Cadre expérimental pour l'évaluation de méthodes hybrides en configuration interactive sous contraintes

Idir Boumbar, Élise Vareilles, Paul Gaborit, Xavier Lorca

Cadre expérimental pour l'évaluation de méthodes hybrides en configuration interactive sous contraintes

Idir. Boumbar[1, 2], Élise. Vareilles[1], Paul. Gaborit[2], Xavier. Lorca[2]

[1] ISAE Supaero, DISC, Toulouse
[2] IMT Mines Albi, CGI, Albi

[1]prenom.nom@isae-supaero.fr [2]prenom.nom@mines-albi.fr

Résumé

La configuration de systèmes est généralement formalisée sous forme d'un problème de satisfaction de contraintes. Le modèle générique d'une famille de systèmes se confond alors avec le CSP. Deux types d'approches permettent de résoudre un CSP et de définir ainsi une solution de configuration : les approches par filtrage (qui agissent de façon itérative sur le réseau de contraintes) et les approches par compilation (qui raisonnent sur l'espace des solutions). Ces deux approches ont chacune des avantages et des inconvénients, en termes de temps de compilation, temps de filtrage, compacité des solutions, etc. Dans nos travaux de thèse, nous souhaitons hybrider ces deux approches afin de tirer profit de leurs avantages et de limiter leurs défauts, sur des problèmes de configurations. Dans ce contexte, le but de cet article est d'établir un cadre expérimental qui permet de valider la pertinence d'une approche hybride des approches par filtrage et par compilation. Nous analyserons les caractéristiques des approches hybrides, pour mettre en évidence des critères comme l'évolutivité d'un CSP. Ensuite, nous reprendrons un protocole de configuration basé sur des CSP majoritairement discrets, avec des problèmes de configuration de tailles différentes, avec et sans variables continues. Nous nous proposons de l'élargir à d'autres types de CSP pour prendre en compte les spécificités des problèmes de configuration (architecture de CSP, CSP dynamique, etc.). Grâce à ce cadre, des premières hypothèses d'hybridation seront formulées.

Mots-clés

configuration hybridation compilation de connaissances propagation de contraintes

Abstract

The configuration of systems is usually formalized as a constraint satisfaction problem. The generic model of a family of systems is then merged with the CSP. Two types of approaches are used to solve a CSP and thus define a configuration solution : filtering approaches (which act iteratively on the constraint network) and compilation approaches (which reason on the solution space). Both approaches have advantages and disadvantages, in terms of compilation time, filtering time, compactness of solutions, etc. In our thesis work, we wish to hybridize these two approaches in order to take advantage of their advantages and limit their drawbacks, on configuration problems. In this context, the aim of this paper is to establish an experimental framework that allows us to validate the relevance of a hybrid approach of the filtering and compilation approaches. We will analyze the characteristics of hybrid methods, to highlight criteria such as the scalability of a CSP. Then, we will take up a configuration protocol based on mostly discrete CSPs, with different sizes of configuration problems, with and without continuous variables. We propose to extend it to other types of CSP to take into account the specificities of configuration problems (CSP architecture, dynamic CSP, etc). Thanks to this framework, first hybridization hypotheses will be formulated.

Keywords

configuration hybridization knowledge compilation constraint propagation

1 Introduction

Depuis les dernières décennies le marché doit faire face à une demande croissante en produits ou services personnalisés dans tous les secteurs. De plus, cette demande s'accompagne d'un essor des technologies du web pour le commerce qui permettent un lien plus direct entre les clients et les entreprises. Pour répondre à ces enjeux, de nombreuses entreprises proposent maintenant à leurs clients potentiels de personnaliser leurs produits ou services directement en ligne. Pour y parvenir les entreprises doivent faire face à deux problématiques : la définition de systèmes configurables et la mise en ligne de configurateurs interactifs.

La définition de systèmes configurables repose sur l'exploitation d'un catalogue de systèmes ou de familles de systèmes mais aussi des différentes connaissances métier directement ou indirectement liées aux systèmes. Pour représenter et manipuler l'ensemble de ces connaissances dont la combinatoire peut rapidement exploser, il est nécessaire de recourir à des formalismes adaptés. Une des méthodes souvent utilisées est la modélisation de la connaissance des familles de systèmes sous forme d'un problème de satisfaction de contraintes ou CSP. D'une part le formalisme CSP

parvient à bien représenter les relations entre les composants du système, son architecture, les compatibilités de ses composants, ses variantes, etc. et les choix des utilisateurs. D'autre part, il fournit de nombreuses approches pour traiter les modèles de connaissances et supporter le processus de configuration interactive.

Parmi ces approches se distinguent les approches par filtrage et les approches par compilation. Les approches par filtrage produisent des déductions sur le problème de satisfaction de contraintes afin de retirer tout au long de la configuration des valeurs qui ne seraient plus cohérentes avec le choix de l'utilisateur. Ces méthodes ont l'avantage de traiter de nombreux types de contraintes mais ne garantissent pas la *complétude* du filtrage : des valeurs incompatibles avec les choix courants peuvent ne pas être retirées. Les approches par compilation proposent de transformer le problème de satisfaction de contraintes sous la forme d'un graphe qui permet de naviguer efficacement dans l'espace des solutions. Bien qu'elles permettent de retirer les valeurs incompatibles avec les choix courants, leur complexité spatiale est théoriquement exponentielle.

Afin d'améliorer la réponse aux exigences de la configuration interactive (garantie d'un espace de solutions atteignables et conformes, temps de réponse réduit, etc.), cet article propose les premières idées d'hybridation des méthodes par filtrage et par compilation[18]. Par hybridation, nous entendons une utilisation conjointe des deux approches pour l'aide à la configuration interactive de systèmes. Ceci permettrait de tirer avantage des deux méthodes et de limiter leurs défauts.

L'objectif de cet article est de décrire un cadre expérimental qui permettrait de valider des approches hybrides de filtrage et de compilation. En effet, pour répondre à cette problématique, il est important d'identifier les critères qui mesurent les exigences de la configuration interactive. Ainsi, grâce à un protocole de configuration[19], il sera possible de valider la pertinence des approches d'hybridation qui sont évoquées dans cet article.

La section 2 pose les définitions générales de configuration, de programmation par contraintes, et de typologie de CSP. Les sections 3 et 4 présentent les deux approches utilisées pour résoudre interactivement un CSP ainsi que leurs avantages et inconvénients. La section 5 présente l'approche hybride de filtrage et de compilation. Ensuite, le cadre expérimental de l'étude ainsi que les premières pistes d'hybridation à explorer sont définies.

2 Définitions générales

La configuration interactive fait souvent appel à la notion de *problème de satisfaction de contraintes* pour représenter et modéliser toute la connaissance sur la famille de systèmes. En effet, le formalisme CSP permet de bien traduire les exigences de la configuration interactive. De plus, ce formalisme offre de nombreuses façon de modéliser la connaissance (différents types de CSP).

2.1 Besoin et exigences fonctionnelles de la configuration interactive

La configuration est une forme de conception routinière qui donne le choix à l'utilisateur des fonctionnalités et/ou composants qu'il désire dans son système. Elle se définit comme suit[13] : soit un ensemble de composants A, décrits par des propriétés et des compatibilités, d'une description désirée B du produit et d'un ensemble de critères d'évaluation C. Configurer consiste à trouver une solutions parmi les composants qui réponde à toutes les exigences soulevées par les éléments de A, B et C.

En configuration interactive [17], l'utilisateur final ou le client spécifie de façon itérative le produit désiré jusqu'à aboutir à une solution acceptable. Chaque choix réalisé par l'utilisateur durant la configuration élimine les solutions qui ne respectent pas les besoins formulés et les critères définis par l'ensemble des composants. Ainsi, l'ensemble des choix possibles se réduit jusqu'à aboutir à une solution complètement configurée.

La notion de modèle générique[1, 12] est souvent utilisée pour formaliser la connaissance sur une famille de systèmes. Le modèle générique décrit :

- la nomenclature, qui structure tous les composants du système ;
- les connaissances sur chaque composant ;
- la cardinalité de chaque composant ;
- les options possibles du système ;
- les contraintes intra/inter composants-composés du système ;
- l'ensemble des KPI ("key performance indicator") du système.

Ainsi, la configuration de certains modèles génériques peut se heurter à la difficulté de représenter des systèmes fortement combinatoire.

Un configurateur est un outil qui permet d'exploiter ce modèle générique. Sa fonction est d'aider à trouver une solution respectant toutes les spécifications et répondant aux besoins de l'utilisateur. Plus particulièrement dans le cadre de la configuration interactive[16, 17], le configurateur doit permettre à chaque itération de :

- visualiser les choix déjà formulés jusqu'à cette étape ;
- formuler des nouveaux choix sur le système parmi ceux respectant le modèle générique ;
- évaluer le système configuré relativement à des indicateurs de performance (coût, délai de livraison).

Enfin le configurateur doit assurer que les choix proposés à l'utilisateur mènent bien à un système réalisable et doit garantir des temps de réponses/traitements.

2.2 Problème de satisfaction de contraintes

L'une des méthodes couramment utilisées par les configurateurs pour décrire un modèle générique est le problèmes de satisfaction de contraintes (CSP : *Constraint Satisfaction Problem*). Un CSP [12] est un triplet $P = \{X, D, C\}$ où X est un ensemble de variables, D un ensemble de domaines et C un ensemble de contraintes portant sur les variables de

X.
Une *assignation* d est un ensemble d'affectations de variables de X. Une assignation est dite *complète* (resp. *partielle*) si elle concerne toutes les (resp. une partie des) variables de X. Une *solution* du CSP est définie comme une assignation complète des variables X qui satisfait toutes les contraintes de C. L'ensemble des solutions est noté S.
Dans un CSP $P = \{X, D, C\}$, une valeur $v \in D_{x_i}$ est globalement cohérente inverse[4] (GIC) ssi il existe $s \in S$ une solution de P telle que $s(x_i) = v$ dans l'assignation de s. Un CSP $\{X, D, C\}$ est globalement cohérent inverse (GIC) ssi les valeurs de tous ses domaines sont GIC.
Ainsi le modèle générique d'un système peut être décrit grâce à un CSP : les caractéristiques des composants génériques sont représentées par des variables, chaque variante d'une caractéristique est représentée par un élément du domaine de sa variable et les exigences et compatibilités entre composants sont représentées par des contraintes. Ensuite, l'utilisation du CSP permet de mener une activité de configuration : la spécification des besoins se traduit par des assignations particulières des variables et le rôle du configurateur est de maintenir le CSP globalement cohérent inverse à chaque assignation.

2.3 Typologie de CSP

En fonction du système, il existe différentes façons de formaliser la connaissance sous forme de CSP. Les variables peuvent être de différents types (booléennes, entières, réelles, ...). Les domaines peuvent adopter des formes différentes (intervalle, collection de valeurs, union d'intervalles...)[12, 17]. Les contraintes peuvent être formulées de différents types (formules logiques, formules mathématiques, tables de combinaisons de valeurs autorisées ou interdites, contraintes aux bornes, ...). De plus, les contraintes peuvent être distinguées de deux façons :

- compatibilité : pour exprimer les relations entre les variables et formaliser l'espace de solutions ;
- activation : pour modifier l'espace de solution par l'ajout explicite de variables, domaines et contraintes.

Plusieurs types de CSP différents doivent ainsi être combinés pour formaliser les connaissances et construire un modèle générique de famille de systèmes :

- les CSP discrets et continus pour formaliser les composants de la nomenclature ;
- les CSP génératifs pour l'aspect hiérarchique de la nomenclature ;
- les CSP dynamiques pour représenter les options ;
- les CSP continus pour évaluer les critères de performance.

3 Méthodes par filtrage

Les méthodes par filtrage filtrent le CSP pour retirer du domaine de chaque variable, les valeurs qui ne satisfont pas les contraintes. On utilise ainsi la notion de propagation de contrainte sur le domaine d'une variable. Pour retirer d'un domaine un élément qui ne satisfait pas une contrainte, il est suffisant de vérifier qu'il ne satisfait pas une sous-partie de cette contrainte : on dit qu'on vérifie une incohérence localement. Bien que cette approche permettent de retirer efficacement (i.e. dans un temps polynomial) des valeurs incohérentes, elle ne garantit pas que toutes les valeurs incohérentes sont supprimées. Ainsi la façon dont la contrainte est évaluée permet de définir différents niveaux de cohérence [14] :

- la cohérence d'arc ;
- la cohérence de domaine ;
- la cohérence d'intervalle.

Le filtrage de cohérence locale regroupe les méthodes qui assurent un niveau de cohérence local : cohérence d'arc, d'intervalle et de domaine. Elles ont l'avantage d'être rapides et donc de garantir l'interaction avec l'utilisateur. De plus, elles permettent de traiter de nombreux types de CSP(continus, dynamiques etc.). Cependant l'utilisation de niveau local ne garantit pas le retrait de toutes les non solutions : ils ne garantissent pas la cohérence globale inverse (GIC). Par exemple, pour établir la cohérence d'arc , des algorithmes performants existent comme AC–7[5] ou encore AC2001[6]. Des algorithmes comme [8, 2] permettent d'assurer une cohérence d'arc sur des contraintes de tables. Des algorithmes de Box-cohérence[3] permettent de traiter les variables à domaines continus en travaillant sur la cohérence d'intervalle. Le filtrage de cohérence globale regroupe les algorithmes qui garantissent la cohérence globale inverse (GIC). On peut par exemple citer l'algorithme GIC4[4] qui réalise un filtrage sur les domaines et vérifie que les valeurs non filtrées permettent d'aboutir à une solution. Cependant, pour assurer la GIC cet algorithme peut nécessiter plus de temps pour le traitement ce qui peut nuire à l'interaction avec l'utilisateur.
Dans le cadre de la configuration interactive, les méthodes de filtrage permettent ainsi de propager les choix de l'utilisateur et de lui permettre de faire des choix qui aboutissent à une solution. Toutefois, le recours à un filtrage de cohérence locale peut conduire à conserver dans le CSP des valeurs incohérentes globalement, bien qu'elles soient cohérentes à un certain niveau. Ceci pouvant mener l'utilisateur à construire une solution inexistante durant le processus de configuration et à retourner en arrière pour corriger ses choix.

• Avantages Raisonnement sur tous types de problèmes de configuration (discret, continu ou mixte, etc.)
• Inconvénients Ne garantit toujours pas la GIC. Possible utilisation du mécanisme de retour en arrière pour rétablir la cohérence.

TABLE 1 – Méthodes par filtrage

4 Méthodes par compilation

Les méthodes par compilation [11, 10] consistent à transformer le CSP en compilant ses contraintes dans une structure de données capable de les représenter de façon efficace.

Ensuite la configuration consiste à permettre à l'utilisateur de naviguer dans l'espace des solutions. La phase de compilation qui se déroule avant la phase de configuration peut se ramener à une recherche exhaustive de toutes les solutions du CSP. De ce fait, elle peut occuper un temps significatif et nécessiter un espace important pour stocker le résultat de compilation.
L'aspect positif de cette approche est qu'il permet de garantir la cohérence globale inverse. De plus cette méthode s'affranchit d'un temps de calcul nécessaire pour filtrer les domaines à chaque assignation de variable contrairement à la méthode de filtrage.
L'utilisation de diagrammes de décision multivaluées (Multivalued Decision Diagram - MDD)[15] est particulièrement intéressante pour la compilation de solutions. En effet, elle fournit des outils de traitement en temps polynomial qui permettent de garantir l'interaction durant la configuration. Néanmoins puisque la compilation du CSP s'effectue en amont de la session de configuration, il n'est pas possible de traiter des problèmes à structure dynamique.

• Avantages : Garantie de parvenir à une solution sans retour en arrière. • Inconvénients : Impossibilité de compiler tous les types de problèmes de configuration. Explosion spatiale dans le pire des cas.

TABLE 2 – Méthodes par compilation

5 Discussions et premières pistes d'hybridation

Les approches par filtrage et les approches par compilation sont toutes deux utilisées en configuration interactive (voir sections 3 et 4). Cependant aucune des deux approches seules ne parvient à répondre complètement aux exigences de celle-ci. D'une part, la garantie de la cohérence globale inverse et l'interactivité ne peuvent être garanties que sur des problèmes de configuration discrets par les approches par compilation. Les approches par filtrage parviennent à traiter des problèmes de configuration continus mais sans garantir la GIC. D'autres part, les approches par compilation nécessitent un certain temps de compilation avant utilisation. Ainsi elles peuvent être incompatibles pour des modèles génériques souvent mis à jour. Cette propriété que l'on définit comme *évolutivité d'un modèle* peut être un désavantage pour l'utilisation de la compilation. L'utilisation de solveurs CP pour garantir la GIC est une approche possible mais elle ne peut être envisagée en configuration. En effet, elle a une complexité temporelle trop importante pour garantir l'interaction avec l'utilisateur.
En réponse à cette problématique, une méthode utilisant conjointement les approches par filtrage et par compilation serait une solution. Cette approche hybride permettrait de garantir le caractère GIC sur des problèmes de configuration à variables continues et de limiter les défauts des deux approches classiques.
Pour établir la pertinence de l'approche hybride, il est nécessaire de valider ses performances dans un cadre expérimental. Le protocole de configuration cité dans [19] permet de simuler des sessions de configuration interactive. Il a été utilisé au préalable sur un cas d'étude du projet BR4CP [1] de configuration sur un problème statique et à variables discrètes. Néanmoins pour prouver l'intérêt de la méthode hybride, il est nécessaire de tester des cas d'étude de problèmes dynamiques et/ou à variables continues. Ensuite, il faut évaluer les grandeurs permettant de vérifier les exigences de la configuration interactive[19]. Ces grandeurs peuvent être regroupée selon plusieurs critères :

— critères d'interaction avec l'utilisateur : temps d'exécution total, temps d'exécution de la phase de propagation ;
— critères de cohérence : nombre de valeurs supprimées dans les domaines à chaque propagation, nombre de situations d'impasse ;
— autres : temps de compilation, mémoire utilisée.

Une des premières approches d'utilisation conjointe des approches par filtrage et par compilation imaginées[18] consiste à diviser le problème de configuration en fonction des sous familles de systèmes et de compiler séparément chaque sous problème. Ensuite, il s'agit de propager les choix de l'utilisateur parmi les différents sous-systèmes grâce à des méthodes de filtrage. La figure 1 illustre un exemple de modèle générique sur lequel pourrait s'appliquer cette approche hybride.

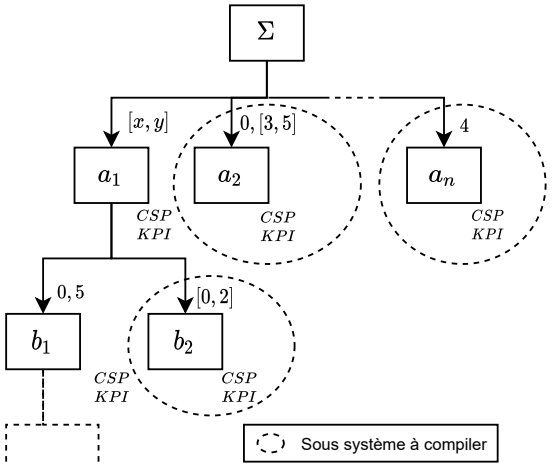

FIGURE 1 – Exemple d'utilisation d'une approche hybride sur un système configurable

Une première implémentation sera menée grâce à l'outil Choco[7]. Après avoir compilé le CSP de chaque sous système en un MDD, l'outil choco sera utilisé pour propager les contraintes durant le processus de configuration. Bien que cette approche soit analogue à l'utilisation de

1. Projet BR4CP ("Business Recommendation for Configurable Products") (ANR-11-BS02-008).

contraintes tables pour représenter les sous systèmes, l'utilisation de MDD est privilégiée car elle permet de garantir la GIC. De plus la forme compilée d'un sous système en MDD a l'avantage d'être bien plus compacte que sous la forme d'une table listant l'ensemble des tuples possibles.
En hypothèse, cette première approche permet de résoudre certaines problématiques des méthodes classiques. D'une part, cela permettrait de traiter des problèmes de configuration dynamiques grâce à la propagation de contraintes d'activation sur des sous systèmes compilés. D'autre part, cela permettrait de traiter des problèmes évolutifs en maintenant plus facilement le système par la compilation unique des sous parties du problème à mettre à jour au lieu de réaliser la compilation de tout le problème.

6 Remerciements

Des remerciements particuliers sont adressés à ISAE-Supaero, IMT Mines Albi et ANITI pour leur soutien scientifique et financier.

Références

[1] M.M. Amini, T. Coudert, É. Vareilles et M. Aldanondo : System Configuration Models : Towards a Specialization Approach. *In IFAC MIM*, 2022.

[2] G. Audemard, C. Lecoutre et M. Maamar : Segmented Tables : An Efficient Modeling Tool for Constraint Reasoning. *In ECAI*, pages 315–322, 2020.

[3] F. Benhamou, D. McAllester et P.V. Hentenryck : CLP(Intervals) Revisited, *In ILPS*, pages 1-21, 1994.

[4] C. Bessiere, H. Fargier et C. Lecoutre : Global Inverse Consistency for Interactive Constraint Satisfaction, *In Proceedings of ICPPCP*, pages 159–174, 2013.

[5] C. Bessiére, E.C. Freuder et J.-C. Regin : Using constraint metaknowledge to reduce arc consistency computation. *In Artificial Intelligence*, pages 125–148, 1999.

[6] C. Bessiere et J.-C. Régin : Refining the Basic Constraint Propagation Algorithm., *In IJCAI*, pages 309–315, 2001.

[7] C. Prud'homme, J-G. Fages et X. Lorca : Choco Solver Documentation. *choco-solver.org/docs/*, 2016

[8] J. Demeulenaere, R. Hartert, C. Lecoutre, G. Perez, L. Perron, J.-C. Régin et P. Schaus : Compact-Table : Efficiently Filtering Table Constraints with Reversible Sparse Bit-Sets, *In Proceedings of ICPPCP*, pages 207–223, 2016.

[9] A. Falkner, A. Haselböck, G. Krames, G. Schenner, H. Schreiner et R. Taupe : Solver Requirements for Interactive Configuration. *In Journal of Universal Computer Science*, pages 343–373, 2020.

[10] H. Fargier, P. Marquis et N. Schmidt : Compacité pratique des diagrammes de décision valués. Normalisation, heuristiques et expérimentations. *Revue d'Intelligence Artificielle*, pages 571–592, 2014.

[11] T. Hadzic, S. Subbarayan, R.M. Jensen, H.R. Andersen, J. Møller et H. Hulgaard : Fast backtrack-free product configuration using a precompiled solution space representation. *In Proceedings of ICPPCP*, 2004.

[12] F.Rossi, P. van Beek, T. Walsh, C. Bessiere et U. Junker : Handbook of Constraint Programming, emphElsevier, pages 27-83 et 835-873, 2006.

[13] D. Sabin et R. Weigel : Product configuration frameworks - A survey. *In : IEEE Intelligent System and their Applications*, pages 30-31, 1998.

[14] T. Schiex : Réseaux de contraintes. *Habilitation à diriger des recherches, INRA*, 2000.

[15] N. Schmidt : Compilation de préférences : application à la configuration de produit *Thèse de doctorat, Artois*, 2015.

[16] J. Tiihonen et T. Soininen : Product Configurators : Information System Support for Configurable Products,*Technical Report, Helsinki University of Technology, Laboratory of Information Processing Science* 1997.

[17] É. Vareilles : Configuration interactive et contraintes : connaissances, filtrage et extensions *Habilitation à diriger des recherches, Institut National Polytechnique de Toulouse*, 2015.

[18] É. Vareilles, H. Fargier, M. Aldanondo et P. Gaborit : ICONIC : Interactive CONstraint-based Configuration, *In 19 Th International Configuration Workshop.* pages 28-32, 2017.

[19] Y. Izza : Résolution interactive et compilation de problèmes de satisfaction de contraintes. *Mémoire de Master, Université Paul Sabatier – Toulouse 3*, 2014.