}) \neq \varnothing$

Un NCSP est k–B($w_1, w_2, \ldots, w_{k-1}$)–consistant ssi tous ses domaines sont k–B($w_1, w_2, \ldots, w_{k-1}$)–consistants. La k–B–consistance est équivalente à la k–B(0,0, ... 0)–consistance. Il eut été possible d'imposer $w_1 = \cdots = w_{k-1}$ pour n'avoir qu'un seul niveau de "largeur de bornes" et simplifier la définition précédente. Les expérimentations réalisées (cf. section 4) montrent toutefois que la différenciation des w est très utile et permet d'améliorer les performances.

$P' = (\mathcal{V}, \mathcal{D}', \mathcal{C})$ sera appelé filtrage par k–B($w_1, w_2, \ldots, w_{k-1}$)–consistance de $P = (\mathcal{V}, \mathcal{D}, \mathcal{C})$ ssi:

(1) $\Phi_k(P) \preceq P' \preceq P$,

(2) P' est k–B($w_1, w_2, \ldots, w_{k-1}$)–consistant.

Un filtrage par k–B($w_1, w_2, \ldots, w_{k-1}$)–consistance n'est pas unique. Si P est k–B($w_1, w_2, \ldots, w_{k-1}$)–consistant et $w'_1 \leq w_1, w'_2 \leq w_2, \ldots, w'_{k-1} \leq w_{k-1}$, alors P est k–B($w'_1, w'_2, \ldots, w'_{k-1}$)–consistant.

Un algorithme de filtrage par k–B–consistance est présenté dans la figure 4. Cet algorithme utilise le point fort de l'abstraction fournie par les *vna* qui permet de déduire l'absence de solutions sur le domaine initial de l'absence de solutions sur l'ensemble des *vna*. La fonction k–B–filtering détermine la taille de la plus grande partie qui peut être ajoutée (resp. retranchée) aux bornes de chaque

domaine sans perdre de solutions. Il s'agit d'une recherche dichotomique qui s'arrête lorsque la réduction devient inférieure à w. La fonction k–B–fixpoint découle directement de la définition du filtrage par k–B($w_1, w_2, \ldots, w_{k-1}$)–consistance. On remarquera qu'aucun point de choix n'a été posé, contrairement aux méthodes de "domain–splitting" [Cle87] ou de "bi–section" [Bli92].

Si m est le nombre de contraintes, n le nombre de variables, d la taille des domaines, et pour $i \in [1..k-1]$ $w_i = w$, alors en posant $a = d/w$, la complexité est en $O(a^{k-1} \times m \times n^{2(k-2)})$.

Il est important de noter que les B-consistances d'ordre supérieur sont utilisables en pratique, et ce malgré une taille des domaines a priori très grande. Ceci provient du fait que:

- la k–B–consistance manipule uniquement les bornes des domaines,
- les paramètres w_i fournissent une graduation supplémentaire (en quelque sorte orthogonale au degré k) qui permet, en tolérant une imprécision sur les bornes des domaines, de paramétrer la complexité des algorithmes.

4 Application: un échangeur de chaleur

4.1 Présentation du problème

L'exemple de l'échangeur de chaleur est emprunté à [Dag93a, Dag93b]. Il est illustré en figure 5. Un fluide chaud et un fluide froid se croisent dans l'échangeur. $DTc, DTf, DT1$ et $DT2$ sont des différences de température. Qc et Qf sont les débits molaires respectifs du fluide chaud et du fluide froid. Cpc et Cpf sont les capacités calorifiques respectives du fluide chaud et du fluide froid.

Les contraintes numériques suivantes traduisent la relation sur les différences de températures et l'équilibre thermique de l'échangeur:

$$DTc - DT1 - DTf + DT2 = 0$$
$$DTc \times Cpc \times Qc = DTf \times Cpf \times Qf \quad .$$

Le système a d'autres contraintes, qui sont des contraintes qualitatives (e.g. "x et y sont comparables"). Pour obtenir un raisonnement précis, il est possible de transformer les contraintes qualitatives en contraintes numériques (e.g. "x et y sont comparables" devient $x = k \times y$, où k est un paramètre, variant par exemple dans $[1/5, 5]$).

Le problème est de trouver les relations qualitatives entre $DT2$ et DTc, DTc et DTf, $DT2$ et DTf, *i.e.,* les domaines de variation de x, y et z :

$$x = DT2/DTc, \quad y = DTc/DTf, \quad z = DT2/DTf$$

La figure 6 récapitule l'ensemble des contraintes qui caractérise un échangeur de chaleur. Les informations supplémentaires résultent des contraintes physiques.

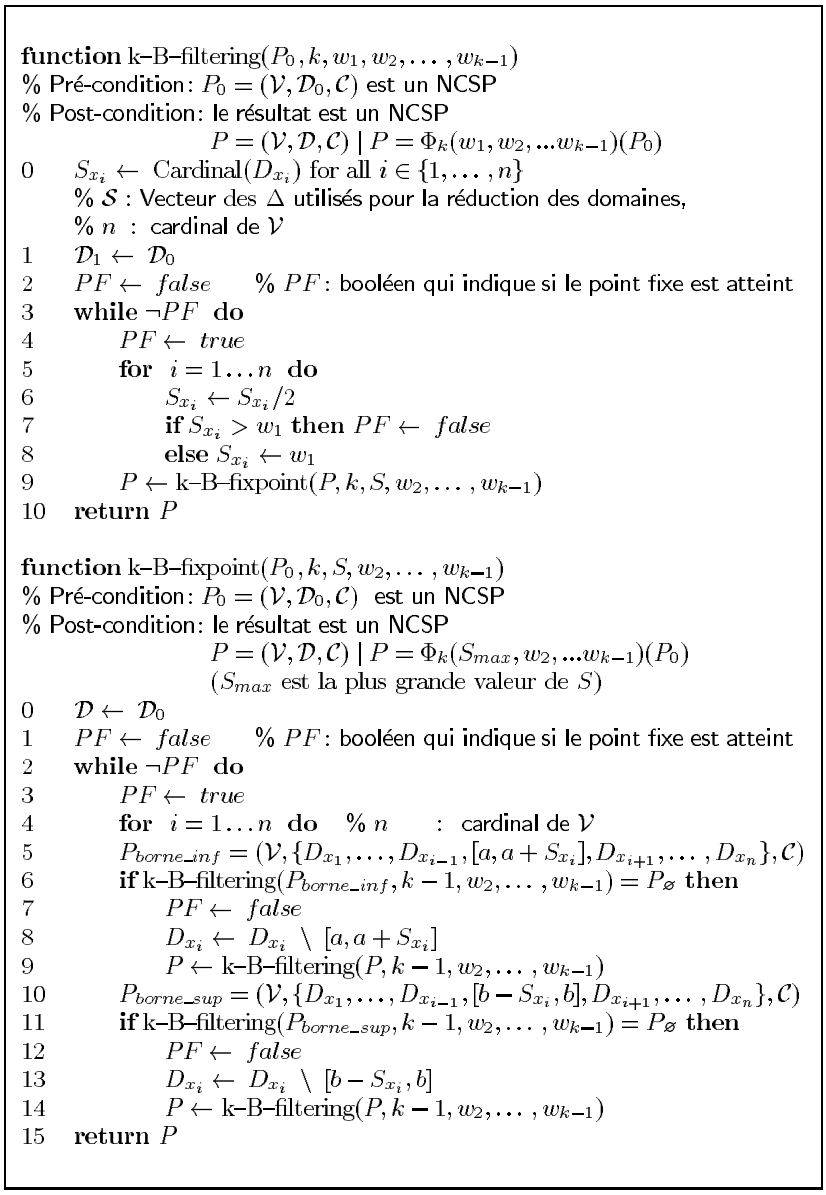

FIG. 4 – *Algorithme de filtrage par k–B–consistance*

4.2 Résultats

La 2–B–consistance ne permet que de réduire D_x à $[0, 50]$, D_y à $[0, +\infty]$ et D_z à $[0, +\infty]$. Un filtrage par 3–B–consistance donne les résultats présentés

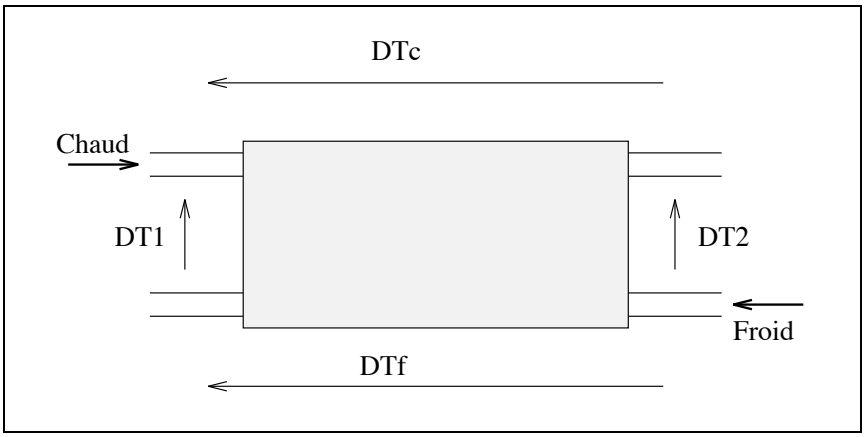

FIG. 5 – *Un échangeur de chaleur*

Contraintes qui traduisent l'équilibre thermique

$$DTc - DT1 - DTf + DT2 = 0$$
$$DTc \times Cpc \times Qc = DTf \times Cpf \times Qf.$$

Contraintes qualitatives

$DT2 = k_1 \times DT1,$	où	$k_1 \in [1/5, 5]$
$DT1 = k_2 \times DTc,$	où	$k2 \in [0, 1/10]$
$Cpc = k3 \times Cpf,$	où	$k3 \in [9/10, 10/9]$
$x = DT2/DTc$	$y = DTc/DTf$	$z = DT2/DTf$

Informations supplémentaires

$D_{DT1} = [1, 10^3]$	$D_{DT2} = [10^{-1}, 10^3]$	$D_{DTc} = [10^{-3}, 10^3]$
$D_{DTh} = [10^{-3}, 10^3]$	$D_{Fc} = [10^{-20}, 10^{20}]$	$D_{Fh} = [10^{-20}, 10^{20}]$
$D_{Kc} = [10^{-20}, 10^{20}]$	$D_{Kh} = [10^{-20}, 10^{20}]$	

FIG. 6 – *Ensemble des contraintes de l'échangeur de chaleur*

en figure 7. Pour les valeurs considérées, $val(w)$ correspond à la taille $|b - a|$ de l'intervalle $[a, b]$ qui contient w *vna*. Remarquons que D_x converge vers $[0, 0.5]$, D_y converge vers $[0.6666, 1.111]$ et D_z converge vers $[0, 0.54]$. Le domaine de variation de x est déterminé et ceux de y et z sont approximés avec une précision qui peut être suffisante.

Un filtrage par 4-B-consistance permet de déterminer les domaines de variation de y et z (cf. figure 8): D_y converge vers $[0.7143, 1.0870]$ et D_z converge vers $[0, 0.3571]$. Ces différents calculs demandent moins de 3 minutes dans l'implémentation actuelle sur IBM 3090.

		D_x	D_y	D_z
$\text{val}(w_1) = 0.2,$	$val(w_2) = 10^{-4}$	$[0, 0.581]$	$[0.486, 1.152]$	$[0, 0.715]$
$\text{val}(w_1) = 0.05,$	$val(w_2) = 10^{-4}$	$[0, 0.512]$	$[0.665, 1.135]$	$[0, 0.576]$
$\text{val}(w_1) = 0.01,$	$val(w_2) = 10^{-4}$	$[0, 0.502]$	$[0.665, 1.120]$	$[0, 0.558]$
$\text{val}(w_1) = 0.002,$	$val(w_2) = 10^{-4}$	$[0, 0.5009]$	$[0.665, 1.111]$	$[0, 0.553]$

FIG. 7 – *Résultats de la 3-b(w_1, w_2)–consistance*

		D_y	D_z
$\text{val}(w_1) = 0.2,$	$\text{val}(w_2) = 0.2$	$[0.509, 1.207]$	$[0, 0.507]$
$\text{val}(w_1) = 0.05,$	$\text{val}(w_2) = 0.05$	$[0.665, 1.135]$	$[0, 0.382]$
$\text{val}(w_1) = 0.01,$	$\text{val}(w_2) = 0.05$	$[0.705, 1.090]$	$[0, 0.362]$

FIG. 8 – *Résultats de la 4-b(w_1, w_2, w_3)–consistance avec val(w_3) = 10^{-4}*

4.3 Comparaison des différents algorithmes

Comme le montre cet exemple, la complexité expérimentale des algorithmes de filtrage par k–B–consistance est généralement très inférieure à la borne supérieure fournie par l'analyse théorique. Les premières expérimentations montrent que le temps de calcul ne croît pas toujours lorsque k croît. Ceci résulte du fait qu'un choix pertinent des w_i peut permettre d'éviter certains problèmes de convergence lente: l'algorithme de filtrage par 3-B(w_1, w_2)–consistance fait appel à un filtrage par 2–B(w_2)–consistance, si ce dernier converge lentement vers la solution, l'idée est de fixer w_2 de façon à ce que le filtrage par 2–B(w_2)–consistance n'entraîne pas de convergence lente. Le filtrage par 3–B–consistance sera ainsi plus précis et moins coûteux que le filtrage par 2–B–consistance.

La figure 9 exprime les relations entre la précision du résultat et le coût de calcul des k–B–consistances. L'axe des ordonnées traduit l'écart à la solution, c'est à dire la distance[7] entre les domaines de variation (plus précisément les intervalles définis par les bornes des domaines de variation) et les domaines courants. L'axe des abscisses représente le temps de calcul. Une courbe, annotée "k–B–consistance" traduit la convergence asymptotique de l'algorithme de k–B–consistance vers un point fixe d'écart e_k. Nous savons que $e_{k+1} \leq e_k$. Supposons $e_4 < e_3 < e_2$. Par conséquent, la courbe "2–B–consistance" et la courbe "3–B–consistance" se croisent. Soit A cette intersection. A partir d'un temps t_A, un

7. Cette distance entre 2 n-uplets d'intervalles peut être par exemple la distance maximale entre deux intervalles d'indice identique dans les deux n-uplets. Quant à la distance entre deux intervalles, nous pouvons, par exemple, considérer: $d([a,b],[c,d]) = \max(|a-c|, |b-d|)$ [Moo66].

filtrage par 3–B–consistance donnera de meilleurs résultats qu'un filtrage par 2–B–consistance. De la même façon, il existe un point B, intersection entre la courbe "3–B–consistance" et la courbe "4-B-consistance". L'asymptote de la courbe "B-consistance globale" est bien sûr l'axe $y = 0$. Dans l'état actuel de l'implémentation, le choix de la k–B–consistance et des paramètres w_i les plus appropriés est faite manuellement. Les possibilités d'une automatisation de cette recherche restent un problème ouvert.

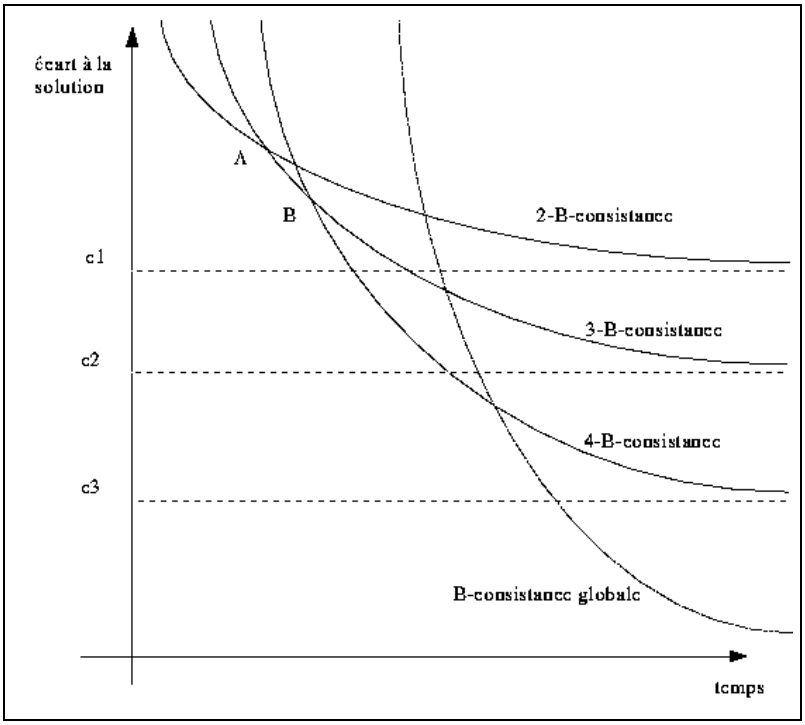

FIG. 9 – *Rapidité de convergence*

5 Discussion

Dans le cas d'une contrainte comme $x + \log x = 0$, calculer la projection de la contrainte sur x consiste à résoudre l'équation. Or on ne dispose pas de méthodes symboliques pour la résolution de contraintes non-polynômiales. Dans cette section, nous discutons d'abord des différentes approches mises en oeuvre pour traiter ce problème. Puis nous comparons la "box-consistency" récemment introduite dans [BMV94] avec la 2–B–consistance et la k–B–consistance. Enfin, nous montrons comment nos travaux s'intègrent dans le schéma $\mathcal{CLP}(\mathcal{X})$.

5.1 Calcul des projections

Deux approches sont actuellement proposées pour calculer la projection d'une contrainte:

(1) Calculer une ***approximation majorante*** de cette projection,

(2) Calculer la projection par une ***méthode numérique***.

La première approche est la plus répandue. La variante la plus simple — retenue dans cet article et utilisée dans les langages `BNR-Prolog` [OV93], `CLP(BNR)` [BO92] `Interlog` [Lho93, Lho94] et `Prolog IV` [Col94, BT95]— exploite directement les propriétés de l'arithmétique des intervalles. Une contrainte portant sur n variables ou occurences de variables[8] peut se ré-écrire en n fonctions de projection, éventuellement multivaluées. Une approximation de la projection de la contrainte sur une variable définie par p fonctions de projection correspond alors à l'intersection de ces p fonctions évaluées dans l'arithmétique des intervalles [Hyv92]. Ceci revient en fait à décomposer une contrainte en un ensemble de contraintes basiques qui correspondent à des opérateurs élémentaires (cf. section 2.4). Notons $\beta(\mathcal{C})$ le système de contraintes basiques qui résulte de la ré-écriture du système de contraintes initial $\mathcal{C}$.

Exemple:
Soit $\mathcal{C} = \{x + logx = 0\}$, $\beta(\mathcal{C}) = \{x = -x_1, x_1 = logx\}$

Les algorithmes de filtrage utilisés dans les systèmes mentionnés ci-dessus établissent alors la consistance partielle non pas sur le système de contraintes initial (*i.e.,* $\mathcal{C}$), mais sur les contraintes basiques issues d'une telle décomposition, *i.e.,* $\beta(\mathcal{C})$.

Une variante de cette première approche consiste à calculer une approximation de la projection d'une contrainte [BMV94] en utilisant la méthode de Newton. L'algorithme correspondant permet alors d'établir une consistance partielle qui prend en compte les contraintes initiales $\mathcal{C}$ et non la décomposition $\beta(\mathcal{C})$.

Les travaux de Faltings [Fal94] s'inscrivent dans le contexte de la seconde approche. Un algorithme dédié calcule la projection non seulement d'une contrainte mais d'un ensemble de contraintes. Cet algorithme impose cependant que les contraintes soient binaires (il n'est pas généralisable aux contraintes n-aires), différentiables et qu'elles portent sur le même couple de variables.

Il est important de noter que [BMV94, Fal94] utilisent un algorithme pour calculer ou approximer chaque projection alors que dans les autres approches ce travail est effectué par l'évaluation d'une fonction dans l'arithmétique des intervalles.

8. Les occurences multiples d'une variable sont considérées comme des variables distinctes mais qui ont le même domaine.

5.2 k–B–consistance *versus* "Box-consistency"

La "box-consistency" [BMV94] repose sur un algorithme de filtrage qui travaille directement sur $\mathcal{C}$ et qui approxime la projection sur les variables des contraintes initiales.

Soit C une contrainte, on note $\vec{\rho}(C)$, la plus petite (au sens de l'inclusion) extension aux intervalles de la relation $\tilde{\rho}(C)$.

Schématiquement on peut dire qu'une variable x vérifie la box-consistency ssi:

- $D_x = [a, b]$,
- Pour toute contrainte $C(x, x_1, \ldots, x_k)$ portant sur x on a $([a, a], D_{x_1}, \ldots, D_{x_k}) \in \tilde{\rho}(C) \wedge ([b, b], D_{x_1}, \ldots, D_{x_k}) \in \tilde{\rho}(C)$.

La box-consistency est une consistance moins forte que la 2–B–consistance:

$$\Phi_{2BC}(< \mathcal{V}, \mathcal{D}, \mathcal{C} >) \preceq \Phi_{Box}(< \mathcal{V}, \mathcal{D}, \mathcal{C} >)$$

Toutefois, il faut noter que la box-consistency travaille sur les contraintes initiales alors que la 2–B–consistance travaille sur des contraintes basiques. Rappelons que $\beta(\mathcal{C})$ désigne l'ensemble des contraintes basiques —générées à partir de $\mathcal{C}$— pour lesquelles l'arithmétique des intervalles fournit les fonctions extremum solution (cf. section 2.4). On obient alors :

- $\Phi_{Box}(< \mathcal{V}, \mathcal{D}, \mathcal{C} >) \preceq \Phi_{2BC}(< \mathcal{V}, \mathcal{D}, \beta(\mathcal{C}) >)$
- $\Phi_{2BC}(< \mathcal{V}, \mathcal{D}, \beta(\mathcal{C}) >) \equiv \Phi_{Box}(< \mathcal{V}, \mathcal{D}, \mathcal{C} >)$ si les contraintes $\mathcal{C}$ sont basiques.

La box-consistency fournit donc de meilleures approximations des domaines de variation des variables. De ce point de vue, la box-consistency peut être considérée comme une consistance plus forte que la 2–B–consistance et peut être comparée aux k–B–consistances. Nous effectuons ci-dessous cette comparaison en examinant successivement la portée des traitements, les méthodes de propagation et les algorithmes.

La box-consistency impose une condition sur les bornes du domaine d'une variable localement à une contrainte. La 3–B–consistance assure une condition sur les bornes du domaine d'une variable "globalement" au système de contraintes. Il résulte des définitions respectives de ces consistances que $\Phi_{3B}(< V, D, \beta(C) >) \preceq \Phi_{Box}(< V, D, C >)$ si l'implantation des filtrages utilise les mêmes fonctions arithmétiques.

> **Exemple:**
> Soit le système $\mathcal{C} = \{y + x - y = 0\}$, $D_x = [-1, 1]$, $D_y = [0, 1]$ extrait de [BMV94]. Ce système de contraintes est box-consistant. La 2–B–consistance est aussi vérifiée sur $\beta(\mathcal{C})$. Par contre un filtrage par 3–B–consistance sur $\beta(\mathcal{C})$ trouve les domaines de variation exacts.

Le principe des deux algorithmes de filtrage est très similaire: il consiste à réduire un domaine par les bornes. La différence essentielle, outre les points

précédents, réside dans une optimisation de ce principe de réduction dans l'algorithme de filtrage par box-consistance proposé dans [BMV94]. Cette optimisation repose sur l'utilisation, au coeur de l'algorithme, de la méthode de Newton et permet d'espérer pour certaines classes de contraintes des convergences très rapides.

Une nouvelle consistance qui intègre ces deux techniques peut ainsi être définie: les réductions du domaine seraient propagées lors du traitement d'une contrainte et l'algorithme pourrait utiliser la méthode de Newton. L'intérêt d'une telle consistance doit toutefois être évalué expérimentalement.

5.3 Relations avec le schéma $\mathcal{CLP}(\mathcal{X})$

Dans [JM94], Jaffar et Lassez ont montré comment les contraintes sur les domaines finis de CHIP peuvent être intégrées dans le schéma $\mathcal{CLP}$. $\mathcal{CLP}(\mathcal{FD})$ est une instance de ce schéma sur la structure des entiers où chaque variable apparaissant dans une contrainte peut être caractérisée par un intervalle. Dans le schéma $\mathcal{CLP}$, la sémantique opérationnelle d'un système s'exprime à l'aide de règles de transitions sur des états. Ces règles sont définies par une combinaison de transitions $\rightarrow_r$ (transitions induites par la résolution), transitions $\rightarrow_c$ (intégration d'une contrainte), transitions $\rightarrow_s$ (test de consistance sur les contraintes actives) et transitions $\rightarrow_i$ (inférence de nouvelles contraintes actives).

[vE94] introduit la notion de sous-ensemble représentable d'un domaine et défini un opérateur de compatibilité pour un CSP générique ("generic constraint satisfaction problem") dont les CSP sur les réels ne sont qu'une instance. Dans un CSP sur les réels, l'opérateur de compatibilité est l'opérateur de "Narrowing" sur les intervalles. Ce schéma lui permet de préciser la sémantique des transitions $\rightarrow_i$ et $\rightarrow_s$ pour un langage de programmation logique avec des contraintes sur les intervalles. Les k–B(w)–consistances proposées dans cet article pourraient être utilisées pour caractériser les états qui résultent de la mise en œuvre de ces transitions. Un intervalle défini sur l'ensemble $\mathcal{A}$ des *vna* correspond à la notion de plus petit sous-ensemble représentable.

6 Conclusion

Dans cet article nous avons proposé un cadre uniforme pour la représentation des domaines continus et des domaines discrets ordonnés. Différentes consistances partielles qui portent uniquement sur les bornes des intervalles associés à ces domaines ont été introduites. L'exemple présenté dans la section 4 illustre non seulement les apports des k–B(w)–consistances, mais il montre aussi que la complexité expérimentale des algorithmes de filtrage par k–B(w)–consistance (avec $k > 3$) autorise leur mise en œuvre effective sur des applications réelles.

Les méthodes de filtrage par k–B–consistance peuvent être considérées comme des algorithmes de résolution de systèmes de contraintes numériques. Cependant ces méthodes n'intègrent pas de traitement dédié aux problèmes particuliers que l'on sait résoudre symboliquement (e.g., contraintes linéaires, équations du second degré). L'étude de nombreuses applications montre toutefois que les techniques numériques et symboliques pourraient être utilisées de manière complémentaire (e.g., certaines convergences lentes pourraient être évitées par des simplifications formelles). Un prolongement possible de ces travaux réside ainsi dans une combinaison des méthodes symboliques et numériques qui peut être envisagée de différentes manières:

- Une coopération entre un solveur symbolique et un solveur par propagation d'intervalles qui communiquent de manière asynchrone. Chaque solveur peut ainsi utiliser directement les réductions de domaine et simplifications effectuées par l'autre. Une telle coopération est proposée dans [Rue95, MR95].
- Une intégration de méthodes de propagation d'intervalles et de résolution symbolique dans un nouvel algorithme. [BB95] proposent ainsi une intégration des techniques de réductions de domaine dans un algorithme du simplexe avec bornes explicites. [CL94] proposent quant à eux un algorithme de propagation d'intervalle pour des systèmes linéaires dérivé de la méthode de Gauss.

Remerciements

Patrick Taillibert, Bernard Botella et Serge Varennes ont apporté beaucoup à ce travail. Nous remercions aussi Hélène Collavizza, Narendra Jussien, Dan Vlasie ainsi que les relecteurs pour leurs commentaires.

Références

[BB95] H. Beringer and B. De Backer. Combinatorial problem solving in constraint logic programming with cooperative solvers. In C. Beierle and L. Plumer, editors, *Logic Programming: Formal Methods and Practical Applications*. North Holland, 1995. (Studies in Computer Science and Artifical Intelligence, Volume 11).

[Ben95] F. Benhamou. Interval Constraint Logic Programming. In Andreas Podelski, editor, *Constraint Programming: Basics and Trends*, LNCS 910. Springer-Verlag, 1995. (Châtillon-sur-Seine Spring School, France, May 1994).

[Bli92] C. Bliek. *Computer Methods for Design Automation*. PhD thesis, Massachusetts Institute of Technology, 1992.

[BMV94] F. Benhamou, D. McAllester, and P. Van Hentenryck. Clp(intervals) revisited. In *Logic Programming: Proceedings of the 1994 International Symposium*, Ithaca, US, 1994. MIT Press.

[BO92] F. Benhamou and W. Older. Applying interval arithmetic to integer and boolean constraints. Technical report, Bell Northern Research, 1992. To appear in The Journal of Logic Programming.

[BS88] M. Bruynooghe and D. De Schreye. Tutorial notes for: Abstract interpretation in logic programming. In *ICLP 88*, page 25, 1988.

[BT93] B. Botella and P. Taillibert. Interlog: constraint logic programming on numeric intervals. In *3rd International Workshop on Software Engineering, Artificial Intelligence and Expert Systems*, Oberammergau, October 1993.

[BT95] F. Benhamou and Touraïvane. Prolog IV: langage et algorithmes. In *Proc. de JFPL'95: 4ème Journées Francophones de Programmation en Logique*, pages 51–64, Dijon, Mai 1995. Teknea.

[CL94] C. K. Chiu and J. H. M. Lee. Towards practical interval constraint solving in logic programming. In *ILPS'94: Proceedings 11th International Logic Programming Symposium*, Ithaca, November 1994.

[Cle87] J.C. Clearly. Logical arithmetic. *Future Computing Systems*, 2(2):125–149, 1987.

[Col94] A. Colmerauer. Un cadre juridique pour discuter de la résolution approchée de contraintes. document de travail, 1994.

[Dag93a] P. Dague. Numeric reasonning with relative orders of magnitude. In *Proc. AAAI, Washington*, 1993.

[Dag93b] P. Dague. Symbolic reasonning with relative orders of magnitude. In *Proc. IJCAI93, Chambery, (France)*, August 1993.

[Dav87] E. Davis. Constraint propagation with interval labels. *Artificial Intelligence*, 32(3):281–331, July 1987.

[DT96] P. Dague and L. Travé-Massuyes. *Raisonnement qualitatif pour les sciences de l'ingénieur*. Masson, 1996. à paraître.

[DvS+88] M. Dincbas, P. van Hentenryck, H. Simonis, A. Aggoun, and T. Graf. Applications of CHIP to Industrial and Engineering Problems. In *First International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, Tullahoma, Tennessee, USA, June 1988.

[Fal94] B. Faltings, 'Arc consistency for continuous variables,'*Artificial Intelligence*, 65(2), 1994.

[FCBM89] P. Fouché, A. Charles, J-P. Barthes, and C. Melin. Qualitative physics: a survey. *Revue d'Intelligence Artificielle*, 3, 1989.

[Fre78] E. C. Freuder. Synthesizing constraint expressions. *Communications of the ACM*, 21:958–966, November 1978.

[GH88] H.W. Güsgen and J. Hertzberg. Some fundamental properties of local constraint propagation. *Artificial Intelligence*, 36(2):237–247, September 1988.

[HS94] Hoon Hong and Volker Stahl. Safe starting regions by fixed points and tightening. *Computing*, 53:323–335, 1994.

[Hyv92] E. Hyvőnen. Constraint reasoning based on interval arithmetic ; the tolerance propagation approach. *Artificial Intelligence*, (58):71–112, 1992.

[Jég91] P. Jégou. *Contribution à l'Etude des Problèmes de Satisfaction de Contrainte*. PhD thesis, Université Montpellier II, 1991.

[JM94] J. Jaffar and M. Maher. Constraint Logic Programming: A Survey. *Journal of Logic Programming*, 19/20:503–581, 1994.

[JV88] P. Janssen and M.C. Vilarem. Problèmes de satisfaction de contraintes: techniques de résolution et application à la synthèse de peptide. Technical Report 54, CRIM, 1988.

[Lho93] O. Lhomme. Consistency techniques for numeric CSPs. In *Proc. IJCAI93, Chambery, (France)*, pages 232–238, August 1993.

[Lho94] O. Lhomme. Contribution à la résolution de contraintes sur les réels par propagation d'intervalles. Thèse de doctorat, University of Nice Sophia Antipolis - CNRS, Route des Colles, B.P. 145, 06903 Sophia Antipolis Cedex, FRANCE, 1994.

[LGR96] O. Lhomme and Arnaud Gotlieb and Michel Rueher and Patrick Taillibert Boosting the Interval Narrowing Algorithm In *JICSLP'96, Bonn, Germany, 2-6 September, 1996*

[Mac77] A. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118, 1977.

[Mon74] U. Montanari. Networks of constraints: Fundamental properties and applications to picture processing. *Information Science*, 7(2):95–132, 1974.

[Moo66] R.E. Moore. *Interval Analysis*. Prentice Hall, 1966.

[MR95] P. Marti and M. Rueher. A distributed cooperating constraints solving system. Special issue of *IJAIT (International Journal on Artificial Intelligence Tools)*, 4(1-2):93–113, June 1995.

[OV90] W.J. Older and A. Velino. Extending prolog with constraint arithmetic on real intervals. In *Proc. of IEEE Canadian conference on Electrical and Computer Engineering*. IEEE Computer Society Press, 1990.

[OV93] W. Older and A. Vellino. Constraint arithmetic on real intervals. In F. Benhamou and Alain Colmerauer, editors, *Constraint Logic Programming: Selected Research*. MIT Press, 1993.

[Rue95] M. Rueher. *An Architecture for Cooperating Constraint Solvers on Reals*. In Andreas Podelski, editor, *Constraint Programming: Basics and Trends*, LNCS 910, pages 231–250. Springer Verlag, March 1995.

[Tsa93] E. Tsang. *Foundations of Constraint Satisfaction*. Academic Press, 1993.

[VDT92] P. Van Hentenryck, Yves Deville, and Choh-Man Teng. A generic arc-consistency algorithm and its specializations. *Artificial Intelligence*, 57(2–3):291–321, October 1992.

[vE94] M.H. van Emden. The compatibility operator for real intervals, herbrand universes, and finite domains. Technical report, University of Victoria, Canada, 1994.

[Wal72] D.L. Waltz. Generating semantic descriptions from drawings of scenes with shadows. Tech. rept. ai-tr-271, IT, Cambridge, MA, 1972.