# CSP dynamiques en configuration

Thomas van Houdenove, Paul Gaborit, Michel Aldanondo, Élise Vareilles

# CSP dynamiques en configuration

**Article Jeune Chercheur JFPC'06**

**Thomas van Oudenhove, Paul Gaborit, Michel Aldanondo, Élise Vareilles**

Laboratoire de Génie Industriel
École des Mines d'Albi-Carmaux,
Route de Teillet,
81013 Albi Cedex 09
{vanouden, gaborit, aldanond, vareille}@enstimac.fr

**Résumé**

Les CSP constituent une bonne approche pour gérer les problèmes de configuration à base de contraintes. Cependant, la configuration fait ressortir des besoins qui peuvent difficilement être traités par des CSP classiques : les composants optionnels et la gestion hiérarchique des composants. Certains travaux apportent des solutions à certains de ces besoins (DCSP pour l'activation de variables, CCSP pour la gestion des groupes), mais utilisent généralement des approches incompatibles entre elles. À travers une étude de la sémantique du dynamisme dans les CSP, nous proposons une tentative d'unification de ces approches, dans le but d'obtenir un modèle basé sur des CSP. Ce modèle permet de traiter les problèmes de configuration autonome mais surtout interactive, en utilisant les avantages de chacune de ces différentes approches. De plus, nous proposons un nouvel opérateur permettant de retarder la prise en compte de parties du problème initial, ceci afin d'allèger les phases de propagation.

## 1 Introduction — besoins en configuration

La configuration est un type de problèmes particuliers pour lesquels l'espace de solution est connu. Les approches à base de contraintes, et plus particulièrement les CSP [10], constituent une des méthodes permettant de gérer des problèmes de configuration [9].

Un processus de configuration peut être accompli de deux façons : autonome ou interactive. La configuration autonome consiste à trouver au moins une solution au problème. La configuration autonome repose sur des méthodes de résolution de CSP (par exemple, le *BackTrack* [5]). La configuration interactive, quant à elle, consiste à réduire progressivement l'espace des solutions en se basant sur les choix de l'utilisateur. Les techniques de configuration interactive reposent sur des méthodes de filtrage de CSP. Généralement, la cohérence d'arc [6] est utilisée ; bien que peu déductive (par rapport à la $k$-cohérence, par exemple), elle permet de garder une certaine interactivité (rapidité de filtrage). Lors d'un processus de configuration interactive, on présente généralement à l'utilisateur les projections des résultats du filtrage sur les domaines. Celui-ci peut alors réduire les domaines ou valuer des variables.

Dans cette communication, nous ne parlerons que des méthodes de configuration interactive. L'implémentation utilisée pour les tests utilise un système de filtrage par cohérence d'arc.

En configuration, plusieurs besoins se dégagent : en premier lieu, celui de pouvoir traiter des variables continues. En effet, nous devons pouvoir gérer par exemple la configuration de meubles sur mesure, qui peuvent comporter des dimensions dont le domaine est continu ; nous devrons donc disposer de méthodes permettant de filtrer ces domaines. Un second besoin est la gestion de composants optionnels. Ainsi, un toit ouvrant sur une voiture est facultatif, et le modèle de la configuration doit prendre en compte cette existence conditionnée. Généralement, la description d'une configuration se base sur un modèle générique

du produit à configurer. Pour l'écriture de ce modèle en CSP, nous pouvons identifier un troisième besoin : l'utilisation de groupes, qui correspondent à des composants regroupant plusieurs variables et/ou contraintes du CSP (par exemple, le moteur d'une voiture constitue un groupe dont certains éléments, représentés par des variables, sont contraints entre eux).

Après un bref état de l'art des concepts permettant de répondre à ces deux derniers besoins, nous allons expliciter le besoin de gérer la pertinence des éléments d'un CSP et apporter des pistes de solutions dans la section suivante. Nous allons ensuite présenter un opérateur permettant de retarder la prise en compte d'éléments du CSP (variables, contraintes, groupes) afin d'alléger certaines phases de propagation. Enfin, nous expliciterons les différentes solutions testées avant de conclure.

## 2 CSP dynamiques[1] : existant et sémantique du dynamisme

### 2.1 État de l'art

Nous avons identifié les besoins relatifs aux problèmes de configuration à base de contraintes (variables discrètes et continues, éléments optionnels et gestion des groupes) ; plusieurs propositions visent à remplir certains de ces besoins :

**DCSP (*Dynamic CSP* ) [8] :** Le cadre général défini par Mittal et Falkenhainer permet la manipulation dynamique de variables : certaines d'entre elles sont inactives au début de la résolution, et sont activables ou inactivables par des contraintes d'activité ; cette proposition répond donc au besoin de variables dynamiques en configuration autonome. Le même type de concept a été revisité par [3] pour résoudre des CSP « conditionnels » ou CondCSP comportant aussi des variables continues en générant un CSP différent par état du CondCSP.

**CCSP (*Composite CSP* ) [12] :** cette proposition permet de manipuler des objets hiérarchiquement. Certains composants du produit sont groupés et peuvent ainsi être configurés sous certaines conditions. L'existence de ces composants dans le problème dépend de la valeur d'une méta-variable qui peut disparaître et être remplacée par un pan de problème.

**CSPe (CSP à états) [15] :** les CSP à états attachent un attribut d'état (actif ou inactif) à chacune des variables, ce qui permet de contrôler leur pertinence dans le problème. Les CSPe permettent donc de gérer des variables dynamiques avec des algorithmes classiques, aussi bien en configuration autonome qu'interactive.

**ACSP (*Activity CSP* ) [4] :** les ACSP permettent de manipuler des contraintes dynamiques, ainsi que des groupes, en y associant une notion d'activité. Geller et Veksler y ont associé l'algorithme AMAC, permettant ainsi la maintenance de cohérence d'arc sur les ACSP.

### 2.2 Sémantique et dynamisme

Nous pouvons distinguer trois éléments dont la pertinence (prise en compte) peut être variable au cours du problème dans un CSP : les variables, les contraintes et les groupes (de variables, contraintes et groupes).

Nous identifions la classification suivante pour les différents éléments d'un CondCSP :

- les variables, groupes et contraintes à prendre en compte, dont l'existence n'est pas conditionnée et dont les domaines sont toujours filtrés (cas des CSP classiques) ;
- les variables, groupes et contraintes qui existent toujours dans le problème, mais dont la prise en compte dans les mécanismes de filtrage dépend d'un attribut d'état, ou de la vérification d'une formule logique (cas des CondCSP, CSPe, ACSP) ;
- les variables, groupes et contraintes n'existant pas au début du problème, qui apparaissent selon la valeur d'une variable (CCSP) ou d'une formule logique (opérateur *Sure*, cf. section 3).

#### 2.2.1 Pertinence des variables

Pour gérer l'activation de variables, il existe plusieurs méthodes, qui peuvent dépendre du type de la variable :

- utilisation d'une variable booléenne ; cette solution n'est utilisable que pour des composants non-paramétrables (sur un vélo : existe-t-il un garde-boue : Oui ou Non) ;
- ajout d'une valeur ($\star$, cf. [1]) dans chacun des domaines des variables ; si la variable prend cette valeur, elle n'est pas pertinente dans le problème en cours — cette solution pose le problème de la gestion de cette valeur particulière pour les variables numériques : comment effectuer des opérations sur cette valeur (additions, multiplications,...) ?
- ajout d'un attribut d'état booléen (CSPe), permettant d'expliciter la pertinence de la variable. Cet attribut peut être considéré obligatoire ou

[1] Nous entendons CSP dynamique au sens de [8] et non au sens d'un echaînement de CSP [14] ; par souci de clarté, nous utiliserons CondCSP par la suite.

facultatif (les variables toujours pertinentes n'en ont pas besoin).

Les CondCSP et CSPe rentrent dans le cadre de l'activation de variables, ils permettent la prise en compte de variables en différenciant les variables pertinentes (à prendre en compte) de l'ensemble total des variables du problème. Cependant, les variables qui ne sont pas prises en compte dès le départ existent tout de même dans le problème. Les CCSP permettent d'éviter cela grâce à une autre approche : les variables inutiles au départ n'existent pas et sont créées grâce à une méta-variable.

#### 2.2.2 Pertinence des contraintes

En ce qui concerne l'activation de contraintes, nous identifions trois types de mécanismes :

- considérer un CondCSP comme un enchaînement de différents CSP auxquels on ajoute des contraintes [2, 3] — cette solution a l'inconvénient d'être lourde à manipuler ;
- une utilisation des variables dynamiques : faire le choix de considérer pertinentes seulement les contraintes dont toutes les variables sont actives ;
- utilisation d'un attribut d'état par contrainte (ou solution équivalente : utiliser l'invariant de la contrainte, cf. équation 1).

$$\begin{array}{l}\text{Contrainte } C : x = 5 \times y \\ \left\{\begin{array}{l}\text{invariant } x = 5 \times y \times \text{inv} \\ \qquad\qquad \text{inv} = 1 \text{ si } C \textit{ active},\ \frac{x}{5\times y} \text{ sinon}\end{array}\right. \\ \quad \Longleftrightarrow \\ \left\{\begin{array}{l}\text{attribut d'état } (x = 5 \times y) \vee (C \textit{ inactive}) \\ \qquad\qquad C^e = 1 \text{ si } C \textit{ active},\ 0 \text{ sinon}\end{array}\right.\end{array} \tag{1}$$

#### 2.2.3 Pertinence des groupes

La notion de groupe est particulièrement utile en configuration, pour gérer des composants du produit regroupant plusieurs variables et/ou contraintes. Les groupes peuvent être considérés comme un raccourci de notation ; en effet, les variables de ces composants peuvent être gérées une à une, mais il est plus aisé de manipuler les variables d'un même composant ensemble.

Pour gérer l'activité de ce regroupement de variables et/ou contraintes, plusieurs solutions ont été proposées :

- l'utilisation d'un attribut d'état pour le groupe (ACSP, [4]), permettant la factorisation des attributs d'état de plusieurs variables — cette méthode est équivalente à celle des attributs d'état sur les variables, sauf qu'un attribut d'état est ici commun à plusieurs éléments ;
- l'utilisation d'une variable à remplacer par un pan de problème ou groupe (CCSP, [12]) — dans ce cas, le groupe n'existe pas dans le CSP tant que la méta-variable gérant son apparition n'est pas valuée. Cette approche a l'inconvénient d'interdire les contraintes entre des éléments d'un groupe et « l'extérieur » de ce groupe.

Les approches de configuration orientée objet [7] gèrent de la même façon les groupes de variables et/ou contraintes.

### 2.3 Synthèse

Nous avons présenté les besoins les plus courants en configuration : gestion par composant (un composant est un groupe de plusieurs variables et/ou contraintes) et gestion de composants optionnels (variables et groupes dynamiques). Quelques travaux issus de la littérature permettent de satisfaire certains de ces besoins. Cependant, ces travaux utilisent des approches différentes et parfois incompatibles, que nous allons essayer d'unifier après avoir introduit un nouvel opérateur.

## 3 Utilisation d'informations sur l'état du filtrage

L'approche consistant à utiliser des informations sur l'état du filtrage au cours du filtrage est à rapprocher des CCSP [12]. Elle est cependant plus permissive car elle ne nécessite pas de méta-variables. Cette méthode permet d'utiliser le résultat d'une formule (ou contrainte) pour activer un groupe. Cette activation, comme dans le cas des CCSP, consiste réellement à ajouter les variables, contraintes et groupes contenus dans ce groupe au problème (*i.e.* ils n'existent pas auparavant).

Ce type d'information simplifie la gestion d'exemples comme celui du réacteur chimique [3] (version modifiée, cf. figure 1). Ce type d'exemple utilise un calcul (ici, le volume de la cuve) dont la formule dépend des valeurs d'autres variables (forme de la cuve). Pour éviter de tester le résultat du calcul à chaque nouvelle passe du filtrage, il est possible d'utiliser l'état du filtrage à un moment donné pour savoir si la variable *forme* est valuée, et ainsi déclencher le calcul du volume.

Ainsi, sur l'exemple de la figure 1, une fois que la variable *cuve* est valuée, il faut éventuellement créer une variable (le *rayon* si la *cuve* est cylindrique), puis lancer le filtrage sur le volume de la cuve selon le domaine de la variable *hauteur*.

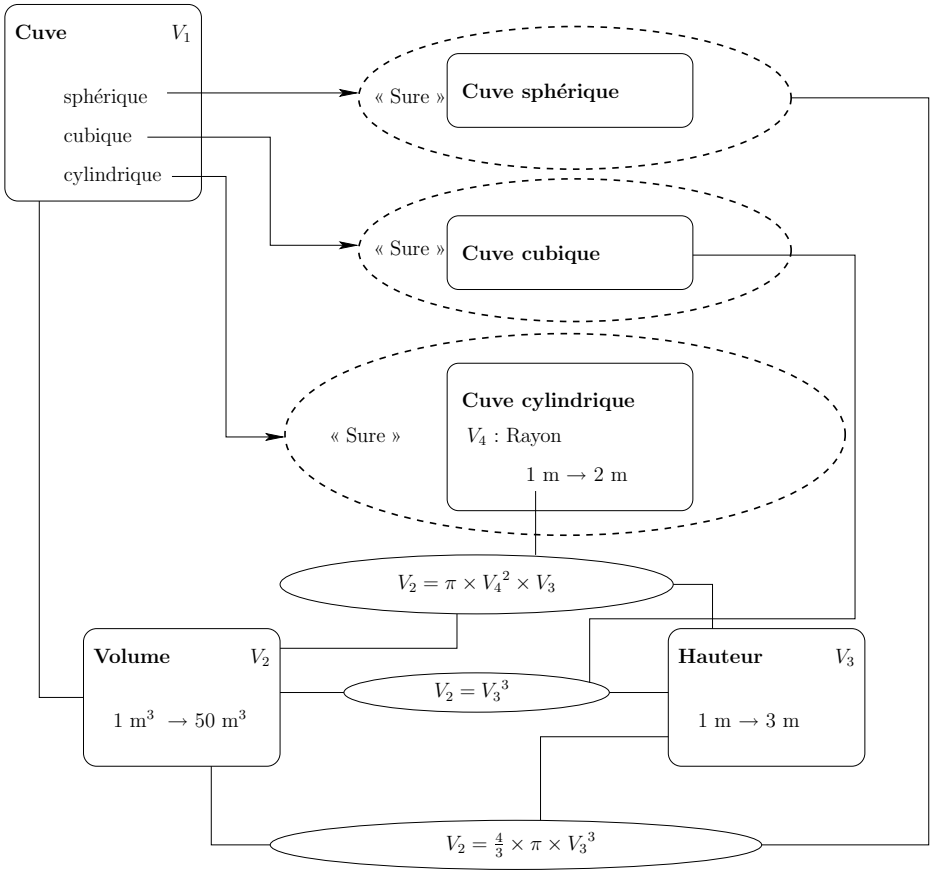


FIG. 1 – Exemple du mixer [3] modifié en utilisant la possibilité d'utiliser des informations sur l'état du filtrage

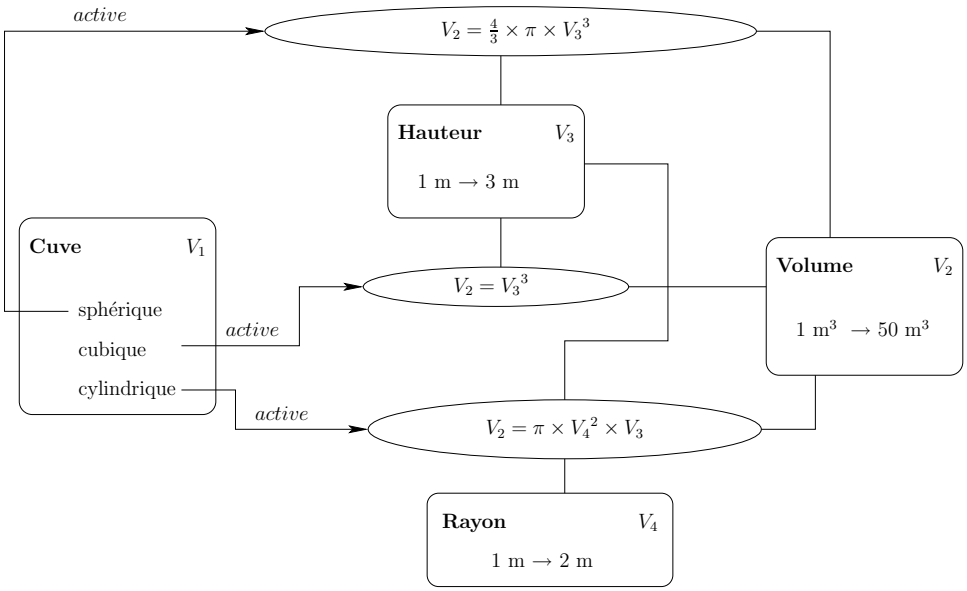


FIG. 2 – Exemple du mixer [3] modifié en utilisant les attributs d'états par contrainte

Nous proposons donc un nouvel opérateur (sûr : *Sure*) qui permet d'attendre qu'une formule soit certainement vraie pour alors la remplacer par un groupe. Le fait d'être certainement vrai implique l'attente que les domaines de certaines variables soient assez réduits pour que la condition d'activation ne puisse plus être fausse, quelque soit les valeurs des variables. Cette condition d'activation est appelée une méta-contrainte car elle raisonne sur l'état du filtrage, et non sur les domaines des variables. Ce groupe permet la déclaration de nouvelles variables et de nouvelles contraintes.

La figure 2 montre le même problème, modélisé grâce à des attributs d'états par contrainte. Dans ce cas, les calculs seront effectués mais leur résultat ne sera pas pris en compte pour filtrer les domaines tant que les contraintes ne seront pas certainement actives (de plus, la variable *rayon*, inutile deux fois sur trois, existe toujours). Ainsi, l'interêt principal de l'opérateur *Sure* est de pouvoir déclencher des calculs et d'ajouter des éléments au problème si et seulement si sa condition d'activation est vérifiée ; ceci peut être particulièrement utile lorsque le groupe dont l'activation est retardée contient un calcul complexe.

# 4 Proposition — vers une sémantique unifiée

Notre proposition consiste à unifier les différentes approches abordées ci-dessus de façon à pouvoir tirer parti des avantages de la plupart d'entre elles. Nous allons illustrer chacun des avantages que nous voyons à ces différentes approches par un petit exemple simple — dans chacun des exemples, les liaisons marquées d'un symbole $\otimes$ marquent des incompatibilités de valeurs. Ces exemples sont volontairement simplistes de façon à illustrer seulement le point étudié. Leur extension sur des exemples conséquents — comme dans [8] — a été testée.

## 4.1 Contraintes

Pour tous les exemples qui suivent, nous avons été contraints de faire un choix sur la pertinence des contraintes. Ainsi, nous adoptons le postulat suivant :

> *Une contrainte est prise en compte si et seulement si toutes les variables sur lesquelles elle porte sont actives. Dans tous les autres cas, la contrainte est considérée comme satisfaite.*

Ainsi, la prise en compte des contraintes sera pilotée par la pertinence des variables qui la composent.

Cependant, il est toujours possible de retarder la prise en compte d'une contrainte par un opérateur *Sure*, en particulier dans le cas d'une contrainte portant sur une partie du problème jugée trop complexe.

## 4.2 Variables symboliques à existence conditionnée

La figure 3 montre un problème de configuration de voiture : une voiture est de type luxe ou normal ; elle comporte des vitres électriques ou manuelles et un intérieur tissu ou cuir ($C_1$). Un toit ouvrant automatique est incompatible avec des vitres manuelles ($C_2$) et un toit ouvrant manuel est incompatible avec un intérieur cuir ($C_3$). Si (et seulement si) la voiture est de type luxe, elle est équipée d'un toit ouvrant, qui peut être manuel ou automatique ($C_4$).

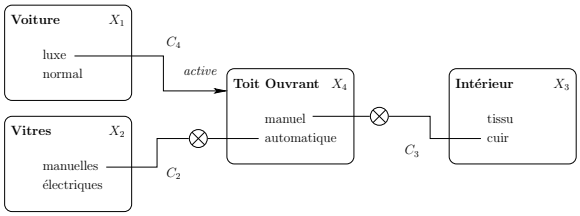


FIG. 3 – Configuration de voiture

Pour étudier la différence de comportement entre les modèles CSPe [15], CSP$\star$ [1] et CSP avec opérateur *Sure*, nous proposons de valuer la variable *vitres* à manuelles et la variable *intérieur* à cuir. Un filtrage (équivalent à de la cohérence d'arc) de qualité devrait déduire que la voiture est obligatoirement de type normal.

### Filtrage en CSPe

La figure 4 illustre la modélisation du problème sous forme d'un CSPe dont l'équation 2 montre les contraintes.

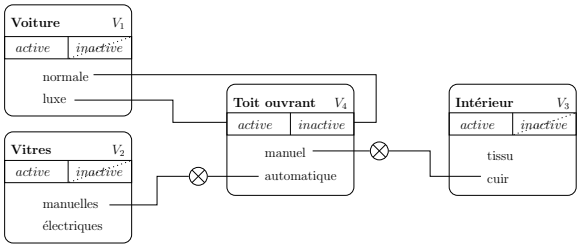


FIG. 4 – Configuration de voiture — modélisation par CSPe

$$\begin{array}{ll} C_1 : & \forall i \in \{1,2,3\}, V_i \textit{ active} \\ C_2 : & V_4 = \text{ automatique } \Leftrightarrow V_2 \neq \text{ manuelles} \\ C_3 : & V_4 = \text{ manuel } \Leftrightarrow V_3 \neq \text{ cuir} \\ C_4 : & V_1 = \text{ luxe } \Leftrightarrow V_4 \textit{ active} \end{array} \quad (2)$$

Le CSPe est ensuite traduit en un CSP : un attribut d'état devient une variable d'état, les variables des variables de base et le postulat est utilisé pour exprimer les contraintes [13]. L'équation 3 montre le CSP issu de la traduction du CSPe.

$$\begin{array}{rl} C_1 : & X_1^e = \mathit{active} \ \wedge X_2^e = \mathit{active} \ \wedge X_3^e = \mathit{active} \\ C_2 & (X_2^e = \mathit{inactive}) \vee (X_4^e = \mathit{inactive}) \vee \\ & (X_4^b = \text{ automatique } \Leftrightarrow X_2^b \neq \text{ manuelles }) \\ C_3 & (X_3^e = \mathit{inactive}) \vee (X_4^e = \mathit{inactive}) \vee \\ & (X_4^b = \text{ manuel } \Leftrightarrow X_3^b \neq \text{ cuir }) \\ C_4 : & (X_1^e = \mathit{inactive}) \\ & \vee (X_1^b = \text{ luxe } \Leftrightarrow X_4^e \ \mathit{active}) \end{array} \tag{3}$$

Après les deux valuations, aucun domaine n'est filtré. En effet, les contraintes $C_2$ et $C_3$ peuvent encore être satisfaites si l'attribut d'état de toit ouvrant est $\mathit{inactif}$. Par la limite de la cohérence d'arc, le moteur ne détecte pas la future incohérence si l'utilisateur choisit voiture à luxe. Nous obtenons donc l'espace des solutions restantes ($S$) suivant :

$$S : \begin{cases} \text{voiture} & = \{\text{normale, luxe}\} \\ \text{vitres} & = \{\text{manuelles}\} \\ \text{intérieur} & = \{\text{cuir}\} \\ \text{toit ouvrant} & = \{\text{automatique, manuel}\} \\ \text{toit ouvrant}_e & = \{\mathit{active}, \mathit{inactive}\} \end{cases} .$$

### Filtrage en CSP⋆

La figure 5 illustre la modélisation du problème sous forme d'un CSP⋆ dont l'équation 4 montre les contraintes.

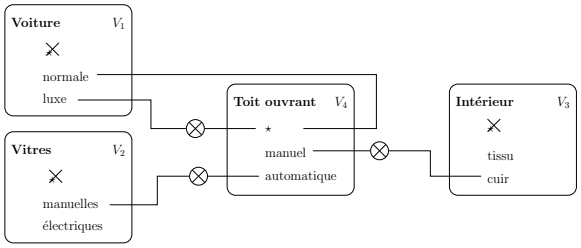


FIG. 5 – Configuration de voiture — modélisation par CSP⋆

$$\begin{array}{ll} C_1 : & V_1 \neq \star \wedge V_2 \neq \star \wedge V_3 \neq \star \\ C_2 : & V_4 = \text{ automatique } \Leftrightarrow V_2 \neq \text{ manuelles} \\ C_3 : & V_4 = \text{ manuel } \Leftrightarrow V_3 \neq \text{ cuir} \\ C_4 : & V_1 = \text{ luxe } \Leftrightarrow V_4 \neq \star \end{array} \tag{4}$$

Après les deux valuations, le domaine de toit ouvrant est réduit à ⋆. Au passage suivant de la cohérence d'arc, le moteur déduit que voiture ne peut plus être valuée à luxe ; son domaine est donc réduit à normal (seule combinaison valide). Nous obtenons donc l'espace des solutions possibles suivant :

$$S : \begin{cases} \text{voiture} & = \{\text{normale}\} \\ \text{vitres} & = \{\text{manuelles}\} \\ \text{intérieur} & = \{\text{cuir}\} \\ \text{toit ouvrant} & = \{\star\} \end{cases} .$$

### Filtrage par opérateur *Sure*

Pour pouvoir utiliser un opérateur *Sure* sur ce problème, la modélisation est légèrement différente : nous obtenons trois variables (voiture, vitres, intérieur), une seule contrainte (avec l'opérateur *Sure*) et un groupe composé d'une variable et deux contraintes. Ce groupe est actif si et seulement si la voiture est de type luxe (cf. figure 6 et équation 5).

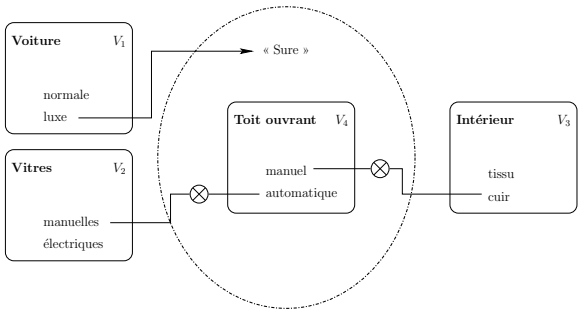


FIG. 6 – Configuration de voiture — modélisation avec opérateur *Sure*

$$\begin{array}{ll} C_1 : & \mathit{Sure}(V_1 = \text{ luxe }) \{ \\ & \quad \exists V_4 \in \{ \text{ manuel }, \text{ automatique } \} \\ & \quad V_4 = \text{ automatique } \Leftrightarrow V_2 \neq \text{ manuelles} \\ & \quad V_4 = \text{ manuel } \Leftrightarrow V_3 \neq \text{ cuir} \\ & \} \end{array} \tag{5}$$

Après les valuations, les domaines à l'intérieur du *Sure* sont réduits, mais comme la condition n'est pas vérifiée, aucune déduction n'est faite. L'espace des solutions restantes est donc similaire (à l'état de la variable toit ouvrant près) à celui obtenu par filtrage du CSPe :

$$S : \begin{cases} \text{voiture} & = \{\text{normale, luxe}\} \\ \text{vitres} & = \{\text{manuelles}\} \\ \text{intérieur} & = \{\text{cuir}\} \\ \text{toit ouvrant} & = \{\text{automatique, manuel}\} \end{cases} .$$

Cependant, le moteur déduit que la seule valuation pouvant mener à une incohérence est $V_1 = \text{luxe}$. Ainsi,

l'espace des valuations pouvant encore mener à une incohérence est réduit à :

$$\not S \ : \{ \text{ voiture}= \{\text{luxe}\} \quad .$$

### Synthèse

Nous pouvons ainsi déduire que l'utilisation de variables dont le domaine est augmenté d'une valeur ($\star$) signifiant l'inactivité de celle-ci procure un filtrage (lors de l'utilisation d'un filtrage équivalent à de la cohérence d'arc) plus efficace que l'utilisation d'un attribut d'état ou d'un opérateur *Sure*. Cet opérateur permet une déduction supplémentaire sur cet exemple (ensemble de non-solutions).

## 4.3 Variables numériques[2] à existence conditionnée

Dans un tel cas, il nous est difficile d'ajouter des valeurs aux domaines de variables réelles : en effet, comment ensuite pouvoir gérer une contrainte de type $x_1 = (x_2 + 5) \times x_3$ ? Une solution serait de redéfinir tous les opérateurs pour prendre en compte, en plus des opérations sur les intervalles (cf. [11]), les valeurs $\star$. Cependant, cette méthode n'amène aucune réduction de domaine lorsqu'une des variables qui composent la contrainte peut être inactive (valuée à $\star$), comme le montre l'équation 6.

$$\begin{array}{ll} \text{Soit le CSP numérique :} & x \in \{\star, [0,10]\} \\ & y \in \{\star, [2,13]\} \\ & C \ : \ x = y + 1 \\ \text{Après filtrage, l'espace de solutions devient :} & \\ S : \left\{ \begin{array}{l} x : \{ \star \cup [0,10]\} \\ y : \{ \star \cup [2,13]\} \end{array} \right. & \end{array} \quad (6)$$

Les domaines ne sont donc pas réduits.

Nous avons donc fait le choix de nous passer de la valeur $\star$ pour les variables continues et de n'utiliser que des attributs d'état.

Cependant, nous ne pourrons pas obtenir la même qualité de filtrage qu'avec des variables discrètes dont l'inactivité est simulée par une valeur particulière du domaine. La technique de filtrage utilisée est basée sur la 2B-cohérence. Ainsi, l'exemple de la figure 7 montre un problème de configuration d'un petit meuble, qui peut être en bois ou en verre. Si le caisson est suffisamment haut, il existe une planche au milieu (de la même longueur que le caisson), mais le problème devient incohérent si le meuble est en verre et le caisson trop long. Nous avons donc le système de contraintes de l'équation 7 (équivalent à celui de la figure 4).

[2]Les variables numériques discrètes sont traitées comme les continues

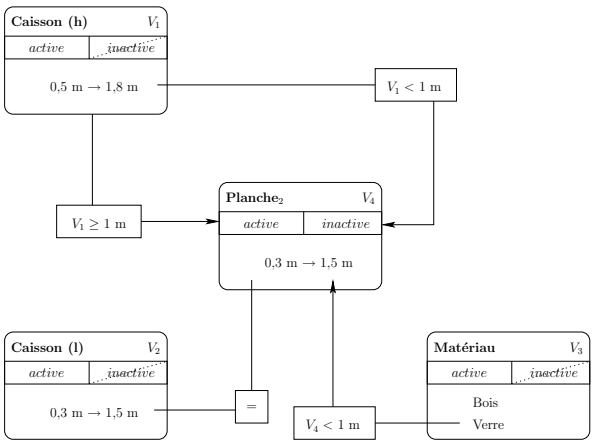


FIG. 7 – Existence conditionnée de variables numériques

$$\begin{array}{ll} C_1 : & \forall i \in \{1,2,3\}, V_i \ \textit{active} \\ C_2 : & V_2 = V_4 \\ C_3 : & V_3 = \text{ Verre } \Rightarrow V_4 < 1 \text{ m} \\ C_4 : & V_1 \geq 1 \text{ m} \Leftrightarrow V_4 \ \textit{active} \end{array} \quad (7)$$

Comme pour le CSPe discret, lorsque l'on value $V_2$ à 1,5 m et que l'on choisit du verre, le moteur devrait en déduire que $V_1$ doit être inférieure à 1 m. Cependant, le choix que nous avons fait en termes de traduction de contraintes empêche cette déduction par arc-cohérence (et même la longueur de la planche n'est pas filtrée, cette variable pouvant encore être inactive), et l'espace de solutions obtenu par filtrage est le suivant :

$$S : \left\{ \begin{array}{ll} \text{hauteur} & = [0,5; 1,8] \text{ m} \\ \text{largeur} & = 1,5 \text{ m} \\ \text{matériau} & = \{\text{verre}\} \\ \text{planche} & = [0,3; 1,5] \text{ m} \\ \text{planche}_e & = \{\textit{active}, \textit{inactive}\} \end{array} \right. .$$

De même que pour le CSPe discret, nous pouvons aussi utiliser l'opérateur *Sure* pour modéliser ce problème, et ainsi aboutir au même résultat.

## 4.4 Groupes

Étant donné que les groupes sont principalement un raccourci de notation pour l'exploitation de modèles génériques de produits, nous pourrions nous passer d'un système de gestion spécifique et utiliser les combinaisons de domaines augmentés de $\star$ ou d'attributs d'états. Cependant, cette solution est peu expressive et alourdit inutilement les contraintes.

Ainsi, il est possible de factoriser et ainsi d'utiliser un attribut d'état valable pour tout le groupe (variables *et* contraintes, cf. [4]). Les variables locales au groupe et les contraintes sont ainsi à prendre

en compte si et seulement si le groupe est activé par ailleurs. En cas de groupes inclus dans d'autres groupes, les groupes fils peuvent être activés si et seulement si l'ensemble des groupes parents est activé.

De plus, il est aussi possible d'utiliser l'opérateur *Sure*, de la même façon que pour les contraintes et les variables, lorsque l'on veut activer un groupe si et seulement si la condition est vérifiée. Les CCSP fonctionnent sur un principe similaire, mais obligent à utiliser une variable supplémentaire dont la valeur activera un groupe de variables et/ou contraintes. En revanche, les CCSP ne permettent pas de contraindre des variables appartenant à un groupe avec des variables n'y appartenant pas.

## 5 Conclusion

Nous avons présenté dans cette communication les différents mécanismes existants pour gérer des CSP dynamiques : CondCSP, CCSP, CSPe, ACSP. Chacune de ces approches possède certains avantages, que nous avons tenté d'unifier pour proposer un cadre général pour la configuration à base de contraintes. Nous proposons donc d'utiliser des variables discrètes avec des domaines augmentés d'une valeur pour symboliser l'activité, des variables continues auxquelles on associe un attribut d'état — de même que pour les groupes — et une utilisation de ces attributs pour simuler l'activité des contraintes. De plus, nous proposons un nouvel opérateur (*Sure*) permettant de remplacer des parties du problème par d'autres grâce à des méta-contraintes.

Dans le cadre de ces travaux, nous sommes en train de tester une implémentation de cette sémantique unifiée, utilisant un arbre des contraintes dont les nœuds sont les opérateurs. Cette implémentation devrait faire l'objet d'une future communication.

## Références


[1] Jérôme Amilhastre. *Représentation par automate d'ensemble de solutions de problèmes de satisfaction de contraintes*. Thèse de doctorat, Université Montpellier II, Montpellier, France, janvier 1999.

[2] Christian Bessière. Arc-Consistency for Non-Binary Dynamic CSPs. *In* Bernd Neumann, coord, *ECAI-92, Proceedings of the 10th European Conference on Artificial Intelligence*, pages 23–27, Vienne, Autriche, août 1992. John Wiley and Sons.

[3] Esther Gelle et Boi Faltings. Solving Mixed and Conditional Constraint Satisfaction Problems. *Constraints*, 8:107–141, 2003.

[4] Felix Geller et Michael Veksler. Assumption-Based Pruning in Conditional CSP. *In CP-05, Proceedings of the Eleventh International Conference on Principles and Practice of Constraint Programming*, pages 241–255, Sitges (Barcelone), Espagne, octobre 2005. Springer.

[5] Solomon W. Golomb et Leonard D. Baumert. Backtrack Programming. *Journal of the ACM*, 12(4):516–524, 1965.

[6] Alan Mackworth. Consistency in networks of relations. *Artificial intelligence*, 8:99–118, 1977.

[7] Daniel Mailharro. A classification and constraint-based framework for configuration. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 12(4):383–397, 1998.

[8] Sanjay Mittal et Brian Falkenhainer. Dynamic Constraint Satisfaction Problem. *In AAAI-90, Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 25–32, Boston, Massachusetts, USA, juillet 1990. AAAI Press / The MIT Press.

[9] Sanjay Mittal et Felix Frayman. Toward a Generic Model of Configuration Tasks. *In IJCAI-89, Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Detroit, Michigan, USA, août 1989. Morgan Kaufmann.

[10] Ugo Montanari. Networks of constraints : Fundamental properties and application to picture processing. *Information sciences*, 7:95–132, 1974.

[11] Ramon E. Moore. *Interval Analysis*. Prentice Hall, 1966.

[12] Daniel Sabin et Eugene C. Freuder. Configuration as Composite Constraint Satisfaction. *In Proceedings of the First Artificial Intelligence and Manufacturing Research Planning Workshop*, 1996.

[13] Thomas van Oudenhove de Saint Géry, Paul Gaborit, et Michel Aldanondo. A Specificity of CSP in Design: Controlling the Relevance of the Variables in the Problem. *In* Laurent Granvilliers et Barry O'Sullivan, coords, *Proceedings of the First International Workshop on Constraints and Design*, pages 34–41, Sitges (Barcelone), Espagne, octobre 2005.

[14] Gérard Verfaillie et Narendra Jussien. Constraint Solving in Uncertain and Dynamic Environments: A Survey. *Constraints*, 10(3):253–281, juillet 2005.

[15] Mathieu Veron et Michel Aldanondo. Yet Another Approach to CCSP for configuration problem. *In ECAI'00, European Conference on Artificial Intelligence, Workshop on Configuration*, pages 59–62, Berlin, Allemagne, 2000.