RAIRO Operations Research
Will be set by the publisher

RENFORCEMENT DE LA RÉSOLUTION DE CSPS BINAIRES PAR LA TECHNIQUE DE PROJECTION DE VALEURS.

Wady Naanaa[1] and Simone Pimont[2]

Abstract. This paper introduces a new algorithm for solving binary CSPs. The proposed algorithm alternates, at each branching point, a classical variable instanciation and a non-deterministic instanciation which uses two values at a time. The choice between the two instanciation strategies is controled by a single parameter which is computed by means of a neural network. Tests carried on randomly generated binary CSPs suggest that the proposed algorithm outperforms the classical Forward-Checking algorithms on many CSP instances.

Keywords: Constraints satisfaction problems, Complete methods, Resolution strategies

Mathematics Subject Classification. Computer science, Heuristics, Search strategies

Résumé. On présente dans cet article un nouvel algorithme de résolution de CSPs binaires qui alterne, à chaque point de choix, instanciation classique de variable et instanciation non-déterministe portant sur deux valeurs à la fois. Nous proposons également un critère permettant de choisir entre les deux stratégies d'instanciation. Ce critère s'appuie sur un paramètre unique calculé par réseau de neurônes. Des tests effectués sur des CSPs binaires aléatoires montrent que les performances de l'algorithme ainsi obtenu sont meilleures que celles de l'algorithme Forward-Checking sur plusieurs instances de CSPs.

The date should precede \maketitle in AMS documentclasses; reported.

[1] UR SOIE, Faculté des Sciences de Monastir, Avenue de l'environnement , 5019 Monastir;e-mail: naanaa.wady@planet.tn

[2] LIRIS, CNRS FRE 2672, Bat Nautibus, Université Claude Bernard Lyon 1, 8 bd Niels Bohr, 69622 Villeurbanne cedex;e-mail: simone.pimont@liris.cnrs.fr

1. Introduction

Les problèmes de satisfaction de contraintes (CSPs) constituent un cadre formel simple et assez général pour la modélisation et la résolution de nombreux problèmes combinatoires. Un CSP est constitué d'un ensemble de variables, chaque variable pouvant prendre une valeur choisie parmi un ensemble de valeurs appelé *domaine de valeurs*. La notion de contrainte, qui est au centre du formalisme, intervient de manière à interdire aux variables de prendre certaines combinaisons de valeurs. Résoudre un CSP revient à affecter des valeurs aux variables qui satisfassent toutes les contraintes du problème. Les CSPs sont, en général, des problèmes NP-complets et leurs applications sont diverses, citons à titre d'exemple des applications dans le domaine de la biologie moléculaire [1], dans le domaine de la télécommunication [19], ou encore des applications d'analyse structurale [12,13] ...

Compte tenu de l'intérêt pratique des CSPs, plusieurs méthodes et techniques de résolution ont été proposées. Dans cet article nous nous intéressons aux méthodes dites complètes qui ont l'avantage de trouver au moins une solution au problème si une telle solution existe. Les algorithmes complets les plus communément utilisés sont *Forward Checking* (FC) [5] et *Maintaining Arc Consistency* (MAC) [15]. Ces deux algorithmes sont deux versions améliorées d'un même algorithme initial: *Backtracking Chronologique* (CB) qui résout les CSPs en effectuant une recherche en profondeur d'abord.

Plusieurs travaux ont montré qu'aucun algorithme de résolution de CSPs n'est meilleur que les autres, bien que MAC semble être le plus utilisé [3, 4]. Toutefois, pour des CSPs de grande taille (impliquant des centaines de variables), la résolution demeure un processus coûteux en temps de calcul. L'émergence de nouveaux problèmes concrets, notamment ceux issus du domaine de la biologie moléculaire [1], renforce le besoin d'algorithmes de plus en plus performants.

Dans ce contexte, et en vue d'améliorer les performances de FC et de MAC, nous avons adapté à la résolution de CSPs une technique issue du domaine de la théorie des graphes connue sous le nom de *projection de clique* [7] ou, plus particulièrement, *projection d'arête* [10]. Nous montrons qu'il est possible d'adapter cette technique au cadre de la résolution des CSPs binaires [11] et d'obtenir ainsi une nouvelle méthode de résolution que nous appelons méthode par *Projection de Valeurs* (VP). Nous proposons ensuite une démarche qui alterne une *projection de valeurs* et une propagation locale de contraintes. Nous avons mené une étude expérimentale sur des CSPs aléatoires. Les résultats obtenus attestent que l'algorithme proposé est plus performant que FC sur de nombreuses instances de CSP binaires.

L'article est structuré comme suit: le paragraphe suivant introduit les définitions de base et des conventions de notations. Le paragraphe 3 présente la méthode proposée. Dans le paragraphe 4, on décrit l'algorithme de base qui met en oeuvre la projection de valeurs. Les premiers résultats expérimentaux ont justifié

l'introduction d'un mécanisme d'alternance contrôlé par un paramètre : résultats et algorithme final sont présentés au paragraphe 5. Enfin, le paragraphe 6 termine l'article par quelques conclusions.

2. Définitions et notations

Définition 2.1. Un problème de satisfaction de contraintes (CSP) est un triplet (X, D, C) où:

- $X = \{i_1, \ldots, i_n\}$ est l'ensemble des variables du problème.
- $D = \{D_1, \ldots, D_n\}$ est l'ensemble des domaines de valeurs associés au variables; D_k étant le domaine de valeurs de la variable i_k.
- $C = \{C_1, \ldots, C_m\}$ est l'ensemble des contraintes du problème. Chaque contrainte C_k porte sur un sous-ensemble de variables $Var(C_k) = \{i_{k_1}, \ldots, i_{k_r}\}$ et est définie pas une relation $Rel(C_k) \subseteq D_{k_1} \times D_{k_2} \ldots \times D_{k_r}$. $Rel(C_k)$ détermine les r-uplets de valeurs admises par la contrainte C_k.

Dans cet article, on s'intéresse aux CSPs qui n'impliquent que des contraintes binaires, c'est-à-dire des contraintes qui ne font intervenir que deux variables. Rappelons que tout CSP peut être transformé en un CSP binaire équivalent. Ainsi une contrainte portant sur les variables i et j sera notée C_{ij}. L'association d'une valeur $a \in D_i$ à une variable i sera notée par le couple variable-valeur (i, a) et désignée dans la suite par "valeur (i, a)".

Définition 2.2. Soient (i, a) et (j, b) deux valeurs telles que $C_{ij} \in C$. Si $((i, a), (j, b)) \in Rel(C_{ij})$ alors (i, a) et (j, b) sont dites compatibles ou cohérentes. Autrement, (i, a) et (j, b) sont incompatibles ou incohérentes.

Définition 2.3. Une instanciation I_Y des variables de $Y = \{i_{k_1}, i_{k_2}, \ldots, i_{k_r}\}$, $Y \subseteq X$ est un élément du produit cartésien $D_{k_1} \times D_{k_2} \times \ldots \times D_{k_r}$. Si $Y = X$ alors I_Y est une instanciation complète , sinon I_Y est partielle. Une solution d'un CSP $P = (X, D, C)$ est une instanciation complète qui satisfait toutes les contraintes de C.

A tout CSP binaire, on associe un graphe simple qui représente la structure du problème : le *graphe de contraintes*, graphe des dépendances entre variables et contraintes. On définit également le *graphe d'incohérences* qui représente plus finement le problème, en intégrant les relations d'incompatibilité entre valeurs des variables. C'est un graphe simple dont les sommets sont définis par les valeurs de $\bigcup_{i \in X} D_i$ et dont les arêtes relient les valeurs incompatibles.

Définition 2.4. (graphe d'incohérences)
Le graphe d'incohérences $\mu_P = (X_D, E_{\bar{C}})$ associé au CSP $P = (X, D, C)$ est un graphe simple défini par:

- $X_D = \{(i, a) | i \in X \wedge a \in D_i\}$
- $E_{\bar{C}} = \{[(i, a), (j, b)] \in X_D \times X_D | C_{ij} \in C \wedge ((i, a), (j, b)) \notin Rel(C_{ij})\}$

Signalons que la méthode que nous proposons pour résoudre les CSPs utilise le graphe d'incohérences.

Définition 2.5. Soit $P = (X, D, C)$ un CSP binaire et soit $\mu_P = (X_D, E_{\bar{C}})$ son graphe d'incohérences. Le voisinage d'un sommet (i, a) de μ_P est désigné par une fonction N définie par:

$$N(\mu_P, i, a) = \{(j, b) \in X_D | [(i, a), (j, b)] \in E_{\bar{C}}\}$$

Pour simplifier les notations, on utilisera $N(i, a)$ au lieu de $N(\mu_P, i, a)$. $N(i, a)$ regroupe donc toutes les valeurs incompatibles avec la valeur (i, a).

3. Méthode proposée

Au début du processus de résolution, les méthodes classiques du type backtracking disposent de trop peu d'informations pour considérer les valeurs dans un ordre qui favorise l'établissement rapide des solutions [8, 9, 18]. Le risque de choisir alors des valeurs qui ne mènent pas à des solutions est donc important. Les conséquences d'un choix erroné effectué à un niveau proche de la racine sont d'autant plus préjudiciables que le problème est difficile, car dans ce cas, un "mauvais" choix n'est remis en cause qu'après avoir exploré une partie importante de l'espace de recherche.

La méthode que nous proposons vise à améliorer le processus de recherche en dirigeant l'exploration par des choix coordonnés de valeurs pour les variables. On se propose ici de remplacer la démarche classique qui consiste à effectuer des choix déterministes portant sur une seule valeur à la fois, au profit d'une démarche qui considère deux valeurs pour une même variable à chaque point de choix.

3.1. Principe

Soit P un CSP binaire, i une variable de P, a et b deux valeurs possibles pour i. Notons $P|_{D_i=\{a,b\}}$ le sous-problème de P obtenu en réduisant D_i à $\{a, b\}$. Un algorithme du type backtracking commence par affecter à i l'une des deux valeurs, par exemple a. Cela conduit à résoudre un premier sous-problème qui ne contient pas i et que l'on note $P|_{i=a}$. Si $P|_{i=a}$ n'admet pas de solution, alors $P|_{i=b}$, le sous-problème obtenu à la suite de l'affectation de b à i, sera considéré à son tour.

Dans la démarche que nous proposons, la résolution de $P|_{D_i=\{a,b\}}$ conduira à un seul sous-problème qui ne contient pas la variable i. La décision qui consiste à affecter a ou b à i sera prise ultérieurement. Notons $P|_{i=a\vee b}$ le sous-problème résultant du choix non déterministe $i = a \vee i = b$. $P|_{i=a\vee b}$ est construit de telle sorte qu'il admette comme ensemble de solutions l'union des ensembles de solutions de $P|_{i=a}$ et $P|_{i=b}$. La résolution de $P|_{i=a}$ et de $P|_{i=b}$ pourra donc être remplacée par celle de $P|_{i=a\vee b}$ (voir figure 1). La définition de $P|_{i=a\vee b}$ repose sur son graphe d'incohérences et se fonde sur une adaptation de la technique de projection d'arête [10] au contexte de la résolution de CSPs binaires. Pour mettre en évidence le fait que l'on s'appuie sur les valeurs des variables pour restreindre la recherche de solutions des CSPs, on choisit d'appeler la technique: *Projection de Valeurs* (VP).

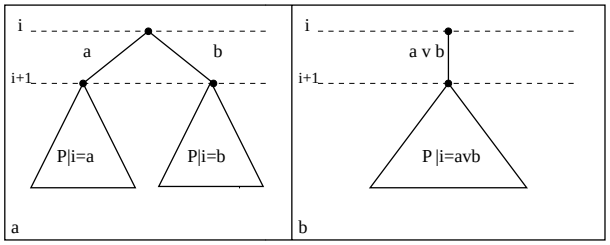

FIGURE 1. Illustration schématique d'une exploration classique et d'une exploration avec projection de valeurs.

3.2. PROJECTION DE VALEURS

Avant de définir de manière formelle la projection de valeurs, on introduit les notations suivantes:

- $N(i, ab)$ désigne $N(i, a) \cap N(i, b)$, soit l'ensemble des valeurs incompatibles à la fois avec (i, a) et (i, b);
- $N(i, a-b)$ désigne $N(i, a) - N(i, b)$, soit l'ensemble des valeurs incompatibles avec (i, a) mais compatibles avec (i, b);
- $E(Y)$ désigne l'ensemble des arêtes incidentes à l'ensemble des sommets Y dans le graphe d'incohérences;
- $E(i, a \times b)$ désigne l'ensemble des arêtes $[(j, c), (k, d)]$ telles que $j \neq k$, $(j, c) \in N(i, a-b)$ et $(k, d) \in N(i, b-a)$, soit l'ensemble des arêtes joignant toutes les valeurs incompatibles avec (i, a) mais compatibles avec (i, b) aux valeurs incompatibles avec (i, b) mais compatibles avec (i, a), à condition que chaque arête ajoutée ne relie pas deux valeurs impliquant la même variable.

L'opération de projection de valeurs est définie comme suit:

Définition 3.1. (Projection de valeurs)
Soit P un CSP binaire et soit i une variable de P telle que $\{a, b\} \subseteq D_i$. Considérons $P|_{D_i=\{a,b\}}$ le sous-problème obtenu à partir de P en réduisant le domaine de i à la paire $\{a, b\}$ et soit $\mu_P|_{D_i=\{a,b\}} = (X_D|_{D_i=\{a,b\}}, E_{\bar{C}}|_{D_i=\{a,b\}})$ son graphe d'incohérences. La projection des valeurs (i, a) et (i, b) sur $\mu_P|_{D_i=\{a,b\}}$ conduit à un sous-problème que l'on note $P|_{i=a\vee b}$ et que l'on définit par le biais de son graphe d'incohérences $\mu_P|_{i=a\vee b} = (X_D|_{i=a\vee b}, E_{\bar{C}}|_{i=a\vee b})$ de la manière suivante:

- $X_D|_{i=a\vee b} = X_D|_{D_i=\{a,b\}} - \{(i, a), (i, b)\} - N(i, ab)$
- $E_{\bar{C}}|_{i=a\vee b} = E_{\bar{C}}|_{D_i=\{a,b\}} - E(i, a) - E(i, b) - E(N(i, ab)) \cup E(i, a \times b)$

$P|_{i=a\vee b}$ est obtenu à partir de $P|_{D_i=\{a,b\}}$ en supprimant du graphe d'incohérences de $P|_{D_i=\{a,b\}}$ les deux sommets (i, a) et (i, b), ainsi que les arêtes qui leur sont incidentes (i.e. $E(i, a)$ et $E(i, b)$). On supprime également les sommets adjacents à (i, a) et à (i, b) à la fois (i.e. $N(i, ab)$), ainsi que les arêtes qui leurs sont incidentes (i.e. $E(N(i, ab))$). Enfin, les valeurs compatibles avec (i, b) mais pas avec (i, a)

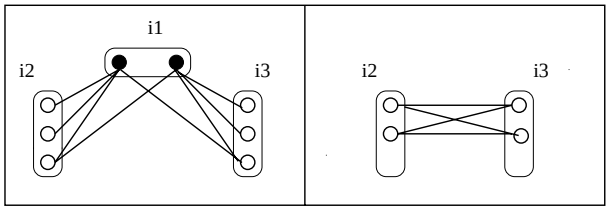

FIGURE 2. Illustration d'une opération de projection de valeurs. Dans cet exemple, on projète les deux valeurs de la variable i_1. Le deuxième graphe est le projeté du premier graphe.

(i.e. $N(i, a-b)$) sont rendues incompatibles avec les valeurs compatibles avec (i, a) mais pas avec (i, b) (i.e. $N(i, b-a)$), et ceci en ajoutant au graphe d'incohérences les arêtes de $E(i, a \times b)$. La figure 2 illustre une opération de projection de valeurs à partir des valeurs de la variable i_1.

3.3. PROJECTION ET CONSISTANCE

Proposition 3.2. *Soit S une instanciation complète d'un CSP P, alors on a:*

$$S \in Sol(P) \Leftrightarrow \forall (i,a) \in S \; S \cap N(i,a) = \emptyset$$

où $Sol(P)$ désigne l'ensemble des solutions de P. Cette équivalence découle immédiatement de la définition d'une solution (def. 2.3) et de la définition du voisinage d'une valeur dans le graphe d'incohérences (def. 2.5).

Proposition 3.3. *$P|_{D_i=\{a,b\}}$ est consistant si et seulement si $P|_{i=a\vee b}$ est consistant.*

Proof. Notons, tout d'abord que i est la seule variable qui appartient à $P|_{D_i=\{a,b\}}$ mais pas à $P|_{i=a\vee b}$.

(i) Soit S une solution de $P|_{D_i=\{a,b\}}$. Montrons qu'il existe une solution S' de $P|_{i=a\vee b}$, calculable à partir de S. Si S est une solution de $P|_{D_i=\{a,b\}}$, alors l'une des deux valeurs (i, a) ou (i, b) est dans S. Soit $S' = S - \{(i, a), (i, b)\}$. Montrons que S' est une solution de $P|_{i=a\vee b}$.

Notons, tout d'abord, que S' est une instanciation complète de $P|_{i=a\vee b}$. Raisonnons par l'absurde et supposons que S' n'est pas une solution de $P|_{i=a\vee b}$. Donc, d'après la proposition 3.2, il existe $(j, c) \in S'$ telle que $S' \cap N(j, c) \neq \emptyset$. Notons (k, d) un élément quelconque de $S' \cap N(j, c)$. Comme (k, d) est une valeur incompatible avec (j, c), on a $[(j, c), (k, d)] \in E_{\bar{C}}|_{i=a\vee b}$. Or $S' \subset S$, donc $(j, c) \in S$ et $(k, d) \in S$. Mais S est une solution de $P|_{D_i=\{a,b\}}$ donc $[(j, c), (k, d)] \notin E_{\bar{C}}|_{D_i=\{a,b\}}$, sinon (j, c) et (k, d) seraient incompatibles tout en faisant partie d'une même solution. Par conséquent, d'après la définition de $E_{\bar{C}}|_{i=a\vee b}$, on a forcément $[(j, c), (k, d)] \in E(i, a \times b)$. Cela implique que cette arête fait partie des arêtes rajoutées, et donc : $(j, c) \in N(i, a - b)$ et $(k, d) \in N(i, b - a)$ ou bien $(k, d) \in N(i, a - b)$ et

$(j,c) \in N(i, b-a)$). Supposons, sans perte de généralité, que le premier cas se réalise. Comme $N(i, a-b) \subset N(i,a)$ et $N(i,b-a) \subset N(i,b)$, on en déduit que $(j,c) \in N(i,a)$ et $(k,d) \in N(i,b)$. Par conséquent: $S \cap N(i,a) \neq \emptyset \wedge S \cap N(i,b) \neq \emptyset$. Or S est une solution telle que $(i,a) \in S$ ou $(i,b) \in S$. Donc, on doit avoir, d'après la proposition 3.2: $S \cap N(i,a) = \emptyset \vee S \cap N(i,b) = \emptyset$. D'où une contradiction.

(ii) Soit S' une solution de $P|_{i=a \vee b}$. Il s'ensuit que S' affecte des valeurs à toutes les variables de $P|_{D_i=\{a,b\}}$ sauf à i. D'après la définition de $\mu_P|_{i=a\vee b}$, S' est donc une solution partielle de $P|_{D_i=\{a,b\}}$. Montrons comment il est possible d'étendre S' à i pour obtenir une solution de $P|_{D_i=\{a,b\}}$.

Considérons dans $E_{\bar{C}}|_{i=a\vee b}$, les arêtes de $E(i, a \times b)$. Ces arêtes rendent chaque valeur de $N(i,a-b)$ incompatible avec celle de $N(i,b-a)$, sous la réserve que les valeurs ne concernent pas la même variable. Donc S' ne peut pas contenir à la fois, un élement de $N(i,a-b)$ et un élément de $N(i,b-a)$. En d'autres termes, on a : $S' \cap N(i,a-b) = \emptyset \vee S' \cap N(i,b-a) = \emptyset$.

On peut supposer sans perte de généralité que $S' \cap N(i,a-b) = \emptyset$. D'autre part, d'après la définition de $X_D|_{i=a\vee b}$, on a $S' \cap N(i,ab) = \emptyset$. Des deux dernières équations, on déduit que $S' \cap N(i,a) = \emptyset$ puisque $N(i,a) = N(i,a-b) \cup N(i,ab)$. Par conséquent, $S = S' \cup \{(i,a)\}$ est une instanciation complète de $P|_{D_i=\{a,b\}}$ qui vérifie les conditions de la proposition 3.2. S est donc une solution de $P|_{D_i=\{a,b\}}$. □

3.4. Recherche de solution

La proposition 3.3 suggère que la démarche consistant à trouver tout d'abord une solution S' à $P|_{i=a\vee b}$, puis à étendre cette solution à la variable i, est une démarche correcte et complète pour résoudre $P|_{D_i=\{a,b\}}$. Dans le cas où le choix de a ou b ne donne pas de solution, on peut considérer les autres valeurs de D_i, en appliquant une projection de valeurs pour chaque autre paire de valeurs de D_i. On étend ainsi la résolution au problème entier. En procédant de la sorte avec les autres variables du CSP, on effectue une contraction de la recherche en considérant les affectations par paires de valeurs. Enfin, dans le cas où le CSP est consistant, la recherche se terminera en une feuille de l'arbre de recherche où un ensemble de paires de valeurs aura été sélectionné. Il est alors possible (d'après le point (ii) de la preuve), à partir de la dernière paire sélectionnée, de construire une solution en ajoutant à chaque fois, une valeur consistante avec la solution partielle courante.

4. Algorithme de résolution par Projection de Valeurs

L'algorithme que nous proposons est, à la base, une procédure de recherche en profondeur d'abord qui réduit les CSPs en effectuant des projections de valeurs, d'où sa dénomination : algorithme par *Projection de Valeurs* (VP). Il opère sur le graphe d'incohérences μ_P du CSP à résoudre.

4.1. Algorithme VP

VP procède de la manière suivante. Une phase de projection de valeurs a tout d'abord lieu : variable par variable, on considère les valeurs par paire, jusqu'à ce que toutes les valeurs soient éliminées. C'est la phase de "descente", réalisée par la procédure *ProjectionDeValeurs*. Puis, si l'algorithme parvient à traiter toutes les variables du problème, alors un traitement supplémentaire est nécessaire pour obtenir une solution. En effet, pour chaque paire retenue lors de la descente, il reste à faire le choix entre les deux valeurs de chacune des variables. C'est la phase de "remontée", réalisée par la procédure *Remontee*.

Algorithme 4.1. VP(X, D, μ_P)

1. **Début**
2. $I \leftarrow$ProjectionDeValeurs$(X, D, \mu_P, \emptyset)$
3. **si** $I \neq \emptyset$ **alors**
4. $S \leftarrow$Remontée(I, μ_P)
5. **Afficher**(S)
6. **fin-si**
7. **fin**

ProjectionDeValeurs choisit, à chaque nœud de l'arbre de recherche, une variable non encore instanciée i. Les valeurs de D_i sont considérées par paire. Dans le cas où D_i contient un nombre impair de valeurs, la dernière valeur sera prise deux fois. La projection de valeurs aura dans ce cas les mêmes effets qu'un filtrage effectué par *Forward-Checking*.

La projection de valeurs proprement dite comporte deux étapes qui sont effectuées par la procédure *ProjeterValeurs*. Tout d'abord, les valeurs de $N(i, ab)$ sont supprimées du graphe d'incohérences, ce qui demande un nombre d'opérations élémentaires de l'ordre de $O(nd)$, où d désigne le nombre de valeur par variable. La deuxième étape consiste à ajouter au graphe d'incohérences, les arêtes de $E(i, a \times b)$. Cette étape nécessite $O(md^2)$ opérations élémentaires.

Procédure 4.2. ProjectionDeValeurs(X, D, μ_P, I)

1. **Début**
2. **si** $X = \emptyset$ **alors**
3. **retourner**(I)
4. **sinon**
5. *Choisir* $i \in X$
6. **pour chaque** $\{a, b\} \subseteq D_i$ **faire**
7. ProjeterValeurs$((i, a), (i, b), D, \mu_P)$
8. $I' \leftarrow$ProjectionDeValeurs$(X - \{i\}, D, \mu_P, I \cup \{\{(i, a), (i, b)\}\})$
9. **si** $I' \neq \emptyset$ **alors**
10. **retourner**(I')
11. **fin-si**
12. Restaurer(μ_P, D)
13. **fin-faire**

14. **retourner**($\emptyset$)
15. **fin-si**
16. **fin**

Lors de la phase de remontée, et comme le suggère la proposition 3.3, les variables sont considérées dans l'ordre inverse de leur instanciation dans la phase de descente. L'algorithme commence donc par la dernière variable instanciée. Une des deux valeurs possibles de cette variable est choisie et toutes les valeurs incompatibles avec cette dernière sont éliminées. L'algorithme procède de la même manière avec l'avant dernière variable, et ainsi de suite jusqu'à remonter à la première variable instanciée. A ce stade, l'algorithme aura affecté une valeur unique à chaque variable du CSP. On obtient ainsi une solution.

Procédure 4.3. Remontée(I, μ_P)

1. **Début**
2. $S \leftarrow \emptyset$
3. **tant que** $I \neq \emptyset$ **faire**
4. $\{(i,a),(i,b)\} \leftarrow$ *Dépiler*(I)
5. **si** $S \cup \{(i,a)\}$ *satisfait toutes les contraintes* **alors**
6. $S \leftarrow S \cup \{(i,a)\}$
7. **sinon**
8. $S \leftarrow S \cup \{(i,b)\}$
9. **fin-si**
10. **fin-faire**
11. **retourner**(S)
12. **fin**

4.2. Cadre de l'expérimentation de VP

L'expérimentation a porté sur des CSPs aléatoires de diverses tailles. Un CSP aléatoire est caractérisé par quatre paramètres n, d, p_1 et p_2, où n est le nombre de variables du problème, d est le nombre de valeurs par domaine de valeurs, p_1 est la probabilité d'existence d'une contrainte entre deux variables et p_2 donne la proportion approximative des couples de valeurs incompatibles dans la définition des contraintes. Pour des valeurs fixées de n, d et p_1, il existe une zone de variation de p_2 où les CSPs sont particulièrement difficiles à résoudre [17]. Les instances de problèmes considérées dans l'expérimentation ont été choisies dans cette zone critique. Pour chaque instance de CSP aléatoire, (i.e. pour chaque quadruplet (n, d, p_1, p_2)), on génère 100 problèmes, considérant que 100 est un nombre suffisant pour atténuer l'effet des perturbations dues à des cas particuliers. Le générateur utilisé est du type désigné par modèle A dans la littérature [14].

(n, d, p_1, p_2)	% pb. solubles	FC	VP
(60,5,0.2,0.205)	46	28	16
(50,5,0.5,0.11)	41	26	19
(50,5,1,0.055)	84	46	33
(50,10,0.1,0.53)	39	164	54
(20,20,0.5,0.45)	56	30	283
(80,5,0.1,0.26)	81	452	14
(40,10,0.5,0.2)	35	783	1430
(80,5,0.5,0.067)	71	3298	1227

FIGURE 3. Quelques résultats obtenus avec FC et VP. Les temps de calcul sont donnés en secondes.

4.3. Résultats de l'expérimentation de VP

L'algorithme VP a été comparé à l'un des algorithmes de résolution les plus communément utilisés: *Forward-Checking* (FC). On a choisi de se limiter à une comparaison avec FC car les performances de MAC sont très liées à la façon dont il est implémenté.

On a d'abord appliqué l'algorithme VP à un ensemble de CSPs aléatoires différents. La comparaison des performances de cet algorithme avec celles obtenues avec FC montre qu'une résolution utilisant la technique de projection de valeurs à chaque point de choix, n'est meilleure que la résolution par FC que lorsque le graphe de contraintes est peu dense (p_1 faible), ou si le rapport n/d est grand (voir figure 3). Il semble que pour des problèmes présentant un graphe de contraintes dense ou de faible rapport n/d, l'effet de la propagation de contraintes effectué par FC est assez efficace, et permet d'obtenir de résultats meilleurs que ceux obtenus avec VP.

Cette remarque suggère d'alterner une résolution classique avec une résolution utilisant la technique de projection de valeurs selon une démarche similaire à celle utilisée dans le cadre d'une comparaison entre FC et MAC [3].

5. Approche alternant projection de valeurs et propagation de contraintes

Souhaitant alterner des propagations locales de contraintes et des projections de valeurs, il faut déterminer, en un point de choix, si l'on opère une projection de valeurs ou si l'on adopte une démarche classique. Pour effectuer ce choix, il est nécessaire d'introduire un critère permettant d'opter pour l'une ou pour l'autre des deux stratégies, dans le but d'obtenir les meilleures performances possibles. Deux critères ont été particulièrement étudiés: la profondeur du point de choix, et le nombre de conflits; d'autres : le paramètre κ [2] et l'estimateur de Knuth [6] ont été envisagés.

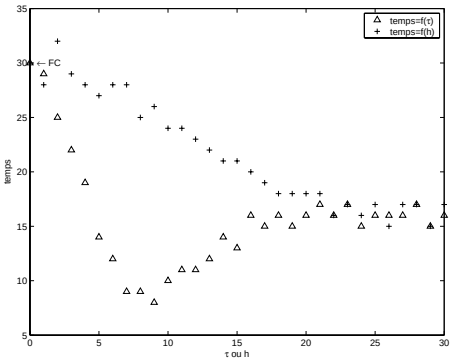

FIGURE 4. Variation du temps de calcul en fonction du nombre de conflits (τ) ou de la profondeur dans l'arbre de recherche (h) pour $(60, 5, .2, .205)$. Les valeurs de τ et de h indiquent respectivement le nombre de conflits et la profondeur au delà desquels l'algorithme bascule de VP à FC. Les points ayant pour absisse 0 donnent la performance de Forward-Checking.

5.1. CRITÈRE D'ALTERNANCE : LE NOMBRE DE CONFLITS

Les meilleurs résultats ont été obtenus en utilisant comme critère le nombre de conflits. Si i est la variable à instancier et a la valeur de D_i choisie pour être affectée à i, alors le critère de choix est le nombre de conflit de la valeur (i, a) (i.e. le nombre d'élément de $N(i, a)$). Si le nombre de conflit de (i, a) est inférieur à une valeur prédéterminée qu'on note τ, alors une projection de valeurs sera effectuée, sinon l'algorithme effectuera une simple propagation locale de contraintes, comme celle effectuée par FC. On appelle l'algorithme générique ainsi obtenu FC+VP, dans lequel les projections de valeurs sont contrôlées par la valeur de τ.

Les résultats obtenus sur une instance de CSP aléatoire avec le critère du nombre de conflits sont présentés à la figure 4. On notera sur la figure 4 qu'un autre critère étudié, celui de la profondeur de choix h, ne donne pas de meilleurs résultats. On constate qu'avec le nombre de conflits, les performances se dégradent si l'on effectue des projections au delà d'une valeur critique. Nous avons cherché alors à déterminer cette valeur critique de τ.

Pour étudier l'effet du paramètre τ sur les performance de FC+VP, on a mené une expérimentation sur diverses instances de CSPs aléatoires, et mesuré le nombre de nœuds développés dans l'arbre de recherche en faisant varier la valeur de τ. On constate, tout comme pour le temps de calcul étudié dans la figure 4, que le nombre de nœuds développés diminue jusqu'à ce que τ atteigne une valeur critique au delà de laquelle on observe une augmentation du nombre de nœuds développés.

Une variation du même type est observée si on fait varier les différents paramètres n, d et p_1 (voir figure 5). La variation du nombre de nœuds développés en fonction de τ suit donc une fonction qui possède un minimum global unique. Nous avons

chercher à déterminer la valeur de τ qui donne le nombre de nœuds développés minimum. Cette valeur particulière, notée τ^*, devra être estimée avant la résolution de chaque problème afin d'obtenir les meilleures performances possibles. L'estimation de τ^* est déterminée en fonction des quatre paramètres n, d, p_1 et p_2, avant la résolution du problème à traiter.

Une étude de la variation de τ^* en fonction des paramètres n, d, p_1 et p_2 a été effectuée, et les résultats montrent que les relations entre τ^* et les différents paramètres ne sont pas toutes linéaires (voir figure 6). En effet, en maintenant constant n et d et en faisant varier p_1 de 0.1 à 1, la variation de τ^* suit une évolution hyperbolique. Cette expérimentation a été répétée avec plusieurs valeurs de n et l'on obtient une évolution similaire; de même, si on fait varier n en maintenant constant d et p_1. Par contre, en fixant les valeurs de n et de p_1 et en faisant varier d, on constate que τ^* croît linéairement. Enfin, si on fait varier p_2 dans la zone critique tout en gardant constant les trois autres paramètres, τ^* reste approximativement constante.

Nous avons opté pour une approximation de τ^* par réseau de neurônes multi-couches, pour la raison essentielle suivante : la variation de τ^* n'est pas une fonction linéaire des paramètres n et p_1. Or, les réseaux multi-couches constituent un moyen d'approximation efficace pour des fonctions non-linéaires [16]. Le réseau que nous utilisons est un réseau multi-couches avec une seule couche cachée. La couche d'entrée contient trois neurônes: un neurône pour chacun des paramètres n, d et p_1. La couche cachée en contient trois également. Ce nombre s'est révélé suffisant pour l'obtention de résultats satisfaisants. La couche de sortie contient un seul neurône donnant en sortie la valeur estimée de τ^*. L'algorithme d'apprentissage est celui de la rétropropagation du gradient. La base d'exemples sur laquelle on a effectué l'apprentissage contient 130 exemples. Les problèmes considérés ont une taille variant de 20 à 100 avec des domaines de valeurs de 5, 10 ou 20, la valeur de p_1 varie entre 0.08 et 1, les valeurs de p_2 variant dans la zone critique.

5.2. Résultats expérimentaux

Pour mettre en évidence la précision des approximations données par le réseau de neurônes, ainsi que l'apport de l'algorithme FC+VP, on a reporté à la figure 7 les résultats obtenus en appliquant FC+VP contrôlé respectivement par τ^* et $\hat{\tau}^*$ la valeur approchée de τ^* calculée par le réseau de neurônes. Les mesures portent sur le nombre de nœuds développés. Les performances obtenues avec FC seul sont également présentées à la figure 7. Ces tests ont été effectués sur des instances de CSPs qui ne font pas partie de la base d'exemples. On constate, en examinant le tableau de la figure 7, que les performances obtenues avec $\hat{\tau}^*$ sont assez proches de celles obtenues avec τ^*.

Afin de mieux comparer les performances respectives de FC et de FC+VP, on a ensuite étudié le rapport entre leurs performances, mesurées par le nombre de nœuds développés et aussi par le temps de calcul (voir figure 8). Ces rapports expriment le gain en nombre de nœuds développés ou en temps de calcul dû à la

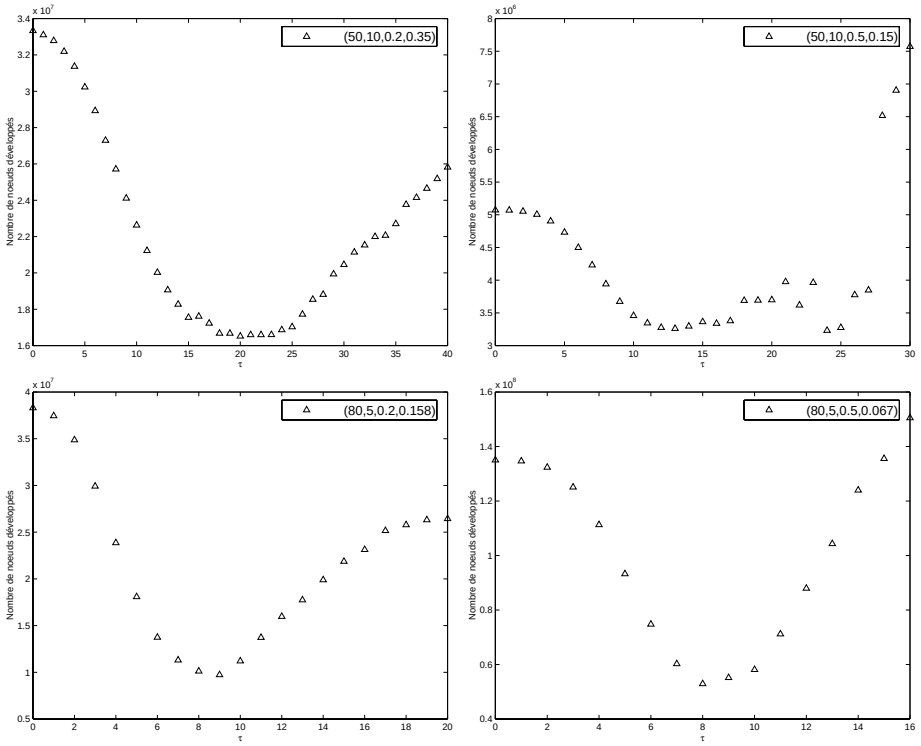

FIGURE 5. Variation du nombre de nœuds développés en fonction du paramètre de contrôle des projections τ. Résultats obtenus sur des l'instances de CSPs aléatoires difficiles. Les points d'abscisse 0 indiquent les performances de FC.

projection de valeurs. On se propose d'étudier la variation de ce gain en fonction du rapport n/d et du paramètre p_1. Notons que le rapport n/d a été préféré à n et d pris séparément car on a constaté que pour des valeurs fixes de p_1, le gain ne varie pratiquement pas si n/d reste constant. Notons aussi que la variation de p_2 n'influe pas sur l'apport de la méthode, si on se limite à des valeurs de p_2 prises dans la zone critique. Les graphes de la figure 8 montrent que l'apport de la méthode est plus significatif pour des valeurs faibles de p_1 et pour des rapports n/d grands.

6. CONCLUSION

Dans cet article nous avons proposé une nouvelle technique de résolution de CSPs qui effectue des instanciations de variables de manière non-déterministe, et qui portent sur deux valeurs à la fois. Cette technique, que nous avons appelée projection de valeurs (VP), est une alternative à la démarche classique adoptée

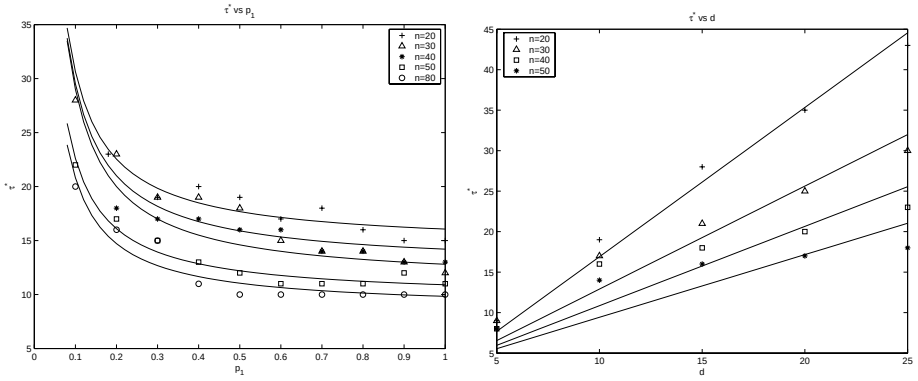

FIGURE 6. Variation de τ^* en fonction des paramètres n, d et p_1. Pour le premier graphique, la valeur de d à été fixée à 10. Pour le second graphique, la valeur de p_1 à été fixée à 0.5.

(n, d, p_1, p_2)	τ^*	$\hat{\tau}^*$	FC+VP(τ^*)	FC+VP($\hat{\tau}^*$)	FC
(20,15,0.5,0.42)	28	24	429	431	519
(20,15,0.7,0.32)	22	21	912	913	1059
(20,20,0.5,0.45)	33	32	1016	1021	1210
(20,20,0.7,0.35)	27	28	3429	3431	3931
(40,10,0.15,0.48)	20	17	680	769	2475
(40,10,0.3,0.3)	17	15	5594	5698	8894
(40,10,0.5,0.19)	15	13	19845	20360	27466
(60,5,0.3,0.145)	9	12	2057	2394	4711
(60,5,0.5,0.088)	8	9	2601	2653	5317
(80,5,0.3,0.11)	8	11	21868	30038	59904
(80,5,0.5,0.068)	9	10	62860	67670	157685

FIGURE 7. Valeurs optimales (τ^*) et valeurs calculées par réseau de neurônes ($\hat{\tau}^*$) du paramètre de contrôle des projections. Performances (en milliers de nombres de nœuds développés) obtenues par des exécutions contrôlées par τ^* et par des exécutions contrôlées par $\hat{\tau}^*$. En dernière colonne figurent les performances obtenues par FC.

par la plupart des algorithmes existants et qui consiste à utiliser une seule valeur dans chaque instanciation. En intégrant cette technique à l'algorithme Forward-Checking, on a obtenu un nouvel algorithme qui alterne instanciation classique et projection de valeurs. Le choix entre une instanciation classique et une projection de valeurs est effectué à chaque nœud de l'arbre de recherche. Ce choix est contrôlé par un paramètre unique. Pour le calcul d'une valeur approchée de ce paramètre de contrôle, nous avons conçu un réseau de neurônes qui a conduit à des approximations satisfaisantes. L'algorithme ainsi obtenu a été testé sur des CSPs aléatoires et les résultats des tests ont montré que l'apport de la projection de valeurs est significatif.

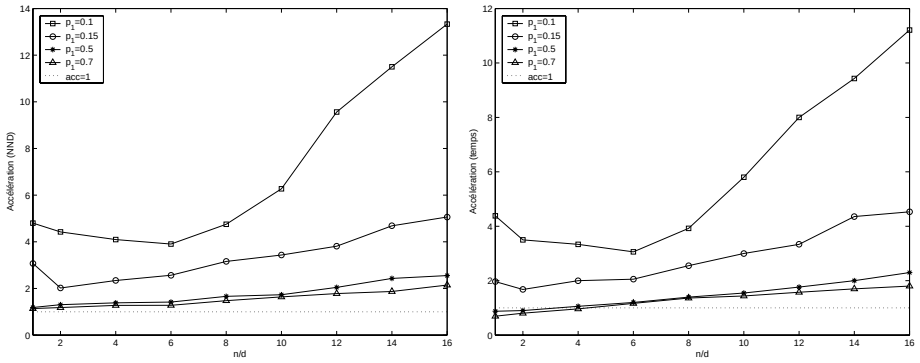

FIGURE 8. Variation de l'accélération exprimée par le rapport des nombres de nœuds développés par les deux algorithmes (graphique de gauche). Variation de l'accélération exprimée par le rapport des temps de calcul consommés par les deux algorithmes: FC et FC+VP (graphique de droite). Le trait en pointillés représente un rapport de 1. En absisse, on fait varier le rapport n/d. Les différents points correspondent à des valeurs différentes de p_1.

Nous estimons que, dans le cadre de la résolution de problèmes de satisfaction et d'optimisation de contraintes (CSOP), l'intégration de la technique de projection de valeurs à un algorithme comme le *Branch and Bound* pourra sensiblement en améliorer les performances.

REFERENCES

[1] I. Eidhammer, I. Jonassen, S. H. Grindhaug, D. Gilbert, M. Ratnayake, A Constraint Based Structure Description Language for Biosequences, *Constraints*, **6**, No. 2/3 (2001) 173-200.

[2] I. P. Gent, E. MacIntyre, P. Prosser and T. Walsh, The Constrainedness of Search, In *Proceedings AAAI-96*, AAAI Press, **1** (1996) 246-252.

[3] I.P. Gent and P. Prosser, Inside MAC and FC, APES Research Group Report APES-20-2000, (2000).

[4] S. A. Grant, Phase Transition Behaviour in Constraint Satisfaction Problems, Ph.D. Thesis, The University of Leeds, GB (1997).

[5] R. Haralick and G. Elliot, Increasing tree search efficiency for constraint satisfaction problems, *Artificial Intelligence*, **14** (1980) 263-313.

[6] D. Knuth, Estimating the efficiency of backtrack programs, *Mathematics of Computation*, **29** No. 129 (1997) 121-136.

[7] L. Lovàsz and M. D. Plummer, *Matching Theory*, North-Holland, Amsterdam (1986).

[8] P. Meseguer, Interleaved depth-first search, In *Proceedings of the 15th IJCAI*, (1997) 1382-1387.

[9] P. Meseguer and T. Walsh, Interleaved and discrepancy based search, In *Henri Prade, editor, Proceedings of the 13 th European Conference on Artificial Intelligence*, Brighton, UK, John Wiley & Sons, Inc. (1998) 239-243.

[10] C. Manino and A. Sassano, Edge projection and the maximum stable set problem, *Clique coloring and satisfiability: Second DIMACS Implementation Challenge*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, (1995).

[11] W. Naanaa, Résolution de problèmes de satisfaction de contraintes intégrant la symétrie et la flexibilité, Ph.D. Thesis. University Claude Bernard - Lyon I, France (1996).

[12] W. Naanaa and S. Pimont, Handling structures and ambiguous constraints in constraint satisfaction problems, *Journal of Experimental and Theoretical Artificial Intelligence*, **10** (1998) 91-102.

[13] J.-M. Nuzillard, W. Naanaa, S. Pimont, Applying the Constraint Satisfaction Problem Paradigm to Structure Generation, *Journal of Chemical Information and Computer Sciences*, **35** (1995) 1068-1073.

[14] E. M. Palmer, *Graphical Evolution*, Wiley, New York, (1985).

[15] D. Sabin and E. C. Freuder, Contradicting conventional wisdom in constraint satisfaction, In *Proceedings of the 11th European Conference on Artificial Intelligence*, (1994).

[16] L. E. Scales, Introduction to Non-Linear Optimization, New York, Springer-Verlag, (1985).

[17] B. M. Smith, Martin E. Dyer. Locating the phase transition in binary constraint satisfaction problems. *Artificial Intelligence*, **8** (1996) 155-181.

[18] T. Walsh, Depth-bounded discrepancy search, In *Proceedings of the 15th IJCAI*, (1997).

[19] M. Yokoo and K. Hirayama, Frequency Assignment for Cellular Mobile Systems Using Constraint Satisfaction Techniques, In *Proceedings of the IEEE Annual Vehicular Technology Conference (VTC2000-Spring)*, (2000).