Une Stratégie de Résolution Orientée par la Topologie des CSPs Numériques

Heikel Batnini, Michel Rueher, Claude Michel

Une Stratégie de Résolution Orientée par la Topologie des CSPs Numériques

Heikel Batnini **Michel Rueher** **Claude Michel**

Université de Nice Sophia-Antipolis
Projet COPRIN I3S-CNRS/INRIA/CERMICS
INRIA, 2004 Route des Lucioles
BP 93, 06902 Sophia-Antipolis
hbatnini@sophia.inria.fr rueher@essi.fr cpjm@essi.fr

Abstract

Les méthodes classiques de résolution de CSPs numériques sont basées sur un algorithme combinant une technique de bissection et un filtrage par consistance locale. En général, le filtrage est basé sur la Hull-consistance ou la Box-consistance. Les algorithmes de filtrage correspondants identifient souvent des *trous* dans les domaines, c'est-à-dire des intervalles sur lesquels certaines contraintes ne sont pas satisfaites. Ces *trous* sont cependant utilisés uniquement pour déterminer le plus petit intervalle englobant les solutions.

Ce papier présente une stratégie de recherche, nommée `TopSearch` (Topological Search), qui exploite les *trous* identifiés par ces filtrages. `TopSearch` utilise ces informations pour sélectionner la direction de coupe ainsi que pour définir le point de coupe dans le domaine sélectionné. Si aucun trou n'est identifié une bissection standard est mise en oeuvre. Les premiers résultats expérimentaux montrent que cette heuristique de recherche dont le sur-coût est minime, peut améliorer de manière très significative les performances de la bissection classique.

1 Introduction

Cet article présente une nouvelle stratégie de recherche pour la résolution de CSPs (Constraint Satisfaction Problem) numériques. Ce problème apparaît dans de nombreux domaines d'application comme en robotique, chimie ou encore en géométrie. L'objectif est de trouver une approximation fine des solutions de systèmes de contraintes non-linéaires.

Les méthodes classiques pour résoudre les CSPs numériques sont basées sur un algorithme combinant une technique de bissection et un filtrage par consistance locale.

La bissection consiste à choisir un intervalle en utilisant diverses stratégies de sélection. Cet intervalle est alors coupé en plusieurs partie, en général en son milieu, puis les sous-problèmes correspondants sont résolus de manière indépendante.

Parmi les stratégies de sélection de domaine, la méthode considérée comme étant la plus efficace en moyenne est le "Round Robin" (`RR`), qui traite les domaines des variables les uns après les autres de manière circulaire. D'autres stratégies de sélection utilisent des informations syntaxiques ou sémantiques. La stratégie Largest First (`LF`), aussi appelée *geometric splitting* [17], consiste à choisir la variable dont le domaine est le plus grand. La stratégie Maximal Smear (`MS`) [6, 13] sélectionne le domaine qui maximise la *smear function* [1]. De manière informelle, `MS` identifie la variable dont la projection a la plus grande pente.

[1] La *smear function* de x_k est :

$$s_k = \max_{1 \le j \le m} \{\max\{|\underline{J}_{i,j}|, |\overline{J}_{i,j}|\} w(\mathbf{x}_i)\},$$

où $\mathbf{J}_{i,j} = [\underline{J}_{i,j}, \overline{J}_{i,j}]$ est la (i,j)-ème entrée de l'extension aux intervalles de la matrice Jacobienne du système

En général, les techniques de filtrage utilisées sont basées sur la Hull-consistance [15, 2] ou la Box-consistance [3, 19]. Les algorithmes de filtrage correspondants identifient souvent des *trous* dans les domaines, c'est-à-dire des intervalles sur lesquels certaines contraintes ne sont pas satisfaites. Ces *trous* sont cependant utilisés uniquement pour déterminer le plus petit intervalle englobant les solutions.

Ce papier présente une stratégie de recherche, nommée `TopSearch` (Topological Search), qui exploite les *trous* identifiés par ces filtrages. `TopSearch` utilise ces informations pour sélectionner la direction de coupe ainsi que pour définir le point de coupe dans le domaine sélectionné. Si aucun trou n'est identifié une bissection standard est mise en oeuvre.

Plus précisément, les trous dans les domaines sont identifiés par l'algorithme HC4 qui permet d'implanter efficacement à la fois la Hull-consistance et la Box-consistance [2]. `TopSearch` récupère ces informations avec un sur-coût négligeable puis les exploite durant la phase de recherche. Plusieurs heuristiques exploitant ces informations peuvent être utilisées, par exemple des heuristiques basées sur la taille des trous identifiés.

Un retour-arrière chronologique est la plupart du temps utilisé pour traiter les sous-CSPs générés par la bissection. D'autres stratégies plus sophistiquées peuvent cependant être utilisées, par exemple une stratégie de retour-arrière dynamique telle que celle qui a été présentée par [11]. `TopSearch` est une stratégie de recherche pouvant être combinée avec n'importe quelle stratégie de retour-arrière.

D'autre part, des consistance plus fortes basées sur l'utilisation des trous identifiés ont été proposées par [8, 4]. L'originalité de `TopSearch` provient du fait que les trous sont exploités uniquement pour la recherche de solutions.

Ce papier est organisé de la manière suivante : la section 2 rappelle les notions de base sur l'arithmétique d'intervalles. Le schéma général de `TopSearch` est décrit dans la section 3. La section 4 présente la technique utilisée pour collecter les trous dans les domaines. Enfin, la section 5 présente quelques résultats expérimentaux sur des problèmes issus de la littérature des contraintes numériques.

2 Notions de base

Cette section présente quelques rappels sur les CSPs numériques, ainsi que les notations utilisées dans le reste du papier. Le lecteur pourra se reporter à [16, 6, 12, 9] pour plus de détails.

2.1 Notations

L'intervalle $\mathbf{x} = [\underline{x}, \overline{x}]$ dénote l'ensemble des réels x tels que $\underline{x} \leq x \leq \overline{x}$. Les variables sont notées x, y, les vecteurs de variables par X, Y et les vecteurs d'intervalles - ou *boîtes* - par $\mathbf{X}, \mathbf{Y}$. Le milieu $m(\mathbf{x})$ de l'intervalle $\mathbf{x}$ est $(\overline{x} + \underline{x})/2$ et sa taille est le nombre positif $\overline{x} - \underline{x}$. Une union d'intervalles sera notée $\mathbf{U_x} = \bigcup \mathbf{x}_{(j)}$, où $\mathbf{x}$ est le plus petit intervalle englobant tous les sous-intervalles $\mathbf{x}_{(j)}$. Précisons que ces $\mathbf{x}_{(j)}$ sont disjoints et triés par borne inférieure croissante, c'est-à-dire $\overline{x}_{(j)} < \underline{x}_{(j+1)}$. Nous considérons dans ce papier les CSPs numériques définis par un ensemble de contraintes $\mathbf{C} = \{c_1, \ldots, c_m\}$, mettant en jeu le vecteur de variables $X = (x_1, \ldots, x_n)$ dont les valeurs sont restreintes à la boîte $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$.

2.2 Arithmétique des intervalles

Soit f, une fonction à valeurs réelle et à n inconnues $X = (x_1, \ldots, x_n)$. L'évaluation par intervalles de f pour une boîte donnée $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ est un intervalle $\mathbf{y}$ tel que :

$$\underline{y} \leq f(x_1, \ldots, x_n) \leq \overline{y}, \ \forall x_i \in \mathbf{x}_i, 1 \leq i \leq n$$

En d'autres termes, $\mathbf{y}$ est un intervalle contenant les valeurs de f lorsque les valeurs des inconnues sont restreintes à la boîte $\mathbf{X}$.

La manière la plus simple de calculer cet intervalle est de substituer tous les opérateurs mathématiques, les constantes et les variables par leur extension aux intervalles. Une arithmétique des intervalles étendue permet de calculer une union d'intervalles contenant les valeurs de f en appliquant le même principe. Par exemple, soit $f(x) = \sqrt{x}$ pour $x \in \mathbf{x} = [4, 9]$. L'évaluation de f par rapport à $\mathbf{x}$ est l'intervalle $[-3, 3]$, alors que l'union d'intervalles contenant toutes les valeurs de f est $[-3, -2] \cup [2, 3]$.

2.3 Consistances locales

La plupart des solveurs de contraintes sont basés sur la Hull-consistance [15, 2] ou la Box-consistance [19, 3]. Ces consistances établissent

une propriété sur les bornes des domaines. De manière informelle, soit c_j une contrainte n-aire définie par $f_j(x_1, \ldots, x_n) = 0$. c_j est consistante si pour chacune de ses variables x_i, les bornes $\underline{x}_i$ et $\overline{x}_i$ possèdent un support dans les domaines des autres variables.

Le filtrage des domaines consiste à appliquer itérativement un opérateur de réduction associé à chaque couple (c_j, x_k). Ces opérateurs réduisent les bornes des domaines de x_k en calculant une approximation par intervalle de la fonction de projection π_j^k, qui exprime la variable x_k en fonction des des autres variables de c_j. Notons que l'évaluation exacte de π_j^k n'est en général pas un simple intervalle, mais souvent une union d'intervalles comme le montre la figure 1.

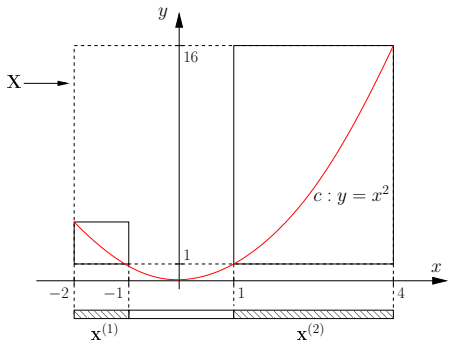
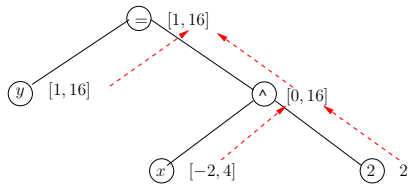
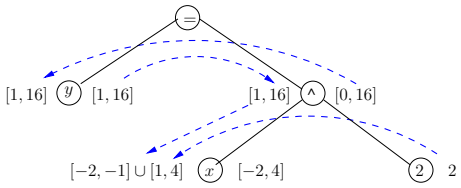


FIG. 1 – Considérons la contrainte $c : y = x^2$ avec $x \in [-2, 4]$ et $y \in [1, 16]$. Les fonctions de projection associées sont $\pi^x(y) = \sqrt{y}$ et $\pi^y(x) = x^2$. L'approximation de $\pi^x(y)$ calculée par la Hull-consistance ou par la Box-consistance sur les intervalles $\mathbf{x}$ et $\mathbf{y}$ est $[-2, 4]$, alors que l'évaluation exacte de $\pi^x(y)$ est $[-2, -1] \cup [1, 4]$.

L'une des implantations les plus efficaces de la Hull-consistance (*Realpaver* [5]) est basée sur la procédure HC4revise qui implémente les opérateurs de réduction. Cette procédure calcule une réduction des domaines en parcourant la représentation arborescente de la contrainte des feuilles à la racine (propagation ascendante) et inversement (propagation descendante). La figure 2 illustre l'exécution de HC4revise sur la contrainte $c : y = x^2$ avec $x \in [-2, 4]$ et $y \in [1, 16]$. La propagation ascendante (figure 2(a)) évalue par intervalle chaque noeud de l'arbre en partant des feuilles et en remontant à la racine. La propagation descendante (figure 2(a)) parcours l'arbre dans le sens inverse en utilisant les fonctions de projections associés aux opérateurs. Par exemple, la projection de c sur x est $x \leftarrow \sqrt{y}$; le domaine de y étant $[1, 16]$, le domaine de x est réduit à $[-2, -1] \cup [1, 4]$. HC4revise identifie un trou dans le domaine de x : l'intervalle ouvert $]-1, 1[$, qui n'a pas de support pour c lorsque y est dans l'intervalle $[1, 16]$.



(a) Propagation ascendante



(b) Propagation descendante

FIG. 2 – L'opérateur de réduction HC4revise.

Les consistances locales basées sur HC4revise identifient donc des trous dans les domaines des variables, c'est-à-dire des intervalles sur lesquels plusieurs contraintes ne sont pas satisfaites. Ces trous permettent de calculer une réduction plus forte des domaines mais ne sont plus utilisés après l'étape de filtrage.

La section suivante montre comment ces trous peuvent être exploités pour orienter la recherche de solutions.

3 Recherche orientée par les trous

Ce papier présente une nouvelle stratégie de recherche, nommée `TopSearch` (Topological Search), dérivée de la bissection standard. Contrairement aux stratégies classiques de bissection, `TopSearch` exploite la distribution des solutions à l'intérieur des domaines. `TopSearch` s'appuie sur les trous identifiés pendant l'étape de filtrage dans les domaines de chaque va-

riable, c'est-à-dire un ensemble d'intervalles disjoints qui ne contiennent aucune solution du CSP.

Function `TopSearch`(in :$\mathbf{X}_0, \mathbf{C}$ out : S)

```
    %% X_0 = (x_1, ..., x_n)
1 : Q ← X_0
2 : S ← ∅
3 : while Q ≠ ∅ do
4 :   Extract X from Q
5 :   X ← Prune(X, C)
6 :   if X ≠ ∅ then
7 :     if w(X) ≤ ω_sol then
8 :       %% S is a solution
9 :       S ← S ∪ X
10 :    else
11 :      U_X ←ComputeGaps(X, C)
12 :      if some gap exists then
13 :        Select U_{x_k} from U_X
14 :        Select g_{k(p)} from U_{x_k}
15 :        x_k^1 ← x̄_{k(p)}
16 :        x_k^2 ← x_{k(p+1)}
17 :      else
            %% Standard bisection
18 :        Select x_k from X
19 :        x_k^1 ← m(x_k)
20 :        x_k^2 ← m(x_k)
21 :      endif
22 :      Q ← Q ∪ (x_1, ..., [x_k, x_k^1], ..., x_n)
23 :      Q ← Q ∪ (x_1, ..., [x_k^2, x̄_k], ..., x_n)
24 :    endif
25 :  endif
26 : endwhile
27 : return S
```

FIG. 3 – Schéma général de l'algorithme `TopSearch`.

Ces trous apportent une information significative à la fois pour choisir la direction de coupe et le point de coupe dans le domaine choisi. `TopSearch` découpe en priorité les domaines pour lesquels des trous ont été identifiés. La découpe proprement dite consiste à éliminer un trou du domaine sélectionné, créant ainsi deux sous-problèmes disjoints.

Le schéma général de ce nouvel algorithme est donné par la figure 3. Le domaine courant est tout d'abord réduit par la procédure `Prune` (ligne 5), qui est un algorithme standard de filtrage basé sur la Box-consistance ou la Hull-consistance. Si la boîte courante $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ n'est pas une solution du CSP, `TopSearch` collecte les trous dans les domaines $\mathbf{x}_i$ (ligne 11).

$\mathbf{U_X}$ dénote le vecteur d'union d'intervalles $(\mathbf{U}_{\mathbf{x}_1}, \ldots, \mathbf{U}_{\mathbf{x}_n})$ retourné par la procédure `ComputeGaps` (voir section 4). Pour chaque variable x_i, cette procédure calcule une union d'intervalles

$$\mathbf{U}_{\mathbf{x}_i} = \bigcup \mathbf{x}_{i(j)},$$

où $\mathbf{x}_{i(j)} = [\underline{x}_{i(j)}, \overline{x}_{i(j)}]$ dénote le j-ème sous-domaine de la variable x_i.

Le p-ème trou de $\mathbf{U}_{\mathbf{x}_i}$ est l'intervalle ouvert entre le p-ème et le $p+1$-ème sous-domaine :

$$\mathbf{g}_{i(p)} =]\overline{x}_{i(p)}, \underline{x}_{i(p+1)}[$$

`TopSearch` sélectionne un domaine contenant au moins un trou (ligne 13). Puis, les points de coupe sont déterminés de manière à éliminer un trou $\mathbf{g}_{k(p)}$ (lignes 15 et 16). Plusieurs heuristiques pour sélectionner la direction de coupe (ligne 13) ainsi que le trou à éliminer (ligne 14) ont été étudiées (voir section 5).

Si aucun trou n'a été identifié, un bissection classique est mise en oeuvre (lignes 18-20). La stratégie de choix de la direction de coupe utilisée est une des heuristiques décrites dans la section 1 (`RR`, `LF`, ou `MS`).

La section suivante présente la procédure `ComputeGaps` qui collecte les trous calculés pendant la phase de filtrage.

4 Calcul des trous

Cette section présente un algorithme pour calculer les trous dans les domaines des variables. `TopSearch` collecte et intersecte les unions d'intervalles qui ont été calculées dans chaque domaine par HC4revise durant la phase de filtrage. Le sur-coût engendré est donc limité à des opérations d'intersections entre ces unions d'intervalles.

En d'autre termes, pour chaque couple (x_i, c_j), HC4revise calcule une union d'intervalles qui approxime la fonction de projection $\pi_i^{(j)}$. Pour chaque variable x_i, l'algorithme `ComputeGaps` calcule l'intersection de toutes les approximations des fonctions de projection associées aux contraintes mettant en jeu x_i (voir figure 4).

Plus précisément, soient c_j une contrainte et x_i une variable de c_j et soit $\mathbf{U}_{\mathbf{x}_i}^{(j)}$ l'approximation exacte de π_j^i qui a été calculée pendant la phase de filtrage. L'union d'intervalles

calculées pour la variable x_i par l'algorithme `ComputeGaps` (voir figure 5) est l'intersection des $\mathbf{U}_{\mathbf{x}_i}^{(j)}$, pour toutes les contraintes c_j dans lesquelles x_i apparaît.

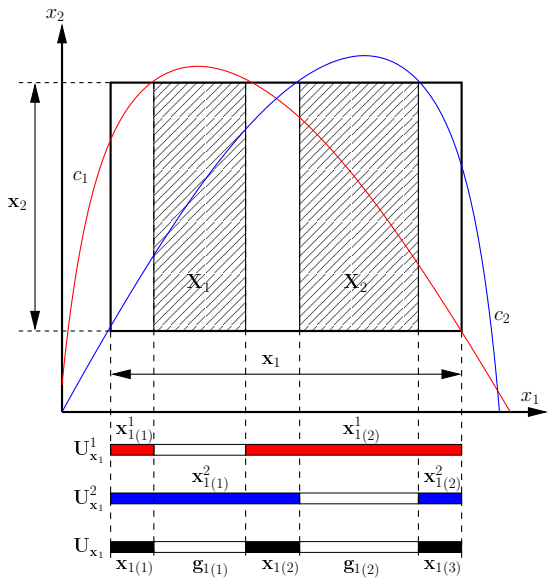


FIG. 4 – L'approximation des projections de c_1 et c_2 sur la variable x_1 sont respectivement $\mathbf{U}_{\mathbf{x}_1}^1 = \mathbf{x}_{1(1)}^1 \cup \mathbf{x}_{1(2)}^1$ et $\mathbf{U}_{\mathbf{x}_1}^2 = \mathbf{x}_{1(1)}^2 \cup \mathbf{x}_{1(2)}^2$. L'union d'intervalles générée par `ComputeGaps` pour la variable x_1 est l'intersection de $\mathbf{U}_{\mathbf{x}_1}^1$ et $\mathbf{U}_{\mathbf{x}_1}^2$: $\mathbf{U}_{\mathbf{x}_1} = \mathbf{x}_{1(1)} \cup \mathbf{x}_{1(2)} \cup \mathbf{x}_{1(3)}$. Les deux trous identifiés, $\mathbf{g}_{1(1)}$ et $\mathbf{g}_{1(1)}$, permettent d'éliminer les boîtes inconsistantes $\mathbf{X}_1$ et $\mathbf{X}_2$, dans lesquelles une des deux contraintes n'est pas vérifiée.

En terme d'implantation, le calcul des $\mathbf{U}_{\mathbf{x}_i}^{(j)}$ a été réalisé par une légère modification de la procédure HC4revise. Pendant l'évaluation descendante de l'arbre, lorsque le noeud considéré est une variable x_i et que des trous ont été détectés dans son domaine $\mathbf{x}_i$, l'union d'intervalles associé à x_i est intersecté avec $\mathbf{U}_{\mathbf{x}_i}$.

La section suivante présente une comparaison entre l'algorithme standard de bissection et l'algorithme de recherche `TopSearch` sur une série de CSPs numériques.

5 Résultats expérimentaux

Cette section présente des résultats obtenus avec `TopSearch` sur une série de problèmes classiques :
- *i4* [19] est un problème classiques issus de

Function `ComputeGaps`(in :$\mathbf{X}, \mathbf{C}$ out : $\mathbf{U}_\mathbf{X}$)

%% $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$, $\mathbf{U}_\mathbf{X} = (\mathbf{U}_{\mathbf{x}_1}, \ldots, \mathbf{U}_{\mathbf{x}_n})$
1 : $\mathbf{U}_\mathbf{X} \leftarrow \mathbf{X}$
%% Set $\mathbf{U}_{\mathbf{x}_i} = \mathbf{x}_i$ for $1 \leq i \leq n$
2 : **for each** $c_j \in \mathbf{C}$ **do**
3 : **for each** $x_i \in \mathrm{Vars}(c_j)$ **do**
4 : $\mathbf{U}_{\mathbf{x}_i} \leftarrow \mathbf{U}_{\mathbf{x}_i} \cap \mathbf{U}_{\mathbf{x}_i}^{(j)}$
%% $\mathbf{U}_{\mathbf{x}_i}^{(j)}$ is computed by HC4revise
5 : **endfor**
6 : **endfor**
7 : **return** $\mathbf{U}_\mathbf{X}$

FIG. 5 – Computation of the gaps

l'arithmétique des intervalles.
- *d1* [7] est une application de la cinématique directe de robots.
- *nbody5* [14] est une instance du problème à 5 corps issu de la mécanique céleste.
- *eco7* et *eco8* [19] sont deux applications de modélisation économique.
- *ponts* [10], *ext-penta* et *ext-penta*X [1] sont des problèmes de contraintes géométriques, où *ext-penta*X est une instance particulière de *ext-penta*.
- *caprasse* [18] est un système d'équations polynomiales issu des problèmes de Posso.

	n	m	s	$\mathbf{X}$
d1	12	12	16	$[-10^3, 10^3]$
eco8	8	8	8	$[-10^3, 10^3]$
eco7	7	7	8	$[-10^3, 10^3]$
ext-penta0	18	19	64	$[-10, 10]$
ext-penta1	18	19	64	$[-10, 10]$
ext-penta2	18	19	320	$[-10, 10]$
ponts	26	26	128	$[-10^2, 10^2]$
ext-penta	19	19	448	$[-10, 10]$
nbody5	6	6	12	$[-10^8, 10^8]$
i4	10	10	1024	$[-1, 1]$
caprasse	4	6	28	$[-10, 10]$

TAB. 1 – n est le nombre de variables, m est le nombre de contraintes, s est le nombre solutions dans la boîte initiale $\mathbf{X}$.

La table 1 donne une description syntaxique de ces problèmes.

La bissection classique est comparée avec `TopSearch` ; toutes les deux sont combinées avec `RR`. Le filtrage est réalisé par HC4 seul (tableau 2), ou combiné avec la méthode de New-

	RR		TopSearch(ALG)+RR			Réduction	
	t(s)	B	t(s)	B	H	t	B
$d1$	25.79	264685	20.06	203987	1	**-22.21** %	**-22.93** %
$eco8$	135.02	1614495	113.94	1360061	1	**-15.61** %	**-15.75** %
$eco7$	58.36	754885	15.19	231595	1	**-73.97** %	**-69.32** %
$ext-penta0$	0.34	873	0.12	423	51	**-64.70** %	**-51.54** %
$ext-penta1$	0.33	263	0.06	255	17	**-81.81** %	**-3.04** %
$ext-penta2$	0.79	2825	0.26	2047	9	**-67.08** %	**-27.53** %
$ponts$	35.01	174915	33.78	171251	1043	**-3.51** %	**-2.09** %
$nbody5.1$	127.71	1756139	120.85	1645977	74882	**-5.37** %	**-6.27** %
$i4$	0.86	2047	0.79	2047	1023	**-8.13** %	**0** %
$caprasse$	0.80	6527	0.80	6527	0	0 %	0 %
$ext-penta$	938.4	1006031	823.4	846799	12833	**-12.26** %	**-15.83** %

TAB. 2 – Résultats expérimentaux pour HC4 combiné avec RR et ALG. t est le temps de calcul en secondes, B est le nombre de boîtes générées et H est le nombre de fois qu'un trou a été utilisé par TopSearch

ton multivariables par intervalles (tableau 3). TopSearch est implanté avec Realpaver [5], un des solveurs d'intervalles les plus efficaces.

Différentes heuristiques de choix de direction de coupe et de choix de point de coupe ont été étudiées :
- *Absolute Largest Gap* (ALG) : Le domaine sélectionné $\mathbf{x}_k$ contient le plus grand trou identifié par ComputeGaps :

$$k = \max_{1\leq i\leq n} \{\max_j w(\mathbf{g}_{i(j)})\}$$

Le point de coupe choisi correspond au trou le plus grand. Cette heuristique minimise la taille des sous-problèmes générés.
- *Relative Largest Gap* (RLG) : Une variation de ALG qui consiste à maximiser le rapport entre la taille du plus grand trou et la taille du domaine :

$$k = \max_{1\leq i\leq n} \{\max_j \frac{w(\mathbf{g}_{i(j)})}{w(\mathbf{x}_i)}\}$$

Cette heuristique maximise la réduction du domaine sélectionné relativement à sa taille.
- *Absolute Smallest Gap* (ASG) et *Relative Smallest Gap* (RSG) sont basées sur la même idée que (ALG) et (RLG), mis à part que les trous éliminés minimisent les critères respectifs. Ces heuristiques éliminent de petits trous, souvent difficiles à identifier en utilisant une bissection standard.

Les tables 2 et 3 présentent les résultats obtenus pour TopSearch avec l'heuristique ALG. Les résultats obtenus avec les autres heuristiques sont très proches (moins de 2% de différence en terme de temps de calcul). Notons que tous les résultats ont été obtenus sur un PC muni d'un processeur cadencé à 3Ghz et de 512Mo de RAM. Pour chaque problème, toutes les solutions ont été recherchées.

Le tableau 2 montre que TopSearch améliore les performances de la bissection classique en terme de temps de calcul. Des améliorations significatives ont notamment été obtenues pour *eco7, ext-penta0, ext-penta1* et *ext-penta2* , pour lesquels le temps de calcul est fortement réduit : de 65% à 82% de réduction. Le tableau 2 montre que le nombre de bissections a également été réduit pour la plupart des problèmes, notemment de 70% pour *eco7*.

Pour *eco7* et *eco8*, le premier filtrage a permis de détecter un trou dans deux domaines différents. Autrement dit, TopSearch a été mise en oeuvre une seule fois, lors de la première étape du processus d'enumération. Malgré tout, le temps de calcul a été considérablement réduit (de 15% pour *eco8* et de 74% pour *eco7*. TopSearch indique des choix de variables qui peuvent influencer de manière significative le processus de recherche.

De plus, la réduction du temps de calcul n'est pas nécessairement liée à la réduction du nombre de boîte générées par la bissection. Par exemple, pour *ext-penta1*, le nombre de bissections n'a été réduit que de 3% alors que le temps de calcul a été réduit de 81%. De même, pour *i4*, toutes les boîtes ont été générées en utilisant TopSearch (2047 boîtes générées = 1023 bissections), ce qui a amélioré l'efficacité de proces-

	RR		TopSearch(ALG)+RR			Réduction	
	t(s)	B	t(s)	B	H	t	B
$d1$	0.40	1447	0.29	1263	1	**-27.50** %	**-12.71** %
$eco8$	57.13	353155	40.88	246927	49	**-28.44** %	**-30.07** %
$eco7$	61.48	468799	12.87	107817	11	**-79.06** %	**-77.00** %
$ext-penta0$	0.44	873	0.16	423	51	**-63.63** %	**-51.54** %
$ext-penta1$	0.37	263	0.09	255	17	**-75.67** %	**-3.04** %
$ext-penta2$	1.17	2825	0.56	2047	9	**-52.13** %	**-27.53** %
$ponts$	23.76	32643	16.31	22071	1057	**-31.35** %	**-32.38**%
$nbody5.1$	76.09	841461	67.48	752063	84203	**-11.31** %	**-10.62** %
$i4$	1.14	2047	1.04	2047	1023	**-8.77** %	**0** %
$caprasse$	0.61	2567	0.62	2567	0	1.63 %	**0** %
$ext-penta$	448.47	1006031	394.00	846799	12833	**-12.14** %	**-15.82** %

TAB. 3 – Résultats expérimentaux pour `HC4+Newton` combiné avec `RR` et `ALG`. t est le temps de calcul en secondes, B est le nombre de boîtes générées et H est le nombre de fois qu'un trou a été utilisé par `TopSearch`

sus de filtrage et donc le temps de calcul. Cela indique que l'élimination des trous dans les domaines peut influer sur l'efficacité du filtrage. Les limites de `TopSearch` sont illustrées par *caprasse*, pour lequel aucun trou n'a été identifié.

6 Conclusion

Nous avons introduit dans ce papier un nouvel algorithme de recherche pour la résolution de CSPs non-linéaires. Cette stratégie de recherche exploite les trous qui sont identifiés durant la phase de filtrage. Ces trous, calculés avec un sur-coût minime, apportent des informations significatives à la fois pour le choix de la direction de coupe et pour le choix du point de coupe dans le domaine sélectionné. Les résultats expérimentaux montrent que dans le cas où des trous ont été identifiés pendant la recherche de solution, le nombre de bissection ainsi que le temps de résolution peuvent être réduits de manière très significative. `TopSearch` fournit donc une alternative intéressante à la bissection classique.

Références

[1] H. Batnini and M. Rueher. Décomposition sémantique pour la résolution de systèmes d'équations de distances. *JEDAI(Journal Electronique d'Intelligence Artificielle)*, 2(1), 2004. Édition spéciale JNPC 2003.

[2] F. Benhamou, F. Goualard, L. Granvilliers, and J.F. Puget. Revising hull and box consistency. In *International Conference on Logic Programming*, pages 230–244, 1999.

[3] F. Benhamou, D. McAllister, and P. Van Hentenryck. CLP(intervals) revisited. In Maurice Bruynooghe, editor, *Proceedings of the 1994 International Symposium*, pages 124–138. MIT Press, 1994.

[4] B. Faltings. Arc-consistency for continuous variables. *Artif. Intell.*, 65(2) :363–376, 1994.

[5] L. Granvilliers. On the combination of interval constraint solvers. *Reliable Computing*, 7(6) :467–483, 2001.

[6] E.R. Hansen. *Global optimization using interval analysis.* Marcel Deckler, 1992.

[7] H. Hong and V. Stahl. Safe starting regions by fixed points and tightening. *Computing*, 53(3-4) :322–335, 1995.

[8] E. Hyvönen. Constraint reasoning based on interval arithmetic : the tolerance propagation approach. *Artificial Intelligence*, 58(1) :71–112, December 1992.

[9] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis.* Springer, 2001.

[10] C. Jermann, G. Trombettoni, B. Neveu, and M. Rueher. A constraint programming approach for solving rigid geometric systems. In *Proc. of CP'00 : Sixth Internatio-*

nal Conference on "Principles and Practice of Constraint Programming", LNCS 1894, pages 233–248, Singapore, September 2000. Springer Verlag.

[11] Narendra Jussien and Olivier Lhomme. Dynamic domain splitting for numeric CSPs. In *European Conference on Artificial Intelligence*, pages 224–228, 1998.

[12] R.B. Kearfott. A review of techniques in the verified solution of constrained global optimization problems. In R. Baker Kearfott and Vladik Kreinovich, editors, *Applications of Interval Computations*, pages 23–59. Kluwer, Dordrecht, Netherlands, 1996.

[13] R.B. Kearfott. *Rigorous global search : continuous problems*. Kluwer, 1996.

[14] I. Kotsireas and I. Lazard. Central configurations of the 5-body problem with equal masses in three dimensional space. In *Proceedings of CASC*, 1998.

[15] O. Lhomme. Consistency techniques for numerical csps. In *IJCAI-93*, pages 232–238, 1993.

[16] R. Moore. *Interval analysis*. Prentice-Hall, 1977.

[17] D. Ratz. Box-splitting strategies for the interval Gauss–Seidel step in a global optimization method. *Computing*, 53 :337–354, 1994.

[18] C. Traverso. The posso test suite examples, 1993. Available at http ://www.inria.fr/saga/POL/index.html.

[19] P. Van Hentenryck, D. McAllister, and D. Kapur. Solving polynomial systems using a branch and prune approach. *SIAM, Journal of Numerical Analysis*, 34(2) :797–827, April 1997.