Extension des QuadTrees pour la représentation et le filtrage des contraintes numériques définies par des fonctions par morceaux

Élise Vareilles, Khaled Hadj-Hammou, Michel Aldanondo, Paul Gaborit

Extension des QuadTrees pour la représentation et le filtrage des contraintes numériques définies par des fonctions par morceaux

E. Vareilles [1] **K. Hadj-Hamou** [2] **M. Aldanondo** [1] **P. Gaborit** [1]

[1] Centre GI, Ecole des Mines d'Albi, Campus Jarlard, 81000 Albi
[2] Laboratoire GILCO, INP Grenoble, 46 Avenue Félix Viallet, 38000 Grenoble
(Vareille,Aldanondo,Gaborit)@enstimac.fr, Hamou@gilco.inpg.fr

Résumé

Cette communication présente une approche de représentation de contraintes continues. La méthode de discrétisation dynamique des domaines des variables définit une représentation de l'espace de recherche sous la forme d'un arbre appelé QuadTree dans le cas de contraintes binaires. Nous proposons une extension de cette méthode pour prendre en compte les contraintes continues exprimées comme des fonctions par morceaux. La première section introduit la méthode de génération de QuadTrees. La seconde section propose une définition des contraintes définies par des fonctions par morceaux. Enfin, la troisième section propose une extension de la méthode de QuadTree permettant de prendre en compte ce type de contraintes.

Abstract

This paper proposes an approach allowing representation of continuous constraints. One solution to process these constraints is to approximate the feasible solution regions by a tree decomposition called QuadTree in the case of binary constraints. We propose an extended QuadTree framework dealing with continuous constraints defined with piecewise functions. The first section introduces the QuadTree method. The second one describes the continuous constraints defined with piecewise functions. The third section proposes an extended QuadTrees method allowing to approximate the feasible solution domains of these constraints.

1 Introduction

Un problème de satisfaction de contraintes (CSP) est défini par un ensemble de variables, chaque variable est associée à un domaine représentant l'ensemble des valeurs ou états que peut prendre cette variable, et d'un ensemble de contraintes, chaque contrainte est définie pour un ensemble de variables sur lesquelles elle porte et par un ensemble de combinaisons de valeurs qu'elle autorise [8]. Une solution d'un CSP est une instanciation des variables respectant toutes les contraintes. Le cadre des CSP est rapidement apparu comme un cadre naturel pour représenter beaucoup de problèmes de décision : configuration et conception de produits, ordonnancement de tâches, allocation de ressources, ...

Il existe de nombreuses approches basées sur les contraintes utilisables dans des domaines particuliers. Le choix d'une ou plusieurs approches dépend de deux aspects :

- les domaines sur lesquels portent les variables qui peuvent être :
  1. symboliques : listes de symboles,
  2. numériques discrets : domaines dénombrables (listes d'entiers, listes de réels ou intervalles d'entiers),
  3. numériques continus : domaines infinis non dénombrables (intervalles de réels),
- le type de contrainte pouvant s'exprimer :
  1. en extension : table de compatibilité (liste des combinaisons autorisées ou non autorisées) portant sur des variables symboliques et/ou numériques discrètes,
  2. en intension : expression mathématique portant sur des variables numériques (soit pour éviter de lister toutes les combinaisons ou parce qu'il est impossible de les lister).

Les CSP peuvent être résolus par l'utilisation de procédures de recherche aidées par des méthodes d'in-

férence de type maintien de cohérence par filtrage. La spécificité des méthodes de filtrage est qu'elles opèrent, en cours de résolution, une réduction du domaine de définition initial des variables de manière consistante vis-à-vis du système de contraintes.
Dans le cas de tables de compatibilité, beaucoup d'algorithmes de maintien de cohérence, tel que le filtrage local par Arc-Cohérence AC, ont été développés [3]. L'AC garantit que l'instanciation de l'une des variables mises en jeu par une contrainte possède un support validant cette contrainte dans les domaines des autres variables.

Cependant, le cadre CSP classique ne permet de représenter que des variables à domaines finis et discrets, alors que l'expérience montre que la plupart des problèmes réels sont des problèmes mixtes, regroupant des variables discrètes et des variables continues, où coexistent des contraintes de compatibilité et des contraintes type expressions mathématiques, qui limitent les valeurs possibles de variables numériques continues. C'est ce qui a justifié l'extension du cadre CSP discrets au CSP numériques ou continus [6]. La propagation des contraintes par maintien de cohérence se fait alors sur les intervalles. Les méthodes les plus utilisées pour les domaines continus sont un mariage entre l'arithmétique des intervalles [9] et les méthodes de filtrage local [2] [5] [7]. Plusieurs techniques ont été mises au point comme la B-consistance [6] et la Box-consistance [1]. Pour un panorama plus détaillé des techniques de maintien de cohérence sur des contraintes continues, voir la thèse de Delobel [4].

La 2B-consistance est une consistance d'arc sur les bornes des intervalles de variables mises en jeu par une contrainte de type expression mathématique $f(x_1, x_2, ..., x_n) =, <, \leq, >, \geq\ 0$. Elle garantit que les bornes du domaine de l'une des variables possèdent un support validant cette contrainte dans les domaines des autres variables. Cet algorithme de filtrage local exploite la forme d'expression des contraintes et présente quelques inconvénients :

- dans le cas où plusieurs contraintes portent sur un même sous-ensemble de variables (exemple de la recherche de l'intersection de deux fonctions). Ce cas génère dans le réseau de contraintes des cycles supplémentaires qui compliquent le processus de propagation,
- lorsqu'il est difficile ou impossible de trouver une fonction de projection pour chacune des variables d'une contrainte,
- lorsque les fonctions de projection ne sont pas continues et/ou monotones,
- le résultat de filtrage peut être différent suivant la forme syntaxique d'une contrainte, par exemple la contrainte : $x.y - x = 0$ et $x(y-1) = 0$,
- l'évaluation des fonctions de projection peut détruire la continuité des domaines.

Pour pallier à certains de ces inconvénients et pour obtenir une représentation puissante des domaines de faisabilité, Sam-Haroud [10] propose d'approximer les contraintes numériques en utilisant une décomposition hiérarchique et dynamique sous la forme d'arbres 2^k (2^k-Trees). Cette représentation, fréquemment utilisée dans le domaine du traitement d'image, est appelée QuadTree pour des contraintes binaires (plan de faisabilité défini par deux variables) et OcTree pour des contraintes ternaires (espace de faisabilité).

Dans la première section, nous exposerons en détail la méthode de génération des contraintes numériques binaires par les QuadTrees. Ensuite, nous présenterons le mécanisme de fusion de QuadTrees pour la prise en compte de plusieurs contraintes numériques.
Si cette méthode de représentation en QuadTrees permet de prendre en compte plusieurs formes d'expressions mathématiques, un de ses inconvénients majeurs est qu'elle ne permet pas de traiter les contraintes numériques définies par morceaux souvent introduites pour approximer des abaques et des résultats expérimentaux dans le cadre d'un problème de conception [11]. Une contrainte de type fonction par morceaux sur un espace de recherche multi-intervalles est définie par un ensemble d'expressions mathématiques couvrant chacune une zone bien déterminée de cet espace.
Dans la seconde section, nous définirons les contraintes numériques définies comme des fonctions par morceaux et dresserons quelques hypothèses concernant la forme de ces contraintes ainsi que les limites d'application de la méthode des QuadTrees.
Enfin, l'objectif de la troisième section est de proposer une extension de la méthode de génération des QuadTrees permettant de prendre en compte les contraintes numériques s'exprimant par des fonctions par morceaux. Pour illustrer notre contribution, nous nous concentrerons uniquement sur les contraintes binaires mettant en œuvre deux variables.

2 Approximation des contraintes numériques par des QuadTrees

Contrairement aux techniques classiques de consistance exploitant les contraintes numériques représentées formellement par des expressions mathématiques, les méthodes de discrétisation consistent à approximer les domaines de faisabilité de ces contraintes. Pour obtenir une approximation puissante de ces domaines, nous supposerons que chaque variable prend ses valeurs dans un domaine continu borné et qu'il lui est associé un niveau de précision.

Nous définirons dans un premier temps les QuadTrees. Ensuite, nous illustrerons ce principe de décomposition sur un exemple. Enfin, nous montrerons comment fusionner plusieurs QuadTrees et aborderons brièvement un algorithme de filtrage.

2.1 Définition des QuadTrees

Une contrainte $C(x, y)$, définie par une expression mathématique entre deux variables continues x et y, peut être approximée par un QuadTree en effectuant une décomposition binaire et hiérarchique de ses domaines initiaux de faisabilité Dx et Dy.
Le principe des QuadTrees est qu'un espace de recherche défini par un nœud père est décomposé, suivant les deux variables x et y, en quatre sous-espaces définis par quatre nœuds fils : NO (Nord-Ouest), SO (Sud-Ouest), SE (Sud-Est) et NE (Nord-Est). Chaque nœud du QuadTree correspond à une zone définie par un couple d'intervalles (δx,δy). Notons que dans le cas de contraintes ternaires, un nœud père est décomposé, suivant les trois variables x, y et z, en huit nœuds pour générer un OcTree.
Une couleur est associée à chaque nœud. La couleur est définie suivant l'intersection entre le rectangle défini par la région (δx,δy) et la contrainte continue numérique $C(x, y)$, tel que :

- *Blanc* : si tous les points de la région (δx,δy) sont consistants avec la contrainte $C(x, y)$,
- *Gris* : si la région (δx,δy) comprend à la fois des points consistants et inconsistants avec la contrainte $C(x, y)$,
- *Noir* : si tous les points de la région (δx,δy) sont inconsistants avec la contrainte $C(x, y)$,

Chaque nœud de couleur *Gris* est à son tour décomposé en quatre nœuds fils. Les nœuds de couleur *Blanc* (consistants) ou *Noir* (inconsistants) ne sont pas décomposés. Le mécanisme de décomposition des nœuds *Gris* est répété jusqu'à ce que le niveau de précision des variables soit atteint. A ce dernier niveau de l'arbre, les nœuds unitaires (feuilles de l'arbre) *Gris* deviennent soit *Blancs* s'il l'on interdit de perdre des points consistants, soit *Noirs* s'il l'on interdit de garder des points inconsistants. Le QuadTree final contient uniquement des feuilles *Blanches* et des feuilles *Noires*.

2.2 Exemple de génération de QuadTree

Nous présentons un exemple de génération de QuadTree pour une contrainte numérique continue.
Soit la contrainte binaire suivante (Figure 1) :

$C_1(x, y) : y \geq 1 + (x - 1)^2$

Avec les domaines initiaux des deux variables x et y :

$Dx = [0, 4]$ et $Dy = [0, 10]$

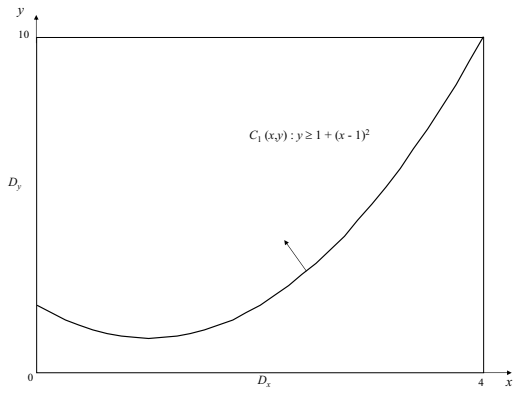

FIG. 1 – La contrainte $C_1(x, y) : y \geq 1 + (x - 1)^2$

La Figure 2 montre les étapes de construction du QuadTree de la contrainte $C_1(x, y)$ avec un niveau de précision de $Dx/16$ et $Dy/16$ pour respectivement x et y et la Figure 3 montre le QuadTree.

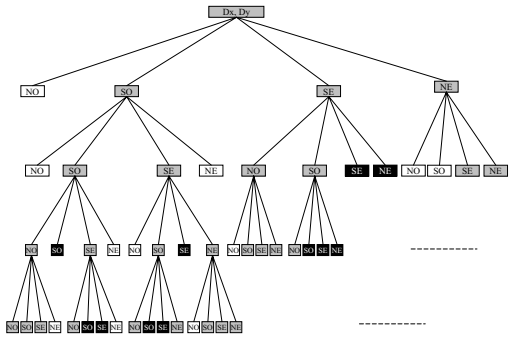

FIG. 2 – QuadTree de la contrainte $C_1(x, y)$

2.3 Fusion de QuadTrees et Filtrage

Lorsqu'il y a deux contraintes continues entre deux variables x et y, le QuadTree de la contrainte totale entre ces deux variables est construit par l'intersection des deux QuadTrees initiaux correspondant aux deux contraintes. Ce mécanisme de fusion n'est possible que si les domaines initiaux des deux variables sont les mêmes pour les deux contraintes. Etant donné un ordre des couleurs : *Blanc* < *Gris* < *Noir*, la couleur de chaque nœud de l'arbre résultant de cette fusion (*Nœud-Fusion*) est définie par la comparaison des couleurs des nœuds (*Nœud-1* et *Nœud-2*) de même niveau correspondant aux deux arbres des deux contraintes suivant la règle suivante :

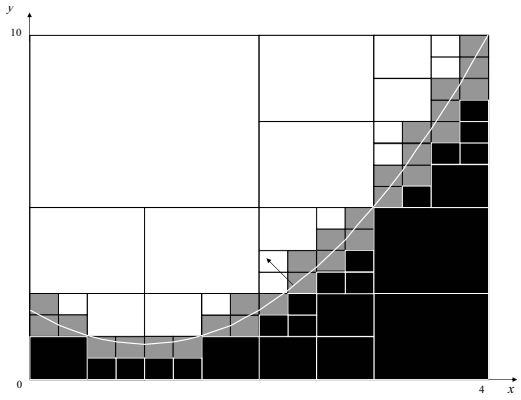

FIG. 3 – QuadTree de la contrainte $C_1(x, y)$

Couleur (*Nœud-Fusion*) =
Max [Couleur (*Nœud-1*) , Couleur (*Nœud-2*)].
Le filtrage des QuadTrees consiste à supprimer les branches de l'arbre devenues totalement inconsistantes avec le problème courant.
La Figure 5 montre le QuadTree résultant de la fusion des QuadTrees de deux contraintes C_1 et C_2.
$C_1(x, y) : y \geq 1 + (x - 1)^2$ (QuadTree : Figure 3)
$C_2(x, y) : y \leq 5 + ln(1/10 + x)$ (QuadTree : Figure 4)

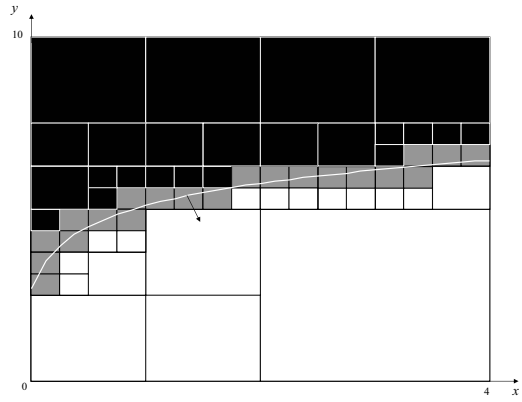

FIG. 4 – QuadTree de la contrainte $C_2(x, y)$

Avec les domaines initiaux des deux variables x et y : $Dx = [0, 4]$ et $Dy = [0, 10]$.

Pour cet exemple, si on choisit d'interdire de garder des points inconsistants, alors les feuilles de couleur *Gris* du QuadTree final deviennent *Noir*. Si les quatre nœuds unitaires fils d'un nœud père sont de même couleur alors ce nœud père devient un nœud unitaire de même couleur en absorbant ses quatre fils. Le QuadTree final après fusion est représenté par la Figure 6.

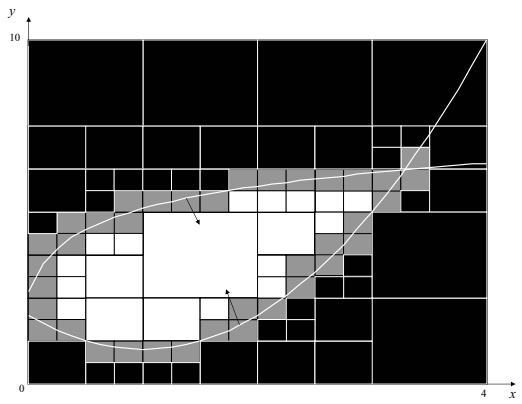

FIG. 5 – Fusion des QuadTrees de C_1 et de C_2

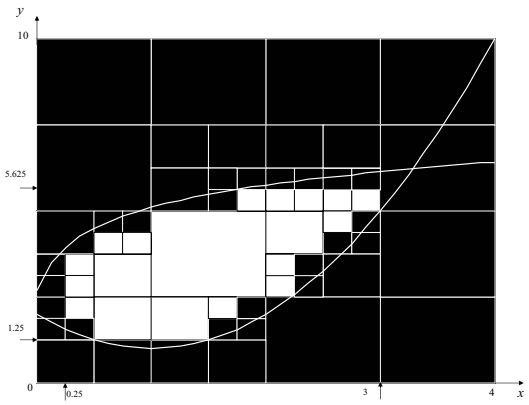

FIG. 6 – QuadTree final après transformation des nœuds *Gris*

Pour le filtrage des domaines par l'utilisation des QuadTrees, Sam-Haroud [10] a proposé une variante de l'algorithme AC-3 nommé τ-AC-3. Le principe de cet algorithme consiste à retirer des domaines des variables les valeurs qui ne seront jamais solutions. Ceci est effectué par la suppression des valeurs des variables correspondant aux nœuds de couleur *Noir* (inconsistants). Ceci est équivalent à calculer l'intersection entre d'une part l'union des intervalles des nœuds consistants du QuadTree (*Blancs*) et d'autre part les domaines courants des variables (Dx et Dy).
Pour plus de détails sur la construction des QuadTrees et des méthodes de filtrage, plusieurs exemples sont donnés dans la thèse de Sam-Haroud [10].
Pour l'exemple précédent, l'application de l'algorithme τ-AC-3 réduit les variables x et y comme suit :
$Dx = [0, 4] \Rightarrow Dx = [0.25, 3]$
$Dy = [0, 10] \Rightarrow Dy = [1.25, 5.625]$

3 Génération de QuadTrees pour des contraintes définies par morceaux

Dans cette section, nous proposons une méthode de génération de QuadTrees pour des contraintes continues binaires exprimées sous forme de fonctions définies par morceaux. Dans la première sous-section, nous définirons les contraintes continues définies par morceaux. Ensuite, en se basant sur la méthode de Sam-Haroud nous développerons deux méthodes de génération et de fusion de ce type de QuadTrees : la première pour les contraintes de type égalités et la seconde pour les inégalités. Notons que dans cette communication, nous n'aborderons pas les algorithmes de filtrage.

3.1 Contraintes continues définies par morceaux

Une contrainte continue binaire $C(x, y)$ est définie par morceaux sur deux intervalles Dx et Dy, s'il existe une collection de plusieurs expressions mathématiques $C^1(x, y)$, $C^2(x, y)$, ..., $C^n(x, y)$ telles que $C^i(x, y)$ est continue sur une région bien déterminée définie sur un sous-intervalle $D^i x$ de la variable x et un sous-intervalle $D^i y$ de la variable y et $C(x, y)$ et $C^i(x, y)$ égales sur la région $(D^i x,\ D^i y)$. Les expressions mathématiques $C^i(x, y)$ peuvent être soient des égalités ou des inégalités. Par exemple, la contrainte binaire définissant l'intérieur d'une forme géométrique d'un losange sur les domaines $Dx = [0, 16]$, $Dy = [0, 16]$ est donnée par la Figure 11 :

$C^1(x, y) : y \leq -0.8x + 18.5$
$\quad D^1x = [7.5, 12.5],\ D^1y = [8.5, 12.5]$
$C^2(x, y) : y \geq 1.25x - 7.125$
$\quad D^2x = [8.5, 13.5],\ D^2y = [3.5, 8.5]$
$C^3(x, y) : y \geq -0.8x + 10.3$
$\quad D^3x = [3.5, 8.5],\ D^3y = [3.5, 7.5]$
$C^4(x, y) : y \leq 1.25x - 3.125$
$\quad D^4x = [3.5, 7.5],\ D^4y = [7.5, 12.5]$

Dans notre proposition, nous distinguons le traitement des contraintes continues par des morceaux exprimés sous la forme d'égalités et ceux exprimés comme des inégalités.
Dans le cas d'inégalités, la contrainte doit avoir absolument une forme géométrique fermée ce qui permet de définir une frontière entre les zones consistantes (intérieur ou extérieur de la forme) et les zones inconsistantes (extérieur ou intérieur de la forme). Nous posons également l'hypothèse de cohérence des morceaux de la contrainte. Une des situations qui peut représenter une contrainte continue incohérente est celle où des morceaux de la même contrainte d'inégalités se croisent. Ce qui génère des points à la fois consistants et inconsistants vis-à-vis d'une même contrainte.

3.2 Niveaux d'information sur les nœuds

Notons que dans le cas d'une contrainte continue classique définie par une seule expression mathématique, la contrainte est définie sur l'ensemble des domaines de définition Dx et Dy des variables x et y. Ce qui signifie que l'on dispose de toute l'information concernant la consistance ou l'inconsistance de chaque nœud du QuadTree. Dans le cas de contraintes définies par morceaux, seules les zones $(D^i x,\ D^i y)$ couvertes par chacun des morceaux $C^i(x, y)$ de la contrainte (domaines de définition des morceaux) possèdent de l'information sur les nœuds consistants et inconsistants vis-à-vis de la contrainte $C(x, y)$.
Notre méthode de génération des QuadTrees associés aux contraintes continues par morceaux est basée sur le niveau d'information de chaque nœud du QuadTree pour déduire sa consistance ou son inconsistance.
En plus des deux états possédant une information totale et qui correspondent aux deux états Cohérent (*Blanc*) et Incohérent (*Noir*), nous définissons quatre autres niveaux d'information que peut avoir un nœud et sont illustrés par la Figure 7 :

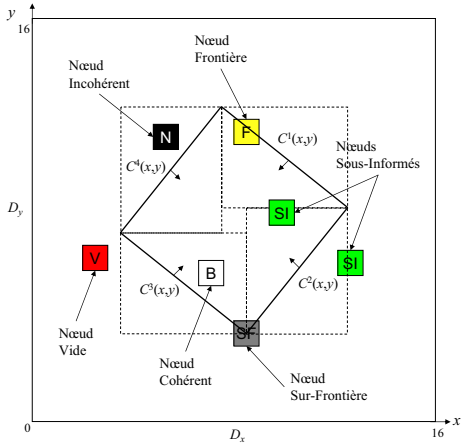

FIG. 7 – Typologie des nœuds suivant le niveau d'information

- nœud ne possédant aucune information (nœud *vide*) : un nœud de domaine (δx,δy) est dit *vide* si l'intersection de son domaine avec tous les domaines de définition des morceaux $C^i(x, y)$ ($\forall i = 1 \ldots n$) de la contrainte est vide. Ce nœud devient alors unitaire.
- nœud ne possédant pas assez d'information (nœud *sous-informé*) : un nœud de domaine (δx,δy) est dit *sous-informé* si son domaine est intersecté par au moins le domaine d'un morceau $C^i(x, y)$ de la contrainte et qu'il n'est traversé par aucun morceau. Ce nœud devient alors unitaire.

– nœud possédant suffisamment d'information (nœud *frontière*) : un nœud de domaine (δx,δy) est dit *frontière* si un et un seul morceau de la contrainte le traverse. Le nœud *frontière* est équivalent au nœud *Gris* de la méthode de Sam-Haroud.
– nœud possédant plusieurs informations (nœud *sur-frontière*) : un nœud de domaine (δx,δy) est dit *sur-frontière* si son domaine est traversé par au moins deux morceaux de la contrainte. Ce type de nœud est décomposé jusqu'au niveau unitaire. Il peut alors donner naissance à des nœuds pouvant être *sur-frontière*, *frontière*, *sous-informé* et/ou *vide*.

Un nœud unitaire possède une information pertinente pour juger son état (consistant ou inconsistant) lorsqu'il est intersecté par un et un seul morceau de la contrainte. Pour cela, il est nécessaire de définir une taille maximale des nœuds unitaires (niveaux de précision de x et y pour les feuilles de l'arbre QuadTree) à partir de laquelle il est possible d'isoler chaque morceau de la contrainte et appliquer ainsi la méthode de génération classique de QuadTree.

3.3 Génération de QuadTrees pour des contraintes d'égalités par morceaux

Les solutions cohérentes pour une contrainte de type égalités par morceaux sont représentées par l'enveloppe des morceaux de la contrainte. Dans ce cas, seuls les nœuds unitaires *frontières* et *sur-frontières* sont cohérents avec la contrainte. La génération du QuadTree pour ce type de contraintes est alors basée sur l'identification de ces nœuds unitaires *frontières* et *sur-frontières* qui vont permettre de statuer ensuite sur l'état des autres nœuds.

1. Lorsqu'un nœud *frontière* est détecté, il peut recevoir une des couleurs suivantes :
   – *Gris* : s'il n'est pas unitaire et doit donc être décomposé. Le sous-QuadTree associé au morceau isolé est ensuite généré avec la même procédure proposée par Sam-Haroud générant ainsi des zones consistantes (*Blanc*) et des zones inconsistantes (*Noir*).
   – *Blanc* : s'il est unitaire. Ce nœud devient *Blanc* car il appartient à l'enveloppe du morceau de la contrainte, donc consistant avec toute la contrainte.
2. Lorsqu'un nœud *sur-frontière* est détecté, il peut recevoir une des couleurs suivantes :
   – *Gris* : s'il n'est pas unitaire et doit donc être décomposé générant ainsi des nœuds fils pouvant être *sur-frontières*, *frontières*, *sous-informés* et/ou *vides*.
   – *Blanc* : s'il est unitaire. Ce nœud devient *Blanc* car il appartient à l'enveloppe du morceau de la contrainte, donc consistant avec toute la contrainte.
3. Les nœuds *vides* et *sous-informés*, qui sont toujours unitaires, se colorent en *Noir* car il ne définissent pas l'enveloppe de la contrainte.

A la fin de cette procédure, tous les nœuds unitaires du QuadTree sont soit *Blancs* (consistants) ou *Noirs* (inconsistants). Pour illustrer cette procédure, nous reprenons dans la Figure 8 l'exemple précédent du losange en considérant tous les morceaux de la contrainte comme des égalités.

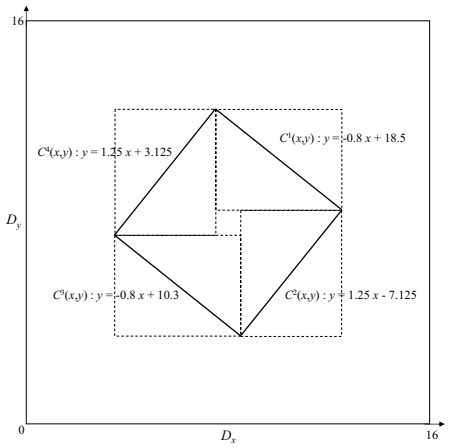

FIG. 8 – Exemple de contrainte égalités par morceaux

Le QuadTree généré par notre approche est illustré par la Figure 9.
Le QuadTree final après transformation de tous les nœuds unitaires (en *Blancs* ou en *Noirs*) est représenté par la Figure 10.

Lorsqu'il y a plusieurs contraintes binaires, le QuadTree représentant la contrainte totale est construit par l'intersection des QuadTrees finaux (ayant uniquement des nœuds unitaires *Blancs* et *Noirs*) en utilisant le même mécanisme de fusion proposé par Sam-Haroud.

3.4 Génération de QuadTrees pour des contraintes d'inégalités par morceaux

Les solutions cohérentes pour une contrainte binaire de type inégalités par morceaux définissent des surfaces. Ces surfaces fermées représentent soit l'intérieur de la forme géométrique (exemple de la Figure 11) ou son extérieur limité par les domaines de définition des variables (exemple de la Figure 12).

Les états des nœuds du QuadTree possédant de l'information (consistants ou inconsistants) sont propagés

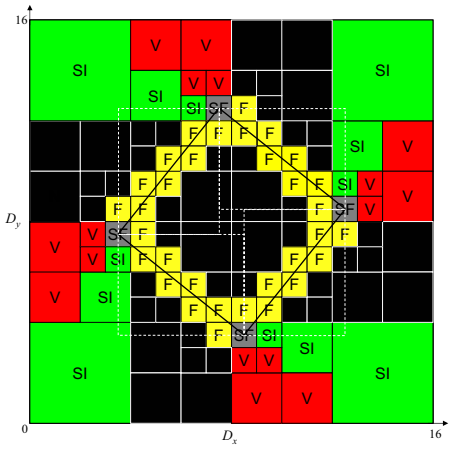

FIG. 9 – Génération du QuadTree de la contrainte égalités par morceaux

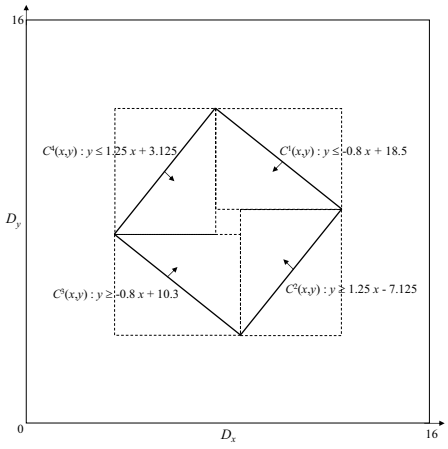

FIG. 11 – Contrainte d'inégalités par morceaux : cas intérieur cohérent

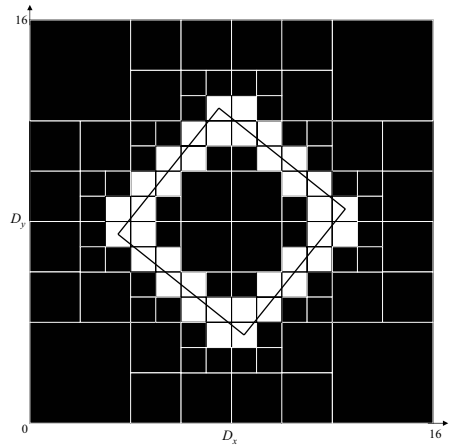

FIG. 10 – QuadTree final de la contrainte égalités par morceaux

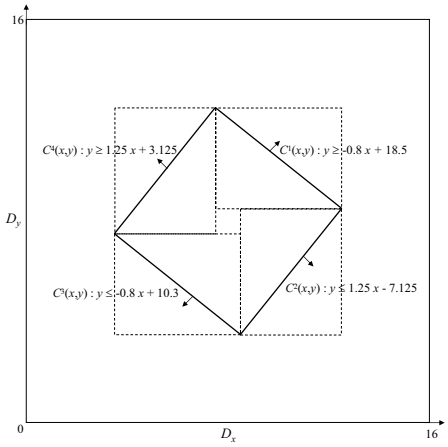

FIG. 12 – Contrainte d'inégalités par morceaux : cas extérieur cohérent

par une procédure de voisinage pour renseigner l'état des autres nœuds. Cette procédure de propagation par voisinage est déclenchée après avoir généré le QuadTree et que chaque nœud possède un des états précédemment définis : *Blanc*, *Noir*, *Vide*, *Sous-informé*, *Frontière* ou *Sur-frontière*. Si la contrainte est cohérente (pas de morceaux croisés) le QuadTree doit posséder au moins un nœud unitaire *Blanc* et/ou un nœud unitaire *Noir* en définissant une précision maximale des variables. Ceci représente une condition nécessaire pour appliquer la procédure de propagation d'information par voisinage. Nous définissons les voisins d'un nœud comme les nœuds unitaires qui lui sont adjacents. Les règles d'affectation des niveaux d'information et des couleurs (*Noir* ou *Blanc*) restent les mêmes que pour le cas d'égalités par morceaux. Ce qui change dans le cas d'inégalités, c'est l'utilisation de la procédure de propagation des états. Nous proposons plusieurs règles de propagation d'information, fonction du niveau d'information initial ou de la couleur initiale des nœuds qui informent leurs voisins. La propagation se fait uniquement à partir des nœuds unitaires *Frontières*, *Blancs* ou *Noirs*.

Pour illustrer cette procédure de propagation par voisinage, nous utilisons l'exemple précédent du losange pour les deux cas : intérieur cohérent (Figure 11) et extérieur cohérent (Figure 12). Les QuadTrees finaux avant propagation sont illustrés par la Figure 13 pour le cas intérieur cohérent et la Figure 14 pour le cas extérieur cohérent.

Les règles de propagation des états par voisinage sont les suivantes :

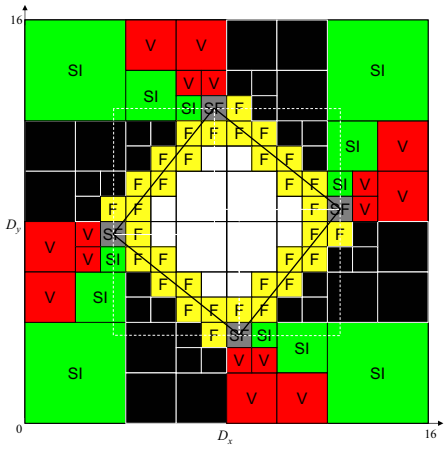

FIG. 13 – QuadTree final avant propagation : cas intérieur cohérent

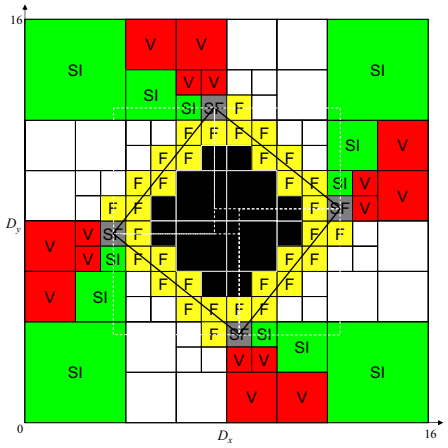

FIG. 14 – QuadTree final avant propagation : cas extérieur cohérent

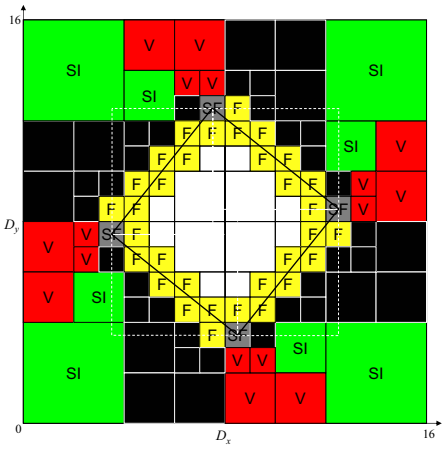

FIG. 15 – QuadTree après propagation des nœuds unitaires *Frontières* : cas de l'intérieur cohérent

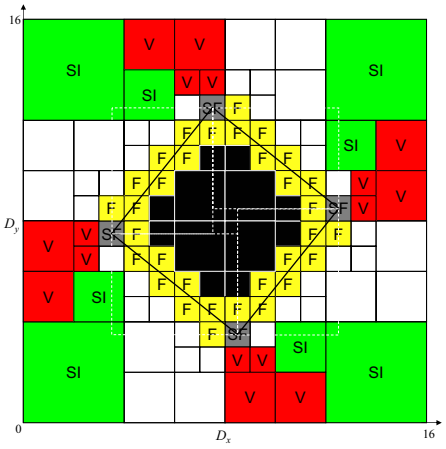

FIG. 16 – QuadTree après propagation des nœuds unitaires *Frontières* : cas de l'extérieur cohérent

- Propagation des nœuds unitaires *Frontières* : les domaines de ces nœuds sont traversés par un et un seul morceau. Ils peuvent alors indiquer à leurs voisins nœuds unitaires *sous-informés* et nœuds unitaires *vides* s'ils se trouvent du bon ou du mauvais côté de la frontière définie par le morceau. Ceci est réalisé en vérifiant si les domaines de définition de chaque nœud voisin vérifient le morceau de la contrainte. Si tel est le cas, le nœud unitaire (*sous-informé* ou *vide*) est cohérent et devient *Blanc* comme illustré par le passage de la Figure 14 à la Figure 16 dans le cas de l'extérieur cohérent, et s'il est incohérent il devient *Noir* comme illustré par le passage de la Figure 13 à la Figure 15 dans le cas de l'intérieur cohérent.
- Propagation des nœuds unitaires *Noirs* : ces nœuds unitaires sont inconsistants avec la contrainte. Ils propagent alors cette inconsistance à leurs nœuds unitaires voisins *sous-informés* et *vides* qui deviennent alors *Noirs*. Cette règle est illustrée par le passage de la Figure 15 à la Figure 17 dans le cas de l'intérieur cohérent.
- Propagation des nœuds unitaires *Blancs* : les nœuds unitaires *Blancs* sont des nœuds consistants. Ils propagent alors cette consistance à leurs nœuds unitaires voisins *sous-informés* et *vides* n'ayant aucun voisin *Noir*. En effet, considérant que le *Noir* domine le *Blanc*, si un nœud unitaire *sous-informé* ou *vide* a au moins un nœud unitaire voisin *Noir*, alors il devient aussi *Noir*. La règle de propagation des nœuds unitaires *Blancs* est illustrée par le passage de la Figure 16 à la Figure 18 dans le cas de l'extérieur cohérent.

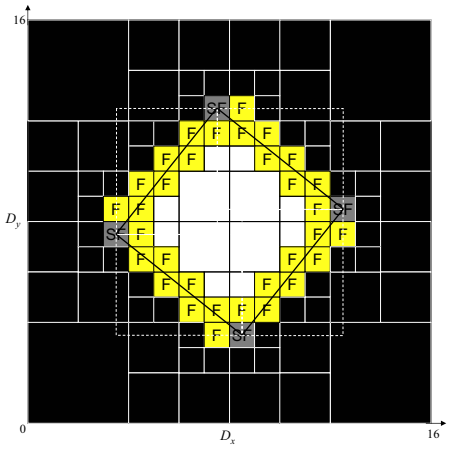

FIG. 17 – QuadTree après propagation des nœuds unitaires *Noirs* et *Blancs* : cas de l'intéieur cohérent

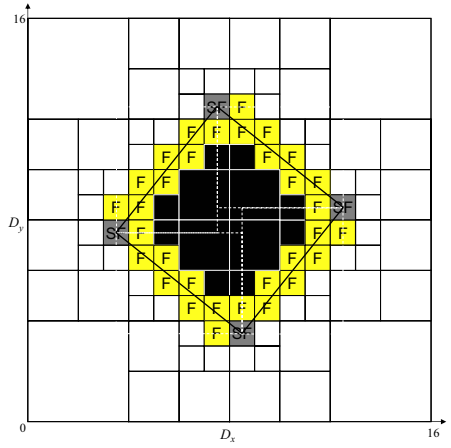

FIG. 18 – QuadTree après propagation des nœuds unitaires *Blancs* et *Noirs* : cas de l'extérieur cohérent

– règles de coloration des nœuds unitaires *Frontières* et *Sur-frontières* : la règle de coloration des nœuds unitaires *Frontières* ou *Sur-frontières* est équivalente à la règle de coloration des nœuds unitaires *Gris* des QuadTrees proposés par Sam-Haroud. A ce dernier niveau de l'arbre, les nœuds unitaires *Frontières* ou *Sur-frontières* deviennent soit *Blancs* s'il l'on interdit de perdre des points consistants, ou *Noirs* s'il l'on interdit de garder des points inconsistants. Cette règle est illustrée par le passage de la Figure 17 à la Figure 19 dans le cas de l'intérieur cohérent et par le passage de la Figure 18 à la Figure 20 dans le cas de l'extérieur cohérent.

A la fin de l'application de ces règles de propagation des états des nœuds par voisinage, le QuadTree final contient uniquement des nœuds unitaires *Blancs* (consistants) et des nœuds unitaires *Noirs* (inconsistants). Ensuite, si les quatre nœuds unitaires fils d'un nœud père sont de même couleur (*Blancs* ou *Noirs*) alors ce nœud père devient un nœud unitaire de même couleur (*Blanc* ou *Noir*) en absorbant ses quatre fils. Les QuadTrees finaux pour les exemples précédents sont illustrés respectivement par la Figure 21 pour le cas de l'intérieur cohérent du losange et par la Figure 22 pour le cas de l'extérieur cohérent du losange.

Lorsqu'il y a plusieurs contraintes continues binaires, le QuadTree représentant la contrainte totale est construit par l'intersection des QuadTrees finaux en utilisant le même mécanisme de fusion proposé par Sam-Haroud.

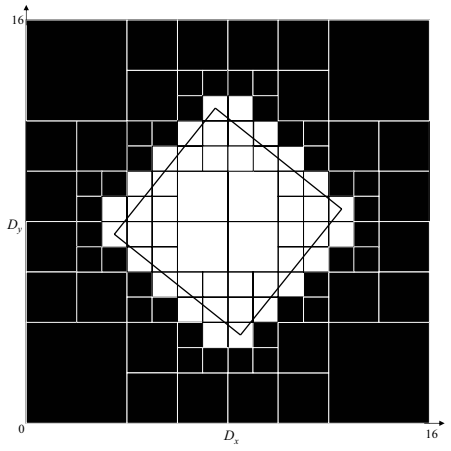

FIG. 19 – QuadTree après coloration des nœuds unitaires *Frontières* et *sur-frontières* : cas de l'intérieur cohérent

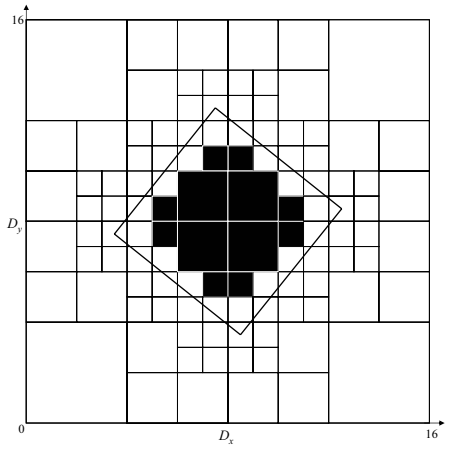

FIG. 20 – QuadTree après coloration des nœuds unitaires *Frontières* et *sur-frontières* : cas de l'extérieur cohérent

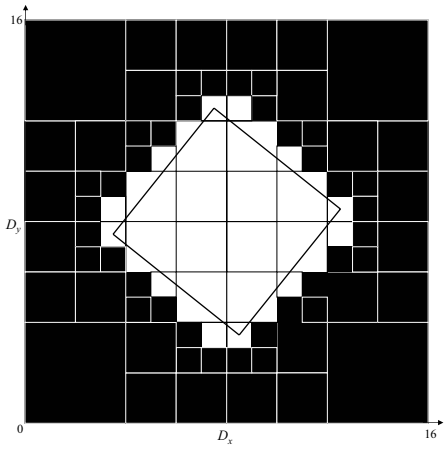

FIG. 21 – QuadTree final : cas de l'intérieur cohérent

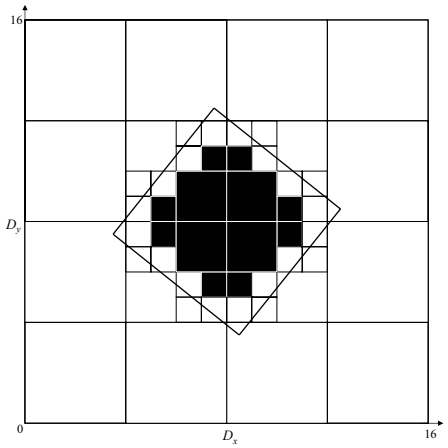

FIG. 22 – QuadTree final : cas de l'extérieur cohérent

4 Conclusion

L'objectif de cette communication était de présenter une extension de la méthode des QuadTrees permettant de traiter des contraintes continues exprimées sous la forme de fonctions définies par morceaux.

Les méthodes de filtrage basées sur les QuadTrees proposées par Haroud [10] permettent de prendre en compte plusieurs types de contraintes continues. Un des inconvénients de cette méthode est qu'elle ne permet pas de traiter les contraintes continues s'exprimant par des fonctions par morceaux. Nous avons introduit une typologie de ce type de contraintes et avons posé quelques hypothèses sur leurs formes.

Pour prendre en compte ce type de contraintes et en plus des deux états des nœuds de l'arbre (*Blanc* et *Noir*), nous avons proposé d'introduire quatre autres états pour les nœuds et qui correspondent aux niveaux d'informations qu'ils contiennent : les nœuds *vides* d'information, les nœuds *sous-informés*, les nœuds *frontières* et les nœuds *sur-frontières*. Ayant identifié des différences entre les contraintes définies par des égalités et des contraintes définies par des inégalités, nous avons proposé une méthode de propagation par voisinage des zones déjà identifiées comme *Blanc* ou *Noir* sur les autres nœuds.

Pour prendre en compte plusieurs contraintes définies ou non par morceaux et qui portent sur les mêmes variables, nous utilisons un mécanisme de fusion des QuadTrees proposé par Sam-Haroud [10]. Nous travaillons actuellement sur une méthode de fusion progressive des QuadTrees permettant de couper au plus tôt des branches inconsistantes des QuadTrees avant d'atteindre les pas de discrétisation. Cette méthode doit s'appuyer sur une extension de la définition de la relation d'ordre entre les différents états des nœuds.

Références

[1] F. Benhamou, D. McAllester and P. van Hentenryck. CLP(intervals) revisited. in *Proceedings of the Int. Logic Programming Symposium*, 1994

[2] E. Davis. Constraint propagation with interval labels. in *Artificial Intelligence*, 32, 1987

[3] R. Debruyne and C. Bessière. Domain Filtering Consistencies. *Journal of Artificial Intelligence Research*, 14 :205–230, 2001

[4] F. Delobel. Résolution de contraintes réelles non linéaires. *Thèse de doctorat, Université de Nice - Sophia Antipolis*, France, 2000

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

[6] O. Lhomme. Consistency techniques for numeric CSPs. in *Proceedings of the 13th International Joint Conference on AI*, 1997

[7] O. Lhomme and M. Rueher. Application des techniques CSP au raisonnement sur les intervalles. in *Revue d'IA*, 11(3) :283–311, 1997

[8] U. Montanari. Networks of constraints : fundamental properties and applications to picture processing. Information sciences, 7 :95-132, 1974. in *Inform. Scie*, 7 :95–132, 1974

[9] R.E. Moore. *Interval Analysis*. Prentice Hall, Englewood Cliffs, NJ, 1966

[10] D. Sam-Haroud. Constraint Consistency Techniques for Continuous Domains. *PhD Thesis, EPFL Lausanne*, Suisse, 1995

[11] E. Vareilles, M. Aldanondo, K. Hadj-Hamou and P. Gaborit. Application des CSP pour la config. d'un traitement thermique. *JFPLC*, Angers, 2004