La propagation d'intervalles vue comme un problème de bandit-manchot non stationnaire

Frédéric Goualard, Christophe Jermann

La propagation d'intervalles vue comme un problème de bandit-manchot non stationnaire

Frédéric Goualard **Christophe Jermann**

LINA, FRE CNRS 2729 – Université de Nantes
{Frederic.Goualard|Christophe.Jermann}@univ-nantes.fr

Résumé

Durant la résolution de systèmes d'équations non linéaires, de nombreuses projections sont utilisées sans résultat. Les heuristiques visant à sélectionner *a priori* les meilleures projections ne donnent pas toujours de bons résultats ; en fait, nous avons démontré récemment qu'aucune telle heuristique ne peut fonctionner en général car l'intérêt d'une projection varie en cours de résolution.
Dans cet article, nous considérons le problème de la sélection dynamique des projections comme un problème de bandit-manchot non stationnaire ; nous montrons que l'utilisation de méthodes d'apprentissage conduit à un nouvel algorithme de propagation qui surpasse les algorithmes standards sur plusieurs problèmes, et surtout offre des performances stables sur l'ensemble des problèmes de notre banc d'essai.

1 Introduction

Nous nous intéressons à la résolution de systèmes d'équations sur les réels de la forme :

$$\begin{aligned} f_1(x_1,\ldots,x_n) &= 0 \\ &\ddots \\ f_n(x_1,\ldots,x_n) &= 0 \end{aligned} \tag{1}$$

La résolution numérique de tels systèmes consiste en la résolution itérative d'une séquence de n équations univariées, dites *projections*, de la forme :

$$f_i(\alpha_1,\ldots,\alpha_{i-1},x_i,\alpha_{i+1},\ldots,\alpha_n) = 0 \tag{2}$$

où les α_j sont les valeurs courantes des variables x_j. Cette séquence est itérée jusqu'à atteindre un point fixe (ou jusqu'à ce que les valeurs des variables ne varient plus beaucoup).

Dans le cas linéaire, la vitesse de convergence de ces méthodes dépend crucialement de l'ordre des équations et des variables qui détermine la *transversale* utilisée, *i.e.*, l'ensemble des n *projections* (f_i, x_i) considérées dans la séquence d'équations (2)). Un résultat classique indique que les équations et variables doivent être réarrangées de sorte à rendre la matrice des coefficients strictement diagonale dominante [14].

Dans le cas non linéaire, cet ordre a aussi son importance, ce qui a conduit plusieurs chercheurs [15, 5, 8] à proposer des critères heuristiques permettant de choisir ces n projections. Nous avons cependant montré [6] que ces tentatives sont vouées à l'échec dans le cas général, les problèmes non-linéaires ne disposant pas toujours d'une transversale statique. Plusieurs travaux [7, 13, 10, 6] suggèrent qu'il existe néanmoins un sous-ensemble de projections, pas nécessairement réduit à n et dont la composition peut varier en cours de résolution, permettant une convergence plus rapide.

Dans cet article, nous rapprochons le problème de sélection dynamique des meilleures projections (PSP) et le problème du bandit-manchot (*multi-armed bandit problem*, MABP), et plus précisément de sa variante non stationnaire (MABP_{ns}) [16]. Nous proposons alors d'adapter l'algorithme d'apprentissage par renforcement *recency weighted average* (`rwa`), considéré comme la meilleure approche pour le MABP_{ns}, au PSP. Pour ce faire, nous l'introduisons dans un algorithme de propagation de contraintes [11]. Un banc d'essai comportant une douzaine de problèmes aux caractéristiques très différentes nous permet de montrer que ce nouvel algorithme résout certains de ces problèmes de façon plus efficace que la propagation classique ou les heuristiques de choix statique, mais surtout qu'il a des performances stables sur l'ensemble de ces problèmes, offrant un temps de résolution toujours proche de celui de la meilleur méthode pour chaque problème.

La suite de l'article est organisée comme suit : la section 2 rappelle les principes de la propagation d'intervalles et établit le lien entre le PSP et le MABP$_{ns}$; la section 3 introduit l'algorithme d'apprentissage `rwa` et présente son intégration dans un algorithme de propagation de contraintes ; la section 4 présente les résultats de notre étude expérimentale sur un banc d'essai d'une douzaine de problèmes ; la conclusion discute ces résultats et propose des pistes de recherche dans la continuité de ce travail.

2 Propagation et MABP

La résolution de contraintes par intervalles [2] combine l'arithmétique d'intervalles [12] (afin de calculer de façon fiable les solutions des équations du système ((1))) et la propagation de contraintes [11] (afin de tirer parti de la structure des problèmes).

L'algorithme de filtrage par *box consistency* [3] (aussi appelé `BC3` [5]) est un bon exemple d'un tel algorithme de propagation (cf. algorithme 1). Il est similaire à la méthode de Gauss-Seidel non linéaire généralisée : un intervalle de réels I_j représente le domaine de chaque variable x_j ; `BC3` réduit les domaines en appliquant l'algorithme *bc3revise* sur chaque projection (f_i, x_j) devant être traitée ; une projection doit être traitée si le domaine des variables qu'elle utilise a changé depuis la dernière fois qu'elle a été traitée. Les détails de l'algorithme *bc3revise* importent peu dans cet article. Il suffit de savoir qu'étant données une projection (f_i, x_j) et une boite $D = I_1 \times \cdots \times I_n$, il réduit autant que possible le domaine I_j de x_j en préservant toutes les solutions de $f_i(I_1, \ldots, I_{j-1}, x_j, I_{j+1}, \ldots, I_n) = 0$.

Algorithm 1 `BC3` (**in** $T = \{(f_i, x_j) \mid i, j \in \{1, \ldots, n\}\}$; **in/out** $D = I_1 \times \cdots \times I_n$)

```
1  S ← T
2  while S ≠ ∅ do
3     (f_i, x_j) ← choix d'une projection dans S
4     D' ← bc3revise(f_i, x_j, D)
5     if I'_j ⊊ I_j then
6        S ← S ∪ {(f_β, x_γ) ∈ T | x_j apparaît dans f_β}
7        D ← D'
8     end if
9     S ← S \ {(f_i, x_j)}
10 end while
```

Les algorithmes de propagation tels que `BC3` sont généralement utilisés dans un algorithme de type *branch-and-prune* qui découpe les domaines après chaque point fixe de propagation et tant que les domaines sont plus larges qu'une précision définie par l'utilisateur. Ces algorithmes collectent ainsi toutes les boites-solutions du système ((1)).

La différence fondamentale entre `BC3` et les algorithmes numériques classiques est qu'il considère toutes les projections (f_i, x_j) (en pire cas, n^2 projections) et non pas seulement une transversale (n projections). Si cette approche évite les mauvais choix de transversale, elle s'avère rapidement impraticable pour les problèmes denses de grande taille ; en effet, le nombre de projections à considérer ralentit alors considérablement l'algorithme.

Plusieurs auteurs ont proposé des heuristiques pour choisir statiquement [15, 5, 8] ou semi-statiquement [7] n projections parmi les n^2 possibles. Ces approches ont fait leurs preuves sur certains problèmes, mais chacune échoue sur d'autres problèmes, produisant des temps de résolution très important par rapport à `BC3`. Effectivement, nous avons montré [6] que le système 1 n'admet en général pas une transversale statique. Par ailleurs, il n'est pas aisé de distinguer les problèmes qui ont une transversale statique des autres, ce qui rend les approches heuristiques peu fiables. Nous avons toutefois observé que pratiquement tous les problèmes disposent bien d'un sous-ensemble de projections permettant une convergence plus rapide de l'algorithme `BC3`, mais cet ensemble peut contenir plus de n projections et sa composition peut varier au cours de la résolution. Ainsi, s'il était possible de connaître à l'avance la *qualité* de chaque projection à n'importe quelle étape de la résolution[1], le choix de la prochaine projection (ligne 3 de l'algorithme `BC3`) pourrait être réalisé de façon optimale.

Considérant que la qualité de chaque projection varie donc de façon inconnue, mais connaissant par ailleurs la qualité de chacune de leurs applications passées, nous cherchons ainsi à déterminer quelle est la prochaine projection à utiliser de sorte à maximiser la réduction. C'est le problème de sélection de la projection (PSP). Nous rapprochons ce problème de celui du bandit-manchot non stationnaire (MABP$_{ns}$) [16] : étant données k machines à sous dont la probabilité de gain varie de façon inconnue, et connaissant les gains obtenus par les tirages passés des différents leviers, choisir le levier qui maximise le gain au prochain tirage ; seuls les gains des leviers effectivement tirés sont observables. Considérant les projections comme les machines à sous, et la qualité d'une projection

[1] Notons que cela est possible en théorie : on peut appliquer chaque projection sur la même boite courante avant de ne retenir que celle qui a produit la réduction la plus importante. Cette approche s'avère bien entendu impraticable puisque l'application d'une seule projection coûte alors le test de toutes les projections. C'est toutefois cette méthode qui nous a permis de déterminer la dynamicité et la variation du sous-ensemble de projections intéressantes [6].

comme le gain, le parallèle entre les deux problèmes devient évident. Nous allons donc pouvoir utiliser les algorithmes adaptés aux MABP$_{ns}$ afin de traiter le PSP. Pour ce faire, nous considèrerons que la qualité d'une projection (f_i, x_j) est mesurée par la *réduction relative* $r^{(ij)}$ qu'elle permet : $r^{(ij)} = (\mathsf{w}(I_j^b) - \mathsf{w}(I_j^a))/\mathsf{w}(I_j^b)$ où $\mathsf{w}(I_j^b)$ (resp. $\mathsf{w}(I_j^a)$) est la taille du domaine de x_j avant (resp. après) application de *bc3revise* sur (f_i, x_j).

Il faut toutefois remarquer que, contrairement au MABP$_{ns}$ où les machines à sous sont indépendantes, les projections sont intrinsèquement liées : elles sont issues d'un même système d'équations. Ainsi, l'utilisation d'une projection peut avoir un impact sur la qualité future d'autres projections. Ceci implique également que, dans le cas général et même en connaissance des gains futurs, le PSP n'admet pas de solution par algorithme glouton : le choix de la meilleure projection à chaque étape n'est pas nécessairement la solution optimale sur le plan global. Dans cet article, nous négligerons ces différences afin de proposer une première application simple de technique d'apprentissage au PSP. Nous discuterons les extensions possibles pour prendre en compte cette dimension spécifique au PSP dans la conclusion.

3 Propagation et apprentissage

La méthode *recency weighted average* (`rwa`) [16] est une méthode d'apprentissage standard pour attaquer le MABP$_{ns}$. Plusieurs autres méthodes, comme ϵ`g` [16] ou encore `Exp3` [1], ont aussi été appliquée avec succès au MABP, mais il semble que `rwa` demeure l'algorithme de référence pour la variante non stationnaire qui nous intéresse ici.

Dans `rwa`, un poids W^k est associé à chaque levier k. Le levier de poids le plus élevé est toujours celui qui est tiré. Son gain g^k est comptabilisé et permet la mise à jour du poids : $W^k \leftarrow W^k + \alpha\big(g^k - W^k\big)$. La méthode nécessite donc deux paramètres : le facteur α à valeur dans $[0.0, 1.0]$ utilisé pour la mise à jour des poids ; et un poids initial W_0^k à valeur réelle positive pour chaque levier.

Afin d'appliquer `rwa` au PSP, on associe donc un poids $W^{(ij)}$ à chaque projection (f_i, x_j) ; le gain utilisé pour la mise à jour devient la réduction relative $r^{(ij)}$. En remplaçant le choix effectué en ligne 3 de l'algorithme `BC3` par l'extraction de la projection de poids maximum de l'ensemble S, et en ajoutant entre les lignes 4 et 5 l'instruction de mise à jour $W^{(ij)} \leftarrow W^{(ij)} + \alpha\big(r^{(ij)} - W^{(ij)}\big)$, on obtient un algorithme de propagation qui exploite l'information des projections appliquées afin d'apprendre les meilleurs projections à appliquer.

Cet algorithme s'avère cependant peu efficace. En effet, utiliser un seul ensemble pour toutes les projections peut créer des cycles dans l'utilisation des projections : il est possible qu'un petit sous-ensemble de projections soient utilisées avec suffisamment de succès pour qu'elle soient appliquées en boucle sans qu'aucune autre ne soit jamais testée (les poids initiaux ne le leur permettant pas), ce qui conduit à des convergence lentes [13, 10]. Afin d'éviter cet écueil, il faut respecter un certain équilibre sur la réduction des domaines des différentes variables. Nous proposons donc d'utiliser un ensemble de projections S_j par variable x_j. Nous appelons `BC3rwa` le nouvel algorithme qui en résulte (cf. algorithme 2). Cet algorithme contient une nouvelle boucle interne qui applique les n (ou moins de n si certains ensembles S_j sont vides) projections de poids maximum sur des variables différentes.

Algorithm 2 `BC3rwa` (**in :** $T = \{(f_i, x_j, W^{(ij)}) \mid i, j \in \{1, \ldots, n\}\}$; **in/out :** $D = I_1 \times \cdots \times I_n$

```
1  for all j ∈ {1,...,n} do
2    S_j ← {(f_i, x_j, W^(ij)) ∈ T | i ∈ {1,...,n}}
3  end for
4  while ∃j ∈ {1,...,n} tel que S_j ≠ ∅ do
5    S ← ⋃_{j=1}^{n} {pop(S_j)}   // Extraie les projections de poids maximum
6    for all (f_i, x_j, W^(ij)) ∈ S do
7      D' ← bc3revise(f_i, x_j, D)
8      r^(ij) ← (w(I_j) − w(I'_j))/w(I_j)
9      W^(ij) ← W^(ij) + α(r^(ij) − W^(ij))
10     if I'_j ⊊ I_j then
11       for all k ∈ {1,...,n} do
12         S_k ← S_k ∪ {(f_β, x_k, W^(βk)) ∈ T | x_j apparaît dans f_β, β ∈ {1,...,n}}
13       end for
14       D ← D'
15     end if
16   end for
17 end while
```

Comme nous l'avons déjà signalé, cet algorithme dispose de 2 paramètres. Nous avons réalisé une étude de l'influence de chacun de ces deux paramètres afin d'en déterminer les meilleures valeurs. Pour cette étude, nous avons utilisé le banc d'essai présenté à la section suivante. Dans les paragraphes qui suivent, nous donnons simplement un résumé de nos expérimentations et en indiquons nos conclusions.

3.1 Paramètre α

Pour mieux comprendre le rôle du α, il faut développer l'équation de mise à jour des poids dans la méthode `rwa` :

$$W_k^{(ij)} = W_{k-1}^{(ij)} + \alpha\big(r_k^{(ij)} - W_{k-1}^{(ij)}\big) = (1-\alpha)^k W_0^{(ij)} + \sum_{l=1}^{k} \alpha(1-\alpha)^{(k-l)} r_l^{(ij)}$$

Ainsi, lorsque α tend vers 0, une importance grandissante est donnée à l'historique complet de la projection et le poids initial joue un rôle majeur. Inversement, lorsque α tend vers 1, l'influence du poids initial disparaît en quelques applications et seul l'historique récent influe sur le poids de la projection. Autrement dit, plus α est grand, et plus l'algorithme est réactif aux variations de la qualité des projections.

Nous avons réalisé des tests avec des valeurs de α allant de 0.1 à 0.9 par pas de 0.2 ; la table 1 présente un échantillon de nos résultats. De ces essais, il ressort que le paramètre α a une influence limitée sur les performances de l'algorithme : le rapport entre le meilleur temps et le pire n'est jamais supérieur à 2. Les performances sont en moyenne un peu meilleures avec $\alpha = 0.1$ et c'est pourquoi nous effectuerons nos essais de la section suivante avec cette valeur du paramètre.

TAB. 1 – Impact du paramètre α

α	0.1	0.3	0.5	0.7	0.9
bb 1000	12	9	8	7	6
bt 20	28	28	30	30	27
ecl 8	32	32	32	34	34
ep 16	2	2	2	2	2
mc 200	976	1 087	1 116	1 368	1 522
tro 200	56	55	56	55	55
yam 10	68	68	69	67	68
Temps en secondes ; $\forall i,j \;\; W_0^{(ij)} = 1.0$					

3.2 Poids initiaux

L'initialisation des poids des projections constitue un biais important de l'algorithme. Ce biais est encore exacerbé par notre choix de fixer α à la valeur 0.1 (cf. paragraphe précédent). Nous avons testé 3 façons différentes de la réaliser :

`BC3rwa-1.` Tous les poids initiaux valent 1.0 ;

`BC3rwa-b.` L'algorithme *bc3revise* est appliqué une fois pour chaque projection sur la boite initiale, et le gain résultant (réduction relative) est utilisé comme poids initial pour la projection ; les poids prennent donc leurs valeurs entre 0.0 et 1.0 ;

`BC3rwa-r.` La Jacobienne J du système est évaluée sur la boite initiale et le poids initial $W^{(ij)}$ de chaque projection (f_i, x_j) est fixé à la mignitude[2] de J_{ij} plus 1 si elle est non nulle, sinon au rapport entre la magnitude[3] de J_{ij} et la plus grande magnitude dans J. Il s'agit là de l'heuristique définie dans [5] pour choisir statiquement les meilleures projections. Intuitivement, elle favorise les projections dont la dérivée partielle correspondante est assurément non nulle (mignitude), et sinon celle dont elle a le plus de chance d'être non nulle (magnitude). Les poids initiaux sont ici arbitrairement grands.

[2] $\mathsf{mig}\, I = \min\{|a| \mid a \in I\}$
[3] $\mathsf{mag}\, I = \max\{|a| \mid a \in I\}$

TAB. 2 – Impact des poids initiaux

Init.	`BC3rwa-1`	`BC3rwa-b`	`BC3rwa-r`
bb 1000	12	11	11
bt 20	28	27	30
ecl 8	32	31	28
ep 16	2	2	2
mc 200	976	208	97
tro 200	56	59	57
yam 10	68	69	68
Temps en secondes ; $\alpha = 0.1$			

La table 2 présente un échantillon de nos essais. Il en ressort que l'initialisation n'a pas non plus une importance cruciale pour l'algorithme exception faite du problème *mc 200*. Ce cas particulier procède de la conjonction de deux facteurs : premièrement, ce problème possède une transversale statique qui est très bien identifiée par l'heuristique basée sur l'évaluation de la Jacobienne ; deuxièmement, les poids initiaux attribués aux projections de cette transversale sont très grands avec `BC3rwa-r`, et sous l'influence du paramètre α, ils déterminent que seuls ses projections seront utilisées au cours de l'algorithme de propagation. `BC3rwa-r` semble en général une meilleure heuristique d'initialisation des poids que `BC3rwa-b`. Toutefois, le coût de calcul (ici pris en compte) de cette heuristique et le risque qu'elle a de se tromper (ou de fournir une information floue si les domaines sont trop grands, conduisant à une évaluation peu informative de la Jacobienne) nous amène à considérer que `BC3rwa-1` peut être un choix par défaut intéressant dans la plupart des cas. Nous réaliserons donc nos essais avec ces deux méthodes à la section suivante.

4 Banc d'essai

Nous allons à présent comparer nos algorithmes de propagation avec apprentissage à des algorithmes de propagation sans apprentissage :

`BC3.` l'algorithme de propagation original [3] qui considère toutes les projections (n^2 en pire cas) ;

`Best1.` Cette méthode [6], similaire à celle utilisée pour l'initialisation des poids par `BC3rwa-b`,

consiste à appliquer l'algorithme *bc3revise* sur la boite initiale pour chaque projection ; les réductions relatives obtenues constituent les entrées M_{ij} d'une matrice carrée $n \times n$. Un couplage de poids maximum est calculé sur cette matrice et les n entrées retenues dans ce couplage correspondent aux n projections formant la transversale retenue pour la propagation. L'algorithme `BC3` est alors appliqué normalement avec en entrée seulement ces n projections ;

`GH`. Cette méthode consiste à réappliquer le calcul d'une transversale à la façon de `Best1` à chaque fois qu'un point fixe de propagation est atteint (c'est-à-dire après chaque bissection dans l'algorithme *branch-and-prune*. Elle fut introduite [7] dans une forme relaxée qui ne calculait pas une transversale mais sélectionnait simplement un sous-ensemble des meilleurs projections permettant de représenter toutes les équations et toutes les variables.

Notre banc d'essai est constitué de 13 problèmes standards aux caractéristiques variées [9] :

bb 1000. *Broyden-banded* est un problème quadratique redimensionnable et peu dense. Nous avons fixé sa taille à 1000 équations et variables, ce qui représente environ 7000 projections ;

bt 20. *Broyden tridiagonal* est un problème quadratique redimensionnable et peu dense. Nous avons fixé sa taille à 20 équations et variables, ce qui représente environ 60 projections ;

cap. *Caprasse* est un problème polynomial dense à 4 équations et variables, soit 16 projections ;

com. *Combustion* est un problème quadratique peu dense à 10 équations et variables, soit 29 projections ;

dbvf 100. *Discrete boundary value function* est un problème polynomial, redimensionnable et peu dense. Sa taille est fixée à 100 équations et variables, soit environ 300 projections ;

ecl 8. *extended Crag-Levy* est un problème non algébrique redimensionnable et peu dense. Sa taille est fixée à 8 équations et variables, soit 16 projections ;

ef 20. *Extended Freudenstein* est un problème polynomial, redimensionnable et peu dense. Sa taille est fixée à 20 équations et variables, soit environ 40 projections ;

ep 16. *extended Powell* est un problème non polynomial, redimensionnable et peu dense. Sa taille est fixée à 16 équations et variables, soit 32 projections ;

mc 200. *Moré-Cosnard* est un problème polynomial redimensionnable et très dense. Sa taille est fixée à 200 équations et variables, soit 40000 projections ;

te1 10. *Trigexp1* est un problème non polynomial redimensionnable et peu dense. Sa taille est fixée à 10 équations et variables, soit environ 30 projections ;

te3 2000. *Trigexp3* est un problème non polynomial redimensionnable et peu dense. Sa taille est fixée à 2000 équations et variables, soit environ 6000 projections ;

tro 200. *Troesch* est un problème non polynomial, redimensionnable et peu dense. Sa taille est fixée à 200 équations et variables, soit environ 600 projections ;

yam 10. *Yamamura* est un problème quasi-linéaire, redimensionnable et dense. Sa taille est fixée à 10 équations et variables, soit 100 projections.

Les domaines des variables pour tout ces problèmes sont ceux proposés sur la page web de Coprin [9].

Toutes nos expérimentations ont été conduites sur un Intel Pentium IV à 3.8 GHz avec 2GB de RAM et une *Standard Unit Time* égale à $50.4\,s$ (10^8 évaluations de Shekel5 [4]). Le solveur utilisé est une bibliothèque C++ que nous avons écrite *ex nihilo*. Tous les temps, donnés en secondes, correspondent à la résolution complète (toutes les solutions sont retournées) des problèmes avec une précision de 10^{-8}.

TAB. 3 – Résolution avec et sans apprentissage

Problème	`BC3`	`Best1`	`GH`	1	R
bb 1000	11	3	5	12	11
bt 20	79	18	170	27	30
cap	370	133	280	201	213
com	0.1	0.6	0.1	0.1	0.1
dbvf 100	128	70	68	70	71
ecl 8	30	33	81	32	28
ef 20	0.2	316	3	0.1	0.1
ep 16	3	47	3	2	2
mc 200	1054	67	189	976	97
te1 10	10	442	52	10	10
te3 2000	1.0	1.9	4.5	0.5	95
tro 200	98	55	57	56	57
yam 10	470	56	1504	68	68

La table 3 présente les résultats que nous obtenons ; les colonnes 1 et R représentent respectivement les résultats des méthodes `BC3rwa`-1 et `BC3rwa`-r. Comme nous l'avons déjà mentionné, `BC3` obtient de mauvais résultats sur les grands problèmes denses (e.g., *mc 200*), s'effondrant sous le nombre de projections

à considérer. L'algorithme `Best1` se comporte correctement sur les problèmes *bb 1000* and *mc 200*. Ces problèmes disposent en effet d'une transversale statique qui est trouvée par la méthode, lui permettant de n'utiliser que les n projections les meilleures. les problèmes *ep 16*, *ef 20* et *te1 10* montrent les limites de cette méthode : ces problèmes n'ont pas de transversale statique, et le sous-ensemble des projections intéressantes y varie grandement au cours de la résolution ; l'algorithme `GH` obtient ici de meilleurs résultats car il reconsidère le choix initial à l'occasion de chaque point fixe de propagation. Cependant, lorsque le problème conduit à réaliser tôt beaucoup de bissection (e.g., *yam 10*), cet algorithme perd beaucoup de temps à réévaluer l'ensemble des projections et à produire une nouvelle transversale.

Les méthodes par apprentissage, bien que n'étant pas les plus rapides en général, se comportent de façon plus régulière, résolvant tous les problèmes dans un temps qui n'excède pas beaucoup celui de la meilleur méthode. Il est intéressant de noter leur complémentarité : pour les problèmes de taille moyenne avec une transversale statique, `BC3rwa-r` est meilleur car l'initialisation est payante ; inversement, pour les grand problèmes où l'initialisation intelligente est trop coûteuse, et pour les problèmes sans transversale statique, `BC3rwa-1` s'avère plus intéressant (e.g., *te3 2000* où 97 % du temps de `BC3rwa-r` passe dans l'évaluation de la Jacobienne).

Afin d'estimer la robustesse des méthodes par apprentissage dans des conditions initiales défavorables aux méthodes par intervalles, nous proposons un second jeu d'expérimentations où les domaines initiaux sont élargis à $[-10^6, 10^8]$ (l'asymétrie est sensée rendre la résolution plus difficile, la bissection étant moins susceptible de découper un domaine autour de 0). Les résultats sont présentés à la table 4. les temps de résolution supérieurs à 1 heure sont remplacés par un *"NA"*.

Hormis quelques *curiosités* (`BC3` résout *yam 10* plus rapidement avec de grands domaines) dont nous soupçonnons la bissection d'être responsable (l'inefficacité de projections avec les grands domaines peut conduire à plus de bissection qui amène, par chance, dans des domaines réduits plus propices à l'exploitation efficace des projections), on peut constater que `Best1` et `GH` sont plus sensibles que les autres méthodes aux domaines initiaux : *yam 10*, *com* et *te1 10* étaient résolus facilement avec leurs domaines standards et ne peuvent plus être résolus en moins d'une heure avec leurs domaines élargis.

Trois problèmes (*ecl 8*, *mc 200*, et *cap*) ne sont plus résolus par aucune méthode. Nos investigations nous amènent à penser que ceci est dû à l'augmentation excessive de leurs nombres de solutions suite à l'élargissement de leurs domaines, qui conduit à les isoler par bissection sans pratiquement utiliser la propagation.

TAB. 4 – Robustesse avec et sans apprentissage

Problèmes	`BC3`	`Best1`	`GH`	1	R
bb 1000	12	4	5	12	12
bt 20	29	18	217	30	31
cap	NA	NA	NA	NA	NA
com	3	NA	4	3	3
dbvf 100	127	70	67	69	70
ecl 8	NA	NA	NA	NA	NA
ef 20	141	7	0.4	0.1	0.1
ep 16	6	44	19	6	6
mc 200	NA	NA	NA	NA	NA
te1 10	129	NA	NA	113	105
te3 2000	2	2	5	0.9	93
tro 200	96	56	56	56	56
yam 10	278	57	NA	68	69

5 Conclusion

Pour résoudre de petits problèmes, ou des problèmes disposant d'une transversale aisément identifiable, la propagation avec apprentissage n'est pas la meilleure méthode même si son surcoût n'est pas rédhibitoire. Rappelons toutefois que les heuristiques de sélection de transversale peuvent échouer et conduire à des temps de calculs très importants. Ces heuristiques sont par ailleurs souvent sensibles aux domaines initiaux.

La propagation avec apprentissage démontre son intérêt sur les problèmes difficiles, de grande taille, sans transversale ou avec une transversale qui varie dynamiquement au cours du temps. Elle réalise le compromis idéal entre l'utilisation de toutes les projections (`BC3`) et la sélection d'une transversale statique (`Best1`). Seul son coût d'initialisation peut s'avérer important, le coût de la mise à jour des poids pouvant être négligé face au coût des opérations mises en jeu lors de la propagation. Ainsi, malgré sa simplicité, la méthode `rwa` intégrée dans l'algorithme de propagation `BC3` conduit à des algorithmes robustes capables de capturer la dynamique des projections. La table 5 présente les rapports entre le temps pris par une méthode et le meilleur temps pour chaque problème. Trouver un critère permettant de savoir quand utiliser une initialisation coûteuse serait intéressant pour utiliser la complémentarité de `BC3rwa-1` et `BC3rwa-r`.

En perspective à ce travail, il faudrait étudier les spécificités du problème de sélection des projections par rapport au problème de bandit-manchot non stationnaire. En effet, Lhomme et al. [13, 10] ont montré

TAB. 5 – Fiabilité de la propagation avec apprentissage

Problèmes	BC3	Best1	GH	1	R
bb 1000	3.7	1.0	1.7	4.0	3.7
bt 20	4.4	1.0	9.5	1.5	1.7
cap	2.8	1.0	2.1	1.5	1.6
com	1.0	6.0	1.0	1.0	1.0
dbvf 100	1.9	1.0	1.0	1.0	1.0
ecl 8	1.1	1.2	2.9	1.1	1.0
ef 20	2.0	3160.0	3.0	1.0	1.0
ep 16	1.5	15.7	1.5	1.0	1.0
mc 200	15.7	1.0	2.8	14.6	1.4
te1 10	1.0	44.2	5.2	1.0	1.0
te3 2000	2.0	3.8	9.0	1.0	190.0
tro 200	1.8	1.0	1.0	1.0	1.0
yam 10	8.4	1.0	26.9	1.2	1.2

que des effets complexes se produisent suivant l'ordre dans lequel les projections sont appliquées, effets dus aux dépendances entre les projections. Ces spécificités pourraient nous amener à lever les hypothèses simplificatrices posées en page 3. Les éléments à prendre en compte dans cette étude sont par exemple la liste des dernières projections utilisées, la taille de la boite courante, ou du domaine courant de certaines variables. Ceci permettrait peut-être de faire entrer une composante prédictive dans le calcul des poids des projections pour le moment basé uniquement sur l'historique des réductions effectuées. Bien entendu, il faudra garder en vue l'impératif d'efficacité de l'étape de mise à jour et de sélection.

Ce travail pourrait être étendu dans plusieurs directions également : nous avons considéré ici un algorithme de propagation spécifique, *bc3revise*. Il serait possible de considérer les projections associées à différents algorithmes de filtrage, nous amenant à évaluer non plus l'efficacité d'une projection mais celle d'une *opérateur de contraction* (couple projection-algorithme de filtrage) et permettant à l'apprentissage de trouver automatiquement la meilleure combinaison de filtrages pour un problème donné ; ce travail est pour l'instant dédié aux systèmes carrés d'équations non linéaires. Il pourrait être étendu au problèmes non carrés contenant également des inégalités.

Références

[1] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The non-stochastic multi-armed bandit problem. *SIAM Journal on Computing*, 32(1) :48–77, 2002.

[2] F. Benhamou. Interval constraints, interval propagation. In Panos M. Pardalos and C. A. Floudas, editors, *Encyclopedia of Optimization*, volume 3, pages 45–48. Kluwer Academic Publishers, 2001.

[3] F. Benhamou, D. McAllester, and P. Van Hentenryck. CLP(Intervals) revisited. In *Procs. Intl. Symp. on Logic Prog.*, pages 124–138. The MIT Press, 1994.

[4] L. C. W. Dixon and G. P. Szegö. The global optimization problem : an introduction. In L. C. W. Dixon and G. P. Szegö, editors, *Towards Global Optimization 2*, pages 1–15. North-Holland, 1978.

[5] F. Goualard. On considering an interval constraint solving algorithm as a free-steering nonlinear gauss-seidel procedure. In *Procs. 20th Annual ACM Symp. on Applied Computing (Reliable Comp. and Applications track)*, volume 2, pages 1434–1438. Ass. for Comp. Machinery, Inc., 2005.

[6] F. Goualard and C. Jermann. On the selection of a transversal to solve nonlinear systems with interval arithmetic. In *Procs. International Conference on Computational Science 2006*, 2006.

[7] L. Granvilliers and G. Hains. A conservative scheme for parallel interval narrowing. *Information Processing Letters*, 74 :141–146, 2000.

[8] S. Herbort and D. Ratz. Improving the efficiency of a nonlinear-system-solver using a componentwise newton method. Research report 2/1997, Institut für Angewandte Mathematik, Universität Karslruhe (TH), 1997.

[9] INRIA project COPRIN : Contraintes, OPtimisation, Résolution par INtervalles. The COPRIN examples page. Web page at `http://www-sop.inria.fr/coprin/logiciels/ALIAS/Benches/benches.html`.

[10] Yahia Lebbah and Olivier Lhomme. Accelerating filtering techniques for numeric csps. *Artificial Intelligence*, 139(1) :109–132, 2002.

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

[12] R. E. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, N. J., 1966.

[13] A. Gotlieb O. Lhomme and M. Rueher. Dynamic optimization of interval narrowing algorithms. *Journal of Logic Programming*, 37(1-2) :165–183, 1998.

[14] J. M. Ortega and W. C. Rheinboldt. *Iterative solutions of nonlinear equations in several variables*. Academic Press Inc., 1970.

[15] D. G. Sotiropoulos, J. A. Nikas, and T. N. Grapsa. Improving the efficiency of a polynomial system solver via a reordering technique. In *Procs. 4th GRACM Congress on Computational Mechanics*, volume III, pages 970–976, 2002.

[16] R. Sutton and A. Barto. *Reinforcement learning : an introduction*. MIT Press, 1998.