Identification et exploitation d'états partiels

Christophe Lecoutre Sébastien Tabary Vincent Vidal
CRIL – CNRS UMR 8188,
Université Lille-Nord de France, Artois
rue de l'université, SP 16, F-62307 Lens
{lecoutre,tabary,vidal}@cril.fr

Résumé

Dans cet article, nous proposons une étude concernant l'identification et l'exploitation des états partiels inconsistants (IPS) dans le cadre de la résolution du problème de satisfaction de contraintes (CSP). Un IPS correspond à un ensemble de "décisions d'appartenance" prises sur un réseau de contraintes et ne menant à aucune solution. Notre étude tend à unifier un certain nombre de travaux récents concernant les nogoods généralisés, le concept de valeurs en échec et la détection de dominance. Nous introduisons un formalisme simple et unificateur et montrons comment la cohérence d'arc généralisée (GAC) peut être établie efficacement sur les contraintes associées aux IPSs tout au long de la recherche. Nous identifions également plusieurs opérateurs qui permettent d'extraire des IPSs à chaque noeud de l'arbre de recherche prouvé comme étant la racine d'un sous-arbre sans solutions.

1 Introduction

Ces dernières années, plusieurs formes d'apprentissage sophistiquées, basées essentiellement sur la détection de conflits, ont été proposées pour la résolution du problème de satisfaction de contraintes (CSP). Il s'agit de travaux concernant la détection de dominance (e.g. [1, 8] pour SBDS et [6, 5] pour SBDD), les nogoods généralisés [10], la notion de valeurs en échec [7, 16, 11] et les états partiels [13, 12]. Ces différentes approches possèdent un certain nombre de points communs qui n'ont pas été clairement établis jusqu'à présent.

Dans cet article[1], nous proposons un cadre unificateur capturant les notions évoquées ci-dessus. Ce cadre est celui revisité des états partiels (inconsistants). Il permet la représentation naturelle et l'exploitation précise et efficace des informations apprises au cours de la recherche. Plus précisément, ces informations sont représentées sous forme de contraintes dites de dominance. La cohérence d'arc (généralisée) est maintenue efficacement sur ces contraintes à l'aide de la structure des "watched literals".

Nous proposons également deux opérateurs d'extraction d'états partiels inconsistants (IPSs) appliqués à chaque noeud de l'arbre de recherche. Un IPS correspond à un ensemble de "décisions d'appartenance" prises sur un réseau de contraintes et ne menant à aucune solution. De manière intuitive, plus la taille d'un IPS est petite (en terme de nombre de décisions) plus sa capacité d'élagage est importante. Le premier opérateur proposé élimine toute décision portant sur une variable dont le domaine courant peut être déduit de l'état partiel. Le second opérateur élimine les décisions portant sur les variables qui ne sont pas impliquées dans la preuve d'incohérence du sous-arbre exploré à partir du noeud courant. Ces deux opérateurs peuvent être combinés.

2 Formalisme

Un réseau de contraintes (CN) P est composé d'un ensemble fini de n variables, noté $vars(P)$, et d'un ensemble fini de e contraintes, noté $cons(P)$. Chaque variable x possède un domaine associé, noté $dom(x)$, qui contient l'ensemble fini des valeurs qui peuvent être assignées à x. Chaque contrainte c implique un ensemble de variables, appelé la *portée* de c et noté $scp(c)$. Chaque contrainte est définie par une relation, notée $rel(c)$, qui contient l'ensemble des tuples autorisés pour les variables impliquées dans c. Une contrainte *binaire* implique exactement 2 variables et un CN binaire implique seulement des contraintes binaires. Une contrainte c de P est *universelle* (entailed) si et seulement si tous les tuples construits à partir des domaines des variables de $scp(C)$ sont autorisés par c.

[1] Une partie de cet article est fondée sur [12], non publié en français.

Dans ce papier, nous considérerons comme étant donnés un CN initial P^{init} et un CN courant P dérivé de P^{init} en réduisant potentiellement les domaines des variables. Le domaine initial d'une variable x est noté $dom^{init}(x)$ tandis que le domaine courant est noté $dom^P(x)$ ou plus simplement $dom(x)$. Pour chaque variable x, nous avons toujours $dom(x) \subseteq dom^{init}(x)$ et nous décrivons ce phénomène par $P \preceq P^{init}$. Une valeur (courante) de P est un couple (x, a) avec $x \in vars(P)$ et $a \in dom(x)$.

Une solution d'un CN correspond à l'assignation d'une valeur à chaque variable telle que toutes les contraintes soient satisfaites. Un CN P est dit être *satisfaisable* si et seulement si il admet au moins une solution. Si le domaine d'une variable de P est vide, P est trivialement insatisfaisable, et ceci est noté $P = \bot$. Le problème de satisfaction de contraintes (CSP) est la tâche NP-difficile de déterminer si un CN donné est satisfaisable ou pas. Une instance CSP est définie par un CN qui est résolu soit en trouvant une solution ou alors en prouvant son insatisfaisabilité. Pour résoudre une instance CSP, on peut utiliser un algorithme de recherche en profondeur d'abord équipé d'un mécanisme de retours-arrières. A chaque étape de la recherche, l'assignation d'une variable est effectuée suivie par une étape de filtrage appelée propagation de contraintes et basée sur des propriétés comme la cohérence d'arc généralisée (GAC). Pour une définition de GAC, appelée cohérence d'arc (AC) lorsque les contraintes sont binaires, voir par exemple [4].

3 Décisions

Les décisions représentent la notion élémentaire utilisée dans cet article.

Définition 1. *Une décision positive (resp. négative) δ est une restriction de la forme $x = a$ (resp. $x \neq a$), où x est une variable et $a \in dom^{init}(x)$. Une décision δ sur un CN P est une décision positive ou négative impliquant une valeur de P.*

En d'autres termes, une décision positive est une *assignation de variable* et une décision négative est une *réfutation de valeur*. Lorsque les décisions sont prises sur un CN P, nous obtenons une nouveau CN $P|_\Delta$ qui correspond au plus grand CN P' tel que $P' \preceq P$ et $P'|_\Delta = P'$. Pour n'importe quel ensemble de décisions Δ, $vars(\Delta)$ représente l'ensemble des variables apparaissant dans les décisions de Δ. Tous les ensembles de décisions ne sont pas forcément bien-formés ; ceux menant systématiquement à un échec ne sont pas pertinents. Un ensemble de décisions Δ est dit être *bien-formé* si et seulement si il existe au moins un CN P tel que $vars(P) = vars(\Delta)$ et $P|_\Delta \neq \bot$.

Un nogood *standard* de P est un ensemble Δ de décisions positives sur P tel que $P|_\Delta$ est insatisfaisable. En considérant à la fois des décisions positives et négatives comme dans [6, 10], on obtient une généralisation des nogoods standards, appelée *nogoods généralisés* :

Définition 2. *Un ensemble de décisions Δ sur un CN P est un nogood généralisé de P si et seulement si $P|_\Delta$ est insatisfaisable.*

Clairement, un nogood (standard) est généralisé mais l'opposé n'est pas nécessairement vrai. Un nogood généralisé peut représenter un nombre exponentiel de nogoods standards [10]. On peut utiliser la notion de décisions d'appartenance pour manipuler différentes classes d'équivalence de nogoods.

Définition 3. *Une décision d'appartenance δ est une restriction de la forme $x \in S_x$, où x est une variable et $\emptyset \subset S_x \subseteq dom^{init}(x)$; δ est stricte si et seulement si $S_x \subset dom^{init}(x)$. Une décision d'appartenance δ sur un CN P est une décision d'appartenance $x \in S_x$ tel que $x \in vars(P)$; δ est stricte sur P si et seulement si $S_x \subset dom^P(x)$ et est valide sur P si et seulement si $S_x \subseteq dom^P(x)$.*

Un ensemble de décisions d'appartenance Δ est bien-formé si et seulement si chaque variable apparaît au plus une fois dans Δ.

Proposition 1. *Pour tout ensemble Δ bien-formé de décisions (positive et/ou négative) sur un CN P, il existe un unique ensemble bien-formé Δ^m de décisions d'appartenances strictes sur P tel que $P|_\Delta = P|_{\Delta^m}$.*

La preuve est omise. Par exemple, si $dom^{init}(x) = dom^{init}(y) = dom^{init}(z) = \{a, b, c\}$ et $\Delta = \{x = a, y \neq b, y \neq c, z \neq b\}$, alors nous avons $\Delta^m = \{x \in \{a\}, y \in \{a\}, z \in \{a, c\}\}$.

4 États partiels inconsistants

Nous introduisons maintenant le concept d'état partiel.

Définition 4. *Un état partiel Δ est un ensemble de décisions d'appartenance ; Δ est strict si et seulement si chaque décision d'appartenance de Δ est stricte. Un état partiel Δ sur un CN P est un ensemble bien-formé de décisions d'appartenance valides sur P ; Δ est strict sur P si et seulement si chaque décision d'appartenance de Δ est stricte sur P.*

Un état partiel strict impose que chaque décision d'appartenance constitue une réelle restriction (cf. Définition 3). Nous notons $vars(\Delta)$ l'ensemble des variables apparaissant dans les décisions d'appartenance de Δ. Si Δ est un état partiel et si $(x \in S_x) \in \Delta$, nous notons S_x par $dom^\Delta(x)$. Il est important de noter que si P est un CN impliquant x et tel que $dom^\Delta(x) \not\subseteq dom^P(x)$, Δ ne peut pas être un état partiel sur P ; chaque décision d'appartenance d'un état partiel sur P doit être valide sur P.

P	Δ_1	Δ_3
$v \in \{a,b,c,d\}$ $w \in \{a,b,c,d\}$ $x \in \{a,b,c,d\}$ $y \in \{a,b,c,d\}$ $z \in \{a,b,c,d\}$	$w \in \{b,c\}$ $z \in \{a,b,d\}$ Δ_2 $v \in \{b,d\}$ $x \in \{c\}$ $z \in \{a,b\}$	$v \in \{a,b,c,d\}$ $w \in \{b,c\}$ $z \in \{a,b,d\}$

FIG. 1 – L'état courant d'un CN P et trois états partiels sur P. Contrairement à Δ_3, Δ_1 et Δ_2 sont stricts sur P.

$$P \xrightarrow{\Delta} P|_\Delta$$

P	Δ	$P\vert_\Delta$
$v \in \{a,b,c,d\}$ $w \in \{a,b,c,d\}$ $x \in \{a,b,c,d\}$ $y \in \{a,b,c,d\}$ $z \in \{a,b,c,d\}$	$w \in \{b,c\}$ $z \in \{a,b,d\}$	$v \in \{a,b,c,d\}$ $w \in \{b,c\}$ $x \in \{a,b,c,d\}$ $y \in \{a,b,c,d\}$ $z \in \{a,b,d\}$

FIG. 2 – La restriction $P|_\Delta$ d'un CN P par un état partiel Δ sur P. Les états courants sont donnés pour P et $P|_\Delta$.

Il existe un état partiel immédiat pour n'importe que CN P. Cet état partiel est construit en prenant en compte toutes les variables de P. Plus précisément, *l'état courant* d'un CN P est l'état partiel $\{(x \in dom^P(x)) \mid x \in vars(P)\}$ sur P. La Figure 1 présente l'état courant d'un CN P, et trois états partiels Δ_1, Δ_2 et Δ_3 sur P. Δ_1 et Δ_2 sont stricts sur P, mais Δ_3 n'est pas strict sur P car $dom^P(v) = dom^{\Delta_3}(v)$. Nous avons $vars(\Delta_1) = \{w, z\}$, $dom^{\Delta_1}(w) = \{b, c\}$ et $dom^{\Delta_1}(z) = \{a, b, d\}$.

La restriction $P|_\Delta$ d'un CN P par un état partiel Δ sur P est le CN obtenu à partir P en restreignant le domaine de chaque variable $x \in vars(\Delta)$ à $dom^\Delta(x)$; pour chaque $x \in vars(\Delta)$, nous avons $dom^{P|_\Delta}(x) = dom^\Delta(x)$ et pour chaque $x \notin vars(\Delta)$, nous avons $dom^{P|_\Delta}(x) = dom^P(x)$.

Le réseau restreint $P|_\Delta$ est plus petit ($\preceq$) que P. Plus précisément, si $\Delta = \emptyset$, nous avons $P|_\Delta = P$ et si Δ est strict sur P et non vide, nous avons $P|_\Delta \prec P$. La figure 2 illustre la restriction d'un CN par un état partiel.

Quand une variable x n'apparaît pas dans un état partiel Δ, cela veut simplement dire que le domaine de x est considéré comme inchangé par Δ. En pratique, il est alors suffisant de ne manipuler que des états partiels stricts qui ont l'avantage d'être plus petits. Un état partiel strict peut être naturellement dérivé de n'importe quel état partiel.

Définition 5. *Soient P un CN et Δ un état partiel sur P. $\Delta^{s(P)}$ représente l'ensemble des décisions d'appartenance de Δ qui sont strictes sur P.*

Par exemple pour la figure 1, $\Delta_1 = \Delta_3^{s(P)}$. De manière importante, Δ et $\Delta^{s(P)}$ sont équivalents car $P|_\Delta = P|_{\Delta^{s(P)}}$. Un état partiel Δ sur un CN P est dit être inconsistant si le réseau défini comme la restriction de P sur Δ est insatisfaisable.

Définition 6. *Soient P un CN et Δ un état partiel sur P. Δ est un état partiel inconsistant (IPS) sur P, si et seulement si $P|_\Delta$ est insatisfaisable.*

Nous obtenons un IPS strict en ne tenant pas compte des décisions d'appartenance qui ne sont pas strictes.

Proposition 2. *Soient P un CN et Δ un IPS sur P. $\Delta^{s(P)}$ est un IPS sur P, ou IPS strict.*

La preuve est immédiate. La domination est définie comme suit.

Définition 7. *Soient P un CN et Δ un état partiel tel que $vars(\Delta) \subseteq vars(P)$. Δ domine P si et seulement si $\forall x \in vars(\Delta)$, $dom^P(x) \subseteq dom^\Delta(x)$.*

Notons que Δ n'est pas nécessairement un état partiel *sur* P. Lorsqu'un état partiel Δ domine un CN P, nous avons $P|_\Delta = P$. Par définition, un état partiel (non vide) strict Δ sur un CN P ne peut pas dominer P. Cette notion de dominance est utilisée pour les CNs strictement plus petit que P comme montré dans la figure 3.

$$P' \prec_d P \succ_d P''$$

P'	P	Δ	P''
$v \in \{a,b,c,d\}$ $w \in \{b,c\}$ $x \in \{a,b,c,d\}$ $y \in \{a,d\}$ $z \in \{b\}$	$v \in \{a,b,c,d\}$ $w \in \{a,b,c,d\}$ $x \in \{a,b,c,d\}$ $y \in \{a,b,c,d\}$ $z \in \{a,b,c,d\}$	$w \in \{b,c\}$ $z \in \{a,b,d\}$	$v \in \{a,b,c,d\}$ $w \in \{b,c\}$ $x \in \{a,b,c,d\}$ $y \in \{a,d\}$ $z \in \{a,b,c,d\}$

FIG. 3 – Deux CNs P' et P'' (strictement) plus petits que P et un état partiel Δ sur P. P' est dominé par Δ. P'' est presque dominé par Δ.

La proposition suivante est au coeur du raisonnement basé sur les états par détection de dominance.

Proposition 3. *Soient P et P' deux CNs tels que $P' \prec P$, et Δ un IPS sur P. Si Δ domine P' alors P' est insatisfaisable.*

Démonstration. Si $P' \prec P$, nous avons par monotonie $P'|_\Delta \preceq P|_\Delta$. Comme P' est dominé par Δ, $P'|_\Delta = P'$. On a alors $P' \preceq P|_\Delta$. Δ est un IPS sur P, ce qui veut dire que $P|_\Delta$ est insatisfaisable. Nous concluons que P' est insatisfaisable. $\square$

Les états partiels sont dits être *globaux* lorsqu'ils sont définis sur le problème initial P^{init} ; dans le cas contraire, ils sont dits être *locaux*. Les IPSs globaux sont valides pour la totalité de l'espace de recherche tandis que les IPSs locaux ne sont utilisables que dans certains sous-arbres de recherche. Un IPS global peut toujours être extrait d'une impasse interne (noeud racine d'un sous-arbre sans solutions) de l'arbre de recherche construit par un algorithme de recherche avec retours-arrières : si v est une impasse interne et $P = cn(v)$ est le CN associé à v, l'état courant de P est un IPS lui-même. Cependant, cet IPS ne peut pas être exploité par la suite, excepté si des redémarrages sont effectuées ou si l'on exploite des symétries. Dans les sections suivantes, nous présentons plusieurs approches pour construire des IPSs pertinents.

5 Propager les IPSs

Dans le contexte d'un algorithme de recherche avec retours-arrières, la proposition 3 peut être exploitée pour élaguer des noeuds dominés par des IPSs précédemment identifiés. Plus précisément, à chaque fois que l'algorithme de recherche génère un nouveau noeud v, on peut vérifier si $cn(v)$, le CN associé à v, est dominé par un IPS préalablement enregistré. Deux mécanismes d'élagage peuvent être employés : soit v est directement rejeté parce qu'il est dominé, soit certains domaines des variables de $cn(v)$ sont filtrés pour garantir que la dominance ne peut apparaître plus tard. Pour clarifier cela, nous introduisons une version affaiblie de la dominance :

Définition 8. *Soient P un CN et Δ un état partiel tel que $vars(\Delta) \subseteq vars(P)$. Δ domine presque P si et seulement si il existe une décision d'appartenance $x \in S_x$ dans Δ telle que $dom^P(x) \not\subseteq dom^\Delta(x)$, $dom^P(x) \cap dom^\Delta(x) \neq \emptyset$ et $\Delta \setminus \{x \in S_x\}$ domine P.*

Lorsque la dominance ou la presque dominance implique un IPS, cela devient intéressant. Si un CN P est dominé par un IPS alors P est nécessairement insatisfaisable ; ceci est une conséquence de la proposition 3. D'un autre coté, un CN P qui est presque dominé par un IPS Δ peut être simplifié en supprimant toutes les valeurs qui pourraient rendre le CN dominé. En effet, si x est la seule variable de P telle que $dom^P(x) \not\subseteq dom^\Delta(x)$, nous pouvons déduire de manière certaine que chaque valeur de $dom^P(x) \cap dom^\Delta(x)$ est inconsistante. Ceci est appelé la cohérence 1-dominance dans [15]. Par exemple, le CN P'' de la figure 3 est presque dominé par Δ. Si Δ est un IPS sur P, les valeurs a, b et d du domaine de z peuvent être supprimées sans perte de solutions.

A strictement parler, comme montré ci-dessous, la cohérence 1-dominance n'est pas exactement une nouvelle cohérence si l'on considère que chaque IPS est représenté par une contrainte de *dominance*. Si Δ est un IPS, alors une contrainte de dominance c_Δ peut être construite telle que sa portée est $vars(\Delta)$ et sa relation interdit tous les tuples qui satisfont simultanément les décisions d'appartenance de Δ. Par exemple, soient x, y et z trois variables telles que $dom^{init}(x) = dom^{init}(y) = dom^{init}(z) = \{a, b, c\}$ et $\Delta = \{x \in \{a, b\}, y \in \{c\}, z \in \{b, c\}\}$ un IPS. La contrainte de dominance ternaire correspondant à Δ est c_Δ telle que $scp(c_\Delta) = \{x, y, z\}$ et $rel(c_\Delta) = dom^{init}(x) \times dom^{init}(y) \times dom^{init}(z) \setminus \{a, b\} \times \{c\} \times \{b, c\}$.

Les contraintes de dominance peuvent être exprimées en intension en appliquant simplement la loi de De Morgan sur les IPSs qui sont alors vus comme des conjonctions logiques de décisions d'appartenance. Par exemple, à partir de l'IPS $\Delta = \{x \in \{a, b\}, y \in \{c\}, z \in \{b, c\}\}$, on peut formuler la contrainte de dominance $c_\Delta : x \notin \{a, b\} \vee y \notin \{c\} \vee z \notin \{b, c\}$, ou de manière équivalente $c_\Delta : x \in \{c\} \vee y \in \{a, b\} \vee z \in \{a\}$ si $dom^{init}(x) = dom^{init}(y) = dom^{init}(z) = \{a, b, c\}$. De manière générale, pour chaque décision $x \in S_x$ apparaissant dans Δ, la décision complémentaire $x \in dom^{init}(x) \setminus S_x$ apparaît dans (l'expression du prédicat de) la contrainte de dominance c_Δ. On se référera à ces décisions complémentaires comme décisions de la contrainte c_Δ ; parfois, c_Δ est considéré comme un ensemble.

Les décisions d'appartenance peuvent avoir trois états différents selon le problème courant. Une décision d'appartenance $x \in S_x$ est dite *satisfaite* (i.e. toujours vraie) si et seulement si $dom(x) \subseteq S_x$. Une décision d'appartenance $x \in S_x$ est dite *falsifiée* (i.e. toujours fausse) si et seulement si $dom(x) \cap S_x = \emptyset$. Une décision d'appartenance qui est ni satisfaite ni falsifiée est dite *libre* (indéterminée) ; nous avons $\emptyset \subset dom(x) \setminus S_x \subset dom(x)$. Par exemple, si $dom(x) = \{a, b\}$, $dom(y) = \{b\}$ et $dom(z) = \{a, c\}$, alors $x \in \{c\} \vee y \in \{a, b\} \vee z \in \{a\}$ est une contrainte telle que la première décision est falsifiée, la seconde est satisfaite et la troisième est libre.

Un IPS Δ, vu comme une contrainte de dominance c_Δ, indique simplement qu'au moins une décision apparaissant dans l'expression du prédicat de c_Δ doit être évaluée à vrai. Quatre cas sont possibles lorsque l'on utilise une contrainte de dominance c_Δ :

1. c_Δ est universelle car une décision de c_Δ est satisfaite : le problème courant ne peut être dominé par Δ.
2. c_Δ est violée (disentailed) parce que toutes les décisions de c_Δ sont falsifiées : le problème courant est dominé par Δ (on doit alors effectuer un retour-arrière).
3. c_Δ ne contient aucune décision satisfaite et exactement une décision libre : le problème courant est presque dominé par Δ (cette décision libre peut être forcée afin de la satisfaire).
4. c_Δ ne contient aucune décision satisfaite et (au moins) deux décisions libres : le problème courant n'est ni dominé ni presque dominé par Δ.

On peut aisément observer que tant qu'il y a (au moins) deux décisions libres dans c_Δ, la contrainte c_Δ est GAC-cohérente. En d'autres termes, la contrainte c_Δ n'est plus GAC-cohérente lorsque Δ domine ou presque domine le réseau courant. Si le problème courant est dominé par un IPS Δ, cela veut dire que la contrainte c_Δ est violée et que par conséquent il faut effectuer un retour-arrière (si possible). Si le problème courant est presque dominé par un IPS Δ, le fait de forcer l'unique décision libre (à être satisfaite) revient à établir GAC en supprimant certaines valeurs du domaine de la variable impliquée dans cette décision. Par conséquent, contrôler la dominance et établir la cohérence 1-dominance [15] sur un IPS Δ est équivalent à établir GAC sur c_Δ.

La structure de données paresseuse des watched literals [20] peut être exploitée pour établir efficacement GAC sur des contraintes de dominance. Le principe est de marquer deux valeurs dans deux décisions d'appartenance **distinctes** de chaque contrainte de dominance c_Δ : ils permettent d'identifier le moment où un IPS domine ou presque domine le problème courant.

Par exemple, supposons que l'on ait identifié deux IPSs (sur P^{init}) $\Delta_1 = \{x \in \{c\}, y \in \{a\}, z \in \{a,b\}\}$ et $\Delta_2 = \{x \in \{b,c\}, w \in \{a,c\}\}$, et que l'on ait enregistré les contraintes de dominance $c_{\Delta_1} : x \in \{a,b\} \vee y \in \{b,c\} \vee z \in \{c\}$ et $c_{\Delta_2} : x \in \{a\} \vee w \in \{b\}$; le domaine initial de chaque variable est $\{a,b,c\}$. La figure 4 montre la base de contraintes de dominance, avec les valeurs (x,a) et (z,c) marquées dans la première contrainte de dominance et les valeurs (x,a) et (w,b) marquées dans la seconde. Les valeurs marquées sont indiquées par les marqueurs w_1 et w_2. Nous avons également un tableau de nd entrées telles que chaque entrée correspond à une valeur (x,a) du problème (initial) et représente la tête d'une liste chaînée, notée $\mathscr{B}_{(x,a)}$, permettant l'accès aux contraintes de dominance de $\mathscr{B}$ qui contiennent (x,a) comme valeur marquée.

Pour chaque contrainte de dominance, tant que les deux valeurs marquées sont présentes, cela veut dire que les décisions d'appartenance dans lesquelles elles apparaissent sont libres (ou satisfaites) et que par conséquence la contrainte est GAC-cohérente. La propagation est guidée par les valeurs supprimées. Lorsqu'une valeur (x,a) est supprimée, la liste $\mathscr{B}_{(x,a)}$ est visitée. Pour chaque contrainte de dominance c de cette liste, une valeur *marquable*, i.e. une valeur présente dans le problème courant, doit être recherchée (mais pas dans le domaine contenant la seconde valeur marquée). Soit cette recherche aboutit et la valeur trouvée devient la nouvelle valeur marquée (remplaçant (x,a)), soit il faut forcer la décision impliquant la seconde valeur marquée de c_Δ afin de rendre c_Δ GAC-cohérente (potentiellement en générant un domaine vide).

Par exemple, la base de contraintes de dominance de la figure 5 est obtenue à partir de la situation illustrée sur la figure 4 lorsque la valeur (z,c) est supprimée. Par conséquent, la valeur (y,b) est maintenant marquée au lieu de (z,c) dans c_{Δ_1}. Imaginons qu'un peu plus tard, l'on assigne la valeur c à la variable x ; les valeurs (x,a) et (x,b) sont alors supprimées. Les deux contraintes de dominance de $\mathscr{B}_{(x,a)}$ permettent d'effectuer des inférences (parce qu'il n'y a plus d'autres valeurs marquables). Ceci est montré sur la figure 6.

Exploiter la technique des valeurs marquées sur des contraintes de dominance possède deux principaux avantages. Tout d'abord, la complexité dans le pire des cas pour établir GAC sur une contrainte de dominance c est linéaire en la taille de c. Plus précisément, si k est le nombre total de valeurs apparaissant dans les décisions d'appartenance de c, i.e. $k = \sum_{(x \in S_x) \in c} |S_x|$, alors c peut être rendu GAC-cohérent en $O(k)$; ce résultat est borné par $O(nd)$. Il suffit de contrôler si une valeur est marquable seulement une fois. Par exemple, w_1 peut parcourir la liste des valeurs apparaissant dans c de la gauche vers la droite tandis que w_2 peut parcourir la même liste dans l'ordre inverse. La recherche d'une valeur marquable est stoppée lorsque w_1 et w_2 font référence à la même décision d'appartenance. Deuxièmement, aucun travail de restauration n'est requis lorsque l'on effectue un retour-arrière. Non seulement cela permet d'économiser du temps, mais en plus cela facilite l'intégration de cette technique dans les solveurs de contraintes. Lorsqu'une valeur (x,a) est supprimée, la base de contraintes de dominance $\mathscr{B}$ est consultée, et plus précisément, les contraintes de dominance accessibles par $\mathscr{B}_{(x,a)}$ sont contrôlées. Quand des valeurs sont restaurées, aucune opération de maintenance n'est nécessaire.

Dans [9, 17], les auteurs ont suggéré l'utilisation de la structure des watched literals pour propager des nogoods généralisés. Comme mentionné dans la section précédente, les nogoods généralisés sont basiquement équivalents aux IPSs. Classiquement, les inférences sont effectuées par la propagation unitaire. Ce mécanisme est strictement plus faible que GAC. Par exemple, soit $x = a \vee x = b \vee y \neq c$ un nogood généralisé (exprimé ici comme une contrainte) tel que $dom^{init}(x) = dom^{init}(y) = \{a,b,c\}$, $dom(x) = \{a,b,c\}$ et $dom(y) = \{c\}$. Dans ce nogood, les deux décisions $x = a$ et $x = b$ sont libres, et donc aucune inférence ne peut être effectuée. L'IPS équivalent à ce nogood généralisé est $x \in \{a,b\} \vee y \in \{a,b\}$. En établissant GAC, on peut déduire $x \neq c$. Dans [17], une approche différente pour stocker et établir GAC sur des nogoods (généralisés) est également proposée. Les nogoods sont capturés par un automate qui permet une représentation compacte mais la compilation dynamique de ces nogoods semble difficile à mettre en place en pratique.

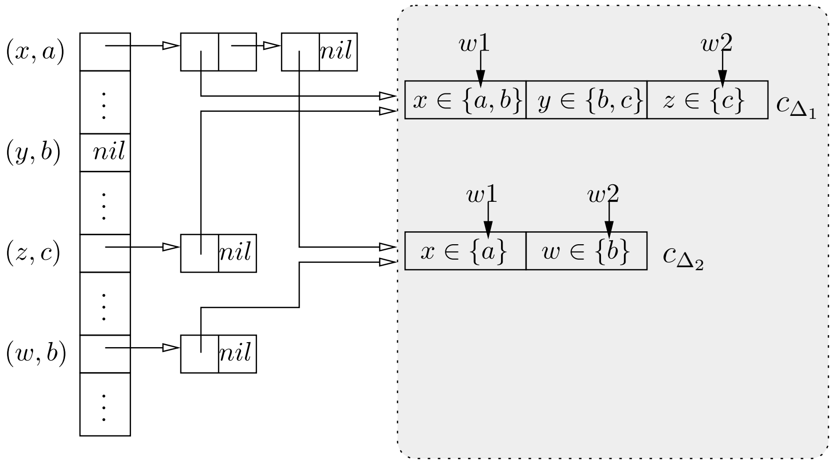

FIG. 4 – Une base de contraintes de dominance $\mathscr{B}$ incluant deux contraintes de dominance c_{Δ_1} : $x \in \{a,b\} \vee y \in \{b,c\} \vee z \in \{c\}$ et $c_{\Delta_2} : x \in \{a\} \vee w \in \{b\}$.

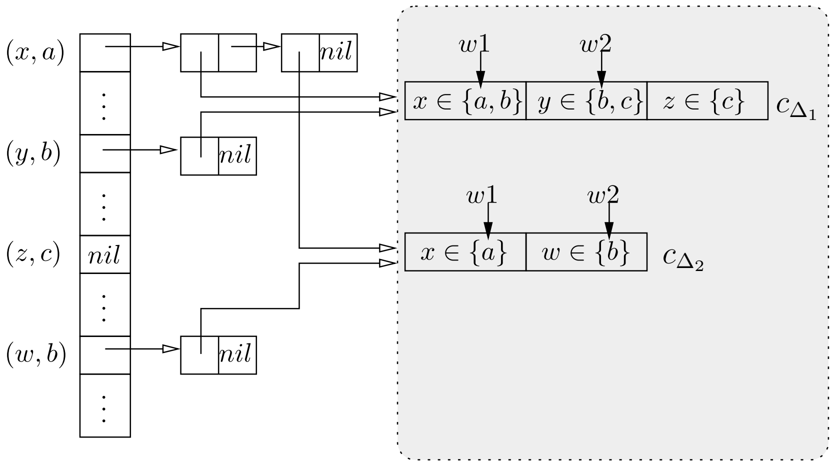

FIG. 5 – La base $\mathscr{B}$ de la figure 4 après que la valeur (z,c) soit éliminée. (y,b) est la nouvelle valeur marquée.

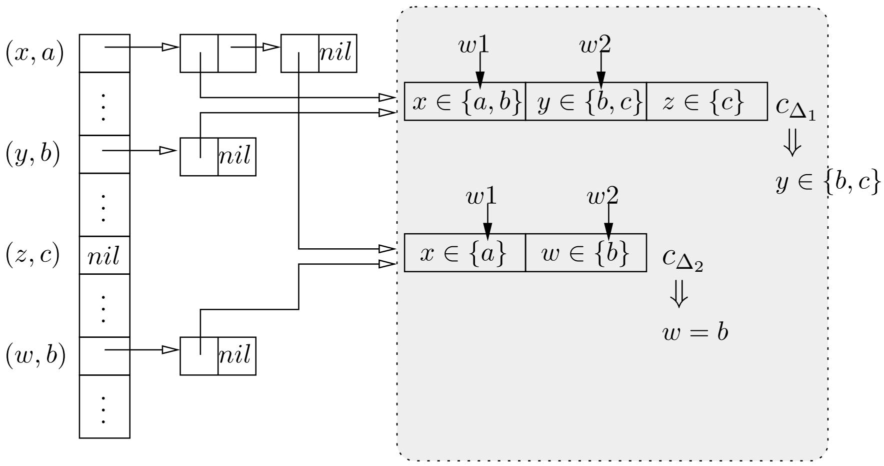

FIG. 6 – La base $\mathscr{B}$ de la figure 5 après que x soit assignée à la valeur c. Les décisions $y \in \{b,c\}$ et $w = b$ sont inferrées.

6 Identifier les IPSs

Les nogoods généralisés et la cohérence (FVC) basée sur les valeurs en échec [11] entrent dans le cadre des états partiels. Par manque de place, nous supprimons cette partie qui n'est pas nécessaire à la compréhension de l'article.

7 Opérateurs de réduction

Pour identifier des IPSs, on peut directement travailler avec les états issus d'impasses internes rencontrées au cours de la recherche. Les impasses internes sont des noeuds qui sont les racines de sous-arbres infructueux. Comme nous l'avons mentionné précédemment, l'état courant de chaque impasse interne est un IPS global. Plus précisément, si v est une impasse interne et $P = cn(v)$ le CN associé à v, alors l'état courant de P est un IPS sur P^{init}. De tels IPSs construits à partir d'impasses internes sont dits *élémentaires*.

Dans cette section, nous présentons plusieurs opérateurs dont le rôle est d'éliminer des variables (à strictement parler, des décisions d'appartenance) des IPSs élémentaires. Les états partiels obtenus après réduction sont des sous-ensembles d'états élémentaires dits *simples*.

Définition 9. *Un état partiel Δ sur un CN P est* simple *ssi $\forall x \in vars(\Delta)$, $dom^{\Delta}(x) = dom^{P}(x)$.*

Par exemple, considérons un CN P tel que $vars(P) = \{x, y, z\}$ avec $dom^P(x) = dom^P(y) = dom^P(z) = \{a, b, c\}$. $\Delta = \{x \in \{a, b, c\}, z \in \{a, b, c\}\}$ est un état partiel simple sur P tandis que $\Delta' = \{x \in \{a, b, c\}, z \in \{a, b\}\}$ est un état partiel sur P qui n'est pas simple car $dom^{\Delta'}(z) \neq dom^P(z)$.

Les états partiels simples obtenus après réduction (tel que proposé dans cette section) sont des IPSs globaux qui peuvent être exploités au cours de la recherche. Intuitivement, plus le nombre de variables impliquées dans un IPS est faible, plus grande sera sa capacité d'élagage. Cela contribue également à réduire la consommation mémoire.

Par la suite, on considère l'algorithme de recherche classique MAC (Maintaining Generalized Arc Consistency). MAC est un algorithme de recherche avec retours-arrières qui est basé sur un schéma de branchement binaire et qui maintient GAC pendant la recherche. Par conséquent, les inférences sont effectuées localement, i.e. au niveau d'une seule contrainte, pendant le processus de propagation de contraintes. GAC peut être établi en utilisant une collection de propagateurs associés à chaque contrainte. Ces propagateurs peuvent correspondre soit à une procédure générique de filtrage soit à une procédure de filtrage spécialisée (e.g. pour des contraintes globales).

7.1 Variables e-eliminables

Nous présentons un premier opérateur qui supprime les variables *e-eliminables*. Une variable e-eliminable est seulement impliquée dans des contraintes universelles ; ainsi elle ne peut plus jouer aucun rôle. ρ^{ent} est un opérateur qui élimine les variables e-eliminables et retourne un état partiel simple.

Définition 10. *Pour tout CN P, $\rho^{ent}(P)$ représente l'état partiel $\{(x \in dom^P(x)) \mid x \in vars(P) \wedge x$ n'est pas une variable e-eliminable de $P\}$.*

La proposition suivante établit le fait que ρ^{ent} est un opérateur permettant l'extraction d'un IPS à partir d'un CN insatisfaisable.

Proposition 4. *Si P est un CN insatisfaisable alors $\rho^{ent}(P)$ est un IPS sur tout CN $P' \succeq P$.*

De manière intéressante, cela veut dire que pour chaque impasse interne v rencontrée pendant la recherche, ρ^{ent} extrait un IPS sur P^{init} à partir de $cn(v)$, qui est un IPS global. Une illustration est donnée dans [13] (où ρ^{ent} est appelé ρ^{uni}).

7.2 Utilisation de preuves d'insatisfaisabilité

Proposition 5. *Si C est un noyau insatisfaisable d'un CN P alors $\Delta = \{(x \in dom^P(x)) \mid x \in vars(C)\}$ est un IPS sur tout CN $P' \succeq P$.*

En particulier, nous savons que Δ est un IPS global, i.e. un IPS sur P^{init}. De manière intéressante, il est possible d'identifier efficacement des noyaux insatisfaisables à chaque impasse interne en gardant la trace de toutes les contraintes impliquées dans des preuves d'insatisfaisabilité [2]. Ces contraintes sont celles utilisées durant la recherche pour supprimer, grâce à leurs propagateurs, au moins une valeur dans le domaine d'une variable. Cette approche peut être adaptée pour extraire des noyaux insatisfaisables à partir d'impasses internes en collectant des informations pertinentes dans les sous-arbres infructueux explorés.

L'algorithme 1 montre comment intégrer cette méthode à MAC. La fonction récursive MAC^{prf} détermine la satisfaisabilité d'un réseau P et retourne un couple composé d'une valeur booléenne (qui indique si P est satisfaisable ou pas), et d'un ensemble de variables. Cet ensemble est soit vide (quand P est satisfaisable) soit représente une preuve d'insatisfaisabilité. Une preuve est composée des variables impliquées dans la portée de contraintes qui ont déclenché au moins la suppression d'une valeur pendant la propagation.

A chaque noeud, une preuve est construite à partir de toutes les inférences produites pendant l'établissement de GAC, noté $GAC(P)$ à la ligne 2, et les preuves (lignes 6 et 8) associées aux sous-arbres gauche et droit (une fois qu'un couple (x, a) a été sélectionné). Quand un noeud est prouvé être une impasse interne après avoir pris en considération les deux branches (la première étiquetée par $x = a$ et la seconde par $x \neq a$), une preuve d'insatisfaisabilité (entre les lignes 9 et 10) est obtenue en fusionnant simplement les preuves associées aux branches droite et gauche ; ici c'est pour P'. Notons que la complexité spatiale dans le pire des cas pour gérer les différentes preuves locales de l'arbre de recherche est $O(n^2 d)$ puisqu'enregistrer une preuve est $O(n)$ et qu'il y a au plus $O(nd)$ noeuds par branche.

Algorithm 1: $\mathsf{MAC}^{prf}(P : \text{CN})$: (Boolean, set of variables)

1 $localProof \leftarrow \emptyset$
2 $P' \leftarrow GAC(P)$; `// ` $localProof$ `est maj par GAC`
3 **if** $P' = \bot$ **then return** *(false,localProof)*
4 **if** $\forall x \in vars(P'), |dom(x)| = 1$ **then return** *(true,$\emptyset$)*
5 select a value (x, a) of P' such that $|dom(x)| > 1$
6 $(sat, leftProof) \leftarrow \mathsf{MAC}^{prf}(P'|_{X=a})$
7 **if** sat **then return** *(true,$\emptyset$)*
8 $(sat, rightProof) \leftarrow \mathsf{MAC}^{prf}(P'|_{X \neq a})$
9 **if** sat **then return** *(true,$\emptyset$)*
`// ` $leftProof \cup rightProof$ `est une preuve pour` P'
10 **return** $(false, localProof \cup leftProof \cup rightProof)$

En utilisant l'algorithme 1, on peut introduire un second opérateur qui ne sélectionne que les variables impliquées dans une preuve d'insatisfaisabilité. Cet opérateur peut être utilisé incrémentalement à n'importe quelle impasse interne d'un arbre construit par MAC.

Définition 11. *Soit P un CN tel que $\mathsf{MAC}^{prf}(P) = (false, proof)$. $\rho^{prf}(P)$ représente l'état partiel simple $\{(x \in dom^P(x)) \mid x \in proof\}$.*

La proposition suivante établit le fait que ρ^{prf} est un opérateur qui permet d'extraire des IPSs.

Proposition 6. *Si P est un CN insatisfaisable alors $\rho^{prf}(P)$ est un IPS sur tout CN $P' \succeq P$.*

Démonstration. Soit $\mathsf{MAC}^{prf}(P) = (false, proof)$. On peut montrer que $C = (proof, \{c \in cons(P) \mid scp(c) \subseteq proof\})$ est un noyau insatisfaisable de P. Nous déduisons le résultat de la définition de l'opérateur ρ^{prf} et de la proposition 5. □

Similairement à ρ^{ent}, ρ^{prf} permet d'extraire des IPSs globaux. En pratique, dans l'algorithme 1, l'opérateur ρ^{prf} peut être appelé pour extraire un IPS entre les lignes 9 et 10. La proposition suivante établit le fait que ρ^{prf} est plus performant que ρ^{ent} (i.e. permet d'extraire des IPSs représentant une plus large portion de l'espace de recherche).

Proposition 7. *Si P est un CN insatisfaisable alors $\rho^{prf}(P) \subseteq \rho^{ent}(P)$.*

Démonstration. Une contrainte universelle ne peut intervenir dans une preuve d'insatisfaisabilité. Une variable e-eliminable n'apparaît que dans des contraintes universelles, donc est nécessairement éliminée par ρ^{prf} (en considérant les hypothèses faites au début de la section). □

7.3 Utilisation de justifications

Le principe de l'opérateur présenté dans cette section est de construire un état partiel simple en éliminant les variables dont le domaine courant peut être déduit des autres. Ceci est possible en gardant la trace des contraintes à l'origine des suppressions de valeurs. Lorsqu'un propagateur associé à une contrainte c supprime une valeur (x, a), la contrainte c est enregistrée comme la *justification* de la suppression de (x, a). C'est une forme d'explication (de valeur éliminée) même si les explications sont classiquement basées sur des décisions (i.e. formées de décisions positives et négatives). Dans une certaine mesure, cela correspond à une utilisation basique de la définition générale de nogood proposée dans [18] qui inclut un ensemble de contraintes jouant le rôle de justification pour le nogood. Des justifications similaires ont été également exploitées pour établir AC sur des instances dynamiques [3].

Pour notre propos, nous avons besoin de raisonner avec un graphe d'implication à gros grain, appelé graphe de dépendance ici, et construit à partir des justifications. Lorsqu'une décision positive $x = a$ est prise (par l'algorithme de recherche), $just(x \neq b)$ est initialisé à nil pour toute valeur $b \in dom(x) \mid b \neq a$, et quand une décision négative $x \neq a$ est prise, $just(x \neq a)$ est également initialisé à nil. D'un autre coté, quand une valeur (x, a) est supprimée par un propagateur associé à une contrainte c, la justification de $x \neq a$ est simplement donnée par c : nous avons $just(x \neq a) = c$. Comme notre but est de circonscrire un état partiel simple, nous avons seulement besoin de savoir pour chaque valeur (x, a) supprimée, les variables responsables de sa suppression ; ce sont celles impliquées dans $just(x \neq a)$. A partir de ces informations, on peut construire un graphe orienté G ou les noeuds correspondent aux variables et les arcs aux dépendances entre les variables. Plus précisément, un arc existe dans G d'une variable x vers une variable y s'il existe une valeur supprimée (y, b) telle que sa justification est une contrainte impliquant x. On ajoute également un noeud spécial nil, et un arc existe entre nil et une variable x si cette variable est impliquée dans une décision (positive ou négative) prise par l'algorithme de recherche, i.e. s'il existe une valeur supprimée (x, a) telle que sa justification soit nil. Le graphe de dépendance peut être utilisé pour réduire les IPSs ; ceci est décrit ci-dessous.

La figure 7 illustre notre propos. Sur la gauche de la figure, on représente le CN binaire initial P^{init}. P^{init} implique quatre variables et trois contraintes ; $vars(P^{init}) =$

$\{w, x, y, z\}$ et $cons(P^{init}) = \{w \neq x, x \geq y, x \leq z\}$. Le CN courant P est également représenté, il est obtenu à partir de P^{init} après avoir assigné la valeur 3 à w et établi AC. La figure 7 fournit les justifications des valeurs supprimées dans P ainsi que le graphe de dépendance construit à partir de ces explications. Les justifications sont obtenues comme suit. Lorsque la décision positive $w = 3$ est prise, les justifications de $w \neq 1$ et $w \neq 2$ sont initialisées à nil. Ces suppressions sont propagées à x grâce à la contrainte $w \neq x$, menant à la suppression de la valeur 3 de $dom(x)$; on obtient alors $just(x \neq 3) = (w \neq x)$. Cette nouvelle suppression est alors propagée aux variables y et z : la valeur 3 est supprimée de $dom(y)$ via la propagation de $x \geq y$ ce qui constitue sa justification et la valeur 0 est supprimée de $dom(z)$ via la propagation de $x \leq z$. Le graphe de dépendance est directement construit à partir de ces justifications.

Définition 12. *Soit x une variable de P et $a \in dom^{init}(x) \setminus dom^{P}(x)$. La justification de la suppression de (x, a), notée $just(x \neq a)$ est, si elle existe, la contrainte dont le propagateur associé a supprimé (x, a) sur le chemin menant de la racine de l'arbre de recherche au noeud v où $cn(v) = P$; sinon $just(x \neq a)$ est nil.*

Les justifications sont utilisées pour extraire un état partiel simple à partir d'un ensemble de variables X. Cet état partiel contient les variables de X qui ne peuvent pas être "expliquées" par X.

Définition 13. *Soit $X \subseteq vars(P)$ un ensemble de variables de P. Une variable $x \in X$ est j-eliminable de P vis-à-vis de X si et seulement si $\forall a \in dom^{P^{init}}(x) \setminus dom^{P}(x)$, $just(x \neq a)$ est une contrainte c telle que $c \neq nil$ et $scp(c) \subseteq X$.*

Le graphe de dépendance mentionné plus haut permet d'identifier directement ces variables. ρ^{jst} est un opérateur qui élimine les variables j-eliminables et retourne un état partiel simple.

Définition 14. *Soit $X \subseteq vars(P)$. $\rho^{jst}_{X}(P)$ est l'état partiel simple $\{(x \in dom^{P}(x)) \mid x \in X \wedge x$ n'est pas une variable j-eliminable de P vis à vis de $X\}$.*

Réduire des états partiels en éliminant les variables j-eliminables ne provoque pas fondamentalement de perte d'informations lorsqu'on considère P^{init} et GAC (la preuve est omise).

Proposition 8. *Soit Δ un état partiel simple sur P et $\Delta' = \rho^{jst}_{vars(\Delta)}(P)$. Nous avons : $GAC(P^{init}|_{\Delta'}) = GAC(P^{init}|_{\Delta})$.*

En utilisant la proposition 8, on peut montrer que pour tout CN P, $\rho^{jst}_{vars(P)}(P)$ produit un IPS global. Cependant, cet IPS n'est pas intéressant parce qu'il est (pour l'essentiel) équivalent à l'état courant P. En effet, c'est un état partiel avec toutes les variables impliquées dans une décision prise par l'algorithme de recherche. Cet IPS est équivalent au nogood généralisé correspondant à l'ensemble des décisions prises sur la branche menant de la racine de l'arbre de recherche à P. Heureusement, on peut utiliser l'opérateur ρ^{jst} après application d'un autre opérateur produisant un IPS (simple), comme montré dans le corollaire suivant.

Corollaire 1. *Soit Δ un état partiel simple sur P et $\Delta' = \rho^{jst}_{vars(\Delta)}(P)$. Si Δ est un IPS sur P^{init} alors Δ' est un IPS sur P^{init}.*

Démonstration. Si Δ est un IPS sur P^{init} alors $P^{init}|_{\Delta}$ est insatisfaisable. Comme GAC préserve la satisfaisabilité, et $GAC(P^{init}|_{\Delta'}) = GAC(P^{init}|_{\Delta})$ par la proposition 8, on peut déduire que $P^{init}|_{\Delta'}$ est insatisfaisable. Par conséquence Δ' est un IPS sur P^{init}. □

En conséquence du corollaire 1, on a la garantie que les deux opérateurs suivants produisent des IPSs à partir de CNs insatisfaisables.

Définition 15. *Soit P un CN.*
- *$\rho^{jst \odot ent}(P) = \rho^{jst}_{vars(\Delta)}(P)$ avec $\Delta = \rho^{ent}(P)$.*
- *$\rho^{jst \odot prf}(P) = \rho^{jst}_{vars(\Delta)}(P)$ avec $\Delta = \rho^{prf}(P)$.*

La figure 8 illustre (ici, sur des états partiels consistants) le comportement de ρ^{ent}, ρ^{jst} et leur combinaison $\rho^{jst \odot ent}$. Appliquer ρ^{ent} sur le réseau P de la figure 7 provoque l'élimination de w, menant à l'état partiel Δ_1, parce que w est seulement impliquée dans des contraintes universelles. En effet, la valeur restante 3 dans $dom(w)$ est compatible avec les deux valeurs restantes 1 et 2 de $dom(x)$ via la contrainte $w \neq x$. Les trois autres variables sont impliquées dans des contraintes qui ne sont pas universelles. Appliquer ρ^{jst} sur le réseau P vis-à-vis de $X = vars(P)$ provoque l'élimination de x, y et z menant à l'état partiel $\Delta_2 = \{w \in \{3\}\}$. En effet, w est la seule variable pour laquelle une suppression est justifiée par nil ; X étant $vars(P)$, c'est la seule condition pertinente pour déterminer les variables d'intérêt. Ceci illustre le fait qu'appliquer l'opérateur ρ^{jst} vis à vis de toutes les variables d'un CN n'a pas d'intérêt : puisque l'on obtient l'ensemble des décisions prises (ici $w = 3$), l'état partiel ne pourra jamais être rencontré ou dominé plus tard sans redémarrage. L'application de $\rho^{jst \odot ent}$ est plus intéressante. Une fois que ρ^{ent} a été appliqué, on obtient l'état partiel Δ_1 dont les variables sont $\{x, y, z\}$. On peut appliquer ρ^{jst} pour déterminer quelles variables de Δ_1 ont un domaine qui peut être déduit des autres variables de Δ_1. La variable x est la seule variable pour laquelle toutes les valeurs supprimées ne peuvent pas être justifiées par les contraintes impliquant des variables *internes* à Δ_1 : $just(x \neq 3)$ implique une variable *externe*. Ceci est directement visible avec le graphe

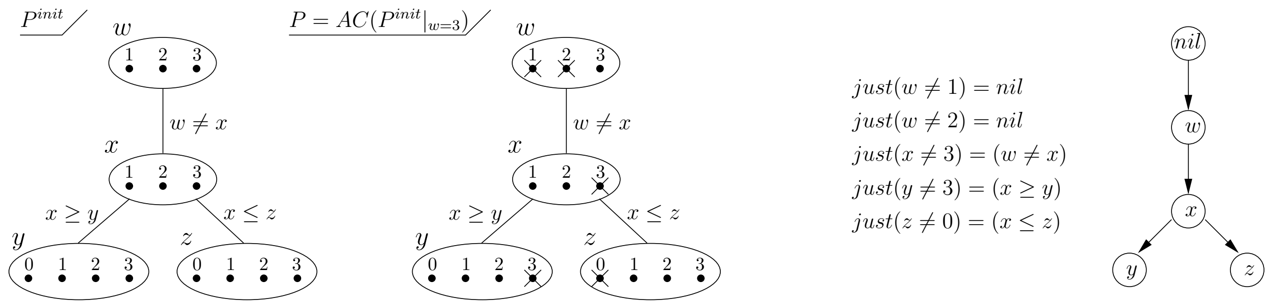

FIG. 7 – Gérer des justifications pendant la recherche.

$$P^{init} : \left\{ \begin{array}{l} w \in \{1,2,3\} \\ x \in \{1,2,3\} \\ y \in \{0,1,2,3\} \\ z \in \{0,1,2,3\} \end{array} \right\} \qquad P = AC(P^{init}|_{w=3}) : \left\{ \begin{array}{l} w \in \{3\} \\ x \in \{1,2\} \\ y \in \{0,1,2\} \\ z \in \{1,2,3\} \end{array} \right\}$$

$$\Delta_1 = \rho^{ent}(P) = \left\{ \begin{array}{l} x \in \{1,2\} \\ y \in \{0,1,2\} \\ z \in \{1,2,3\} \end{array} \right\}$$

$$\Delta_2 = \rho^{jst}_{vars(P)}(P) = \{\ w \in \{3\}\ \}$$

$$\Delta_3 = \rho^{jst \odot ent}(P) = \rho^{jst}_{vars(\Delta_1)}(P) = \{\ x \in \{1,2\}\ \}$$

FIG. 8 – États courants de P^{init} et P de la figure 7, et états partiels extraits en utilisant ρ^{ent}, ρ^{jst} et $\rho^{jst \odot ent}$.

de dépendance sur la figure 7. Nous obtenons alors l'état partiel $\Delta_3 = \{y \in \{1,2\}\}$.

La complexité spatiale pour l'enregistrement des justifications est $O(nd)$ alors que la complexité temporelle pour gérer cette structure est $O(1)$ même si une valeur est supprimée ou restaurée pendant la recherche.

8 Conclusion

Dans ce papier, nous avons brièvement montré les connexions existantes entre différentes approches d'apprentissage à l'aide d'un cadre général : celui des états partiels inconsistants. Deux opérateurs originaux permettant l'extraction d'IPSs ont également été décrits (et testés expérimentalement dans [12]). L'identification et l'exploitation d'IPSs sont deux activités prometteuses au regard des résultats déjà obtenus [10, 16, 13, 12].

Références

[1] R. Backofen and S. Will. Excluding symmetries in constraint based search. In *Proc. of CP'99*, pp. 73–87, 1999.

[2] R.R. Baker, F. Dikker, F.Tempelman, and P.M. Wognum. Diagnosing and solving over-determined constraint satisfaction problems. In *Proc. of IJCAI'93*, pp. 276–281, 1993.

[3] C. Bessiere. Arc-consistency in dynamic constraint satisfaction problems. In *Proc. of AAAI'91*, pp. 221–226, 1991.

[4] C. Bessiere. Constraint propagation. In *Handbook of Constraint Programming*, chapter 3. Elsevier, 2006.

[5] T. Fahle, S. Schamberger, and M. Sellman. Symmetry breaking. In *Proc. of CP'01*, pp. 93–107, 2001.

[6] F. Focacci and M. Milano. Global cut framework for removing symmetries. In *Proc. of CP'01*, pp. 77–92, 2001.

[7] E.C. Freuder and P.D. Hubbe. Using inferred disjunctive constraints to decompose CSPs. In *Proc. of IJCAI'93*, pp. 254–261, 1993.

[8] I. Gent and B. Smith. Symmetry breaking during search. In *Proc. of ECAI'00*, pp. 599–603, 2000.

[9] G. Katsirelos and F. Bacchus. Unrestricted nogood recording in CSP search. In *Proc. of CP'03*, pp. 873–877, 2003.

[10] G. Katsirelos and F. Bacchus. Generalized nogoods in CSPs. In *Proc. of AAAI'05*, pp. 390–396, 2005.

[11] C. Lecoutre and O. Roussel. Cohérences basées sur les valeurs en échec. In *Proc. of JFPC'09*, 2009.

[12] C. Lecoutre, L. Sais, S. Tabary, and V. Vidal. Exploiting past and future : Pruning by inconsistent partial state dominance. In *Proc. of CP'07*, pp. 453–467, 2007.

[13] C. Lecoutre, L. Sais, S. Tabary, and V. Vidal. Transposition Tables for CSPs. In *Proc. of AAAI'07*, pp 243–248, 2007.

[14] J.F. Puget. Symmetry breaking revisited. *Constraints*, 10(1) :23–46, 2005.

[15] I. Razgon and A. Meisels. Maintaining dominance consistency. In *Proc. of CP'03*, pp. 945–949, 2003.

[16] I. Razgon and A. Meisels. A CSP search algorithm with responsibility sets and kernels. *Constraints*, 12(2) :151–177, 2007.

[17] G. Richaud, H. Cambazard, B. O'Sullivan, and N. Jussien. Automata for nogood recording in CSPs. In *Proc. of SAT/CP workshop held with CP'06*, 2006.

[18] T. Schiex and G. Verfaillie. Nogood recording for static and dynamic CSPs. *IJAIT*, 3(2) :187–207, 1994.

[19] L. Zhang, C.F. Madigan, M.W. Moskewicz, and S. Malik. Efficient conflict driven learning in a Boolean satisfiability solver. In *Proc. of ICCAD'01*, pp. 279–285, 2001.

[20] L. Zhang and S. Malik. The quest for efficient Boolean satisfiability solvers. In *Proc. of CADE'02*, pp. 295–313, 2002.