# Apprendre un CSP sans connaître son langage*

**Christian Bessiere Clément Carbonnel Areski Himeur**
Université de Montpellier, CNRS, LIRMM, Montpellier, France
{bessiere, clement.carbonnel, areski.himeur}@lirmm.fr

**Résumé**

L'acquisition de contraintes consiste à apprendre un réseau de contraintes à partir d'exemples. Les systèmes d'acquisition de contraintes existants nécessitent généralement de faire des hypothèses sur le langage de contraintes du réseau. Nous proposons une méthode d'acquisition de contraintes qui apprend un réseau de contraintes à partir d'exemples de solutions et de non-solutions, sans aucune connaissance préalable du langage.

## 1 Introduction

L'un des défis majeurs dans le domaine de la programmation par contraintes est la difficulté d'exprimer un problème à l'aide de contraintes. Pour répondre à ce défi, l'acquisition de contraintes permet d'inférer automatiquement un réseau de contraintes à partir d'exemples de solutions ou de non-solutions (apprentissage passif) ou en interrogeant un oracle (apprentissage actif). Cependant, toutes les approches actuelles d'acquisition de contraintes, telles que ConAcq.1 [3, 4], ModelSeeker [1], BayesAcq [7] et Count-CP [5], requièrent des connaissances préalables sur le langage du modèle cible. Notre objectif est donc de proposer une méthode d'apprentissage passif qui ne nécessite aucune connaissance préalable sur ce langage.

**Notations et définitions.** Soit un domaine $D$. Une *contrainte* est une paire $\langle R, S\rangle$ où $R$ est une relation d'arité $r$ sur $D$ et $S$ est une séquence de $r$ variables. Dans cet article, nous considérons les relations représentées sous la forme d'une liste de tuples acceptés. Une affectation $A : X \to D$ *satisfait* une contrainte $\langle R, S\rangle$ si $A[S] \in R$. Un *réseau de contraintes* est un tuple $N = \langle X, D, C\rangle$ où $X$ est un ensemble fini de variables, $D$ un domaine fini et $C$ est un ensemble de contraintes sur $X$. Une affectation $A : X \to D$ satisfait $N$ si $A$ satisfait toutes les contraintes de $C$. $N$ est sur un langage de contraintes $\Gamma$ si $\forall\langle R, S\rangle \in C, R \in \Gamma$.

Soit un ensemble de variables $X$ et un domaine $D$. Un *exemple* est une paire $e = \langle\phi(e), b(e)\rangle$ où $\phi(e)$ est une affectation $X \to D$ et $b(e)$ est un booléen ($e$ est un exemple positif si $b(e)$ est vrai sinon $e$ est négatif). Un réseau de contraintes $N$ *accepte* un exemple $e$ ssi $\phi(e)$ satisfait $N$, sinon il le *rejette*. Un réseau $N$ est *cohérent* avec un exemple positif (resp. négatif) $e$ si et seulement si $N$ accepte (resp. rejette) $e$. Un réseau $N$ est cohérent avec un ensemble d'exemples $E$ ssi $N$ est cohérent avec chaque exemple de $E$.

## 2 Acquisition de langage

Parmi les langages qui permettent de modéliser un réseau cohérent avec les exemples, certains sont clairement inadéquats d'un point de vue pratique. C'est le cas des langages permettant d'avoir un réseau ayant une seule contrainte couvrant toutes les variables. Notre intuition est que le meilleur langage de contraintes est le plus simple. En première approximation de cette notion de simplicité, nous allons minimiser l'arité et le nombre de relations du langage utilisé. Cette approche nous amène à considérer le problème suivant :

**Definition 1 (Language-Free Acq)** *Étant donné un ensemble d'exemples $E$ et deux entiers naturels $k, r$, le problème* Language-Free Acq *demande s'il existe un réseau de contraintes sur un langage de taille au plus $k$ et d'arité au plus $r$ cohérent avec $E$.*

Nous avons montré que ce problème est NP-complet même pour $k = r = 1$.

## 3 Aperçu de la méthode

Étant donné un ensemble d'exemples $E$, notre objectif est de trouver un réseau de contraintes cohérent

*Résumé de *Learning Constraint Networks over Unknown Constraint Languages* [2] paru dans les actes de IJCAI 2023.

avec $E$ et minimisant $(k, r)$. Nous avons observé, théoriquement et par l'expérience, qu'augmenter l'arité des relations entraîne un plus grand degré de liberté sur le choix du langage qu'augmenter leur nombre. Nous minimisons donc $k + r^2$. Nous présentons dans le paragraphe suivant un modèle pour trouver un réseau cohérent avec $k$ et $r$ fixés. Ce modèle est particulièrement efficace pour de petites valeurs de $(k, r)$. Notre méthode va donc créer un modèle pour chaque $(k, r)$ par ordre croissant de $k + r^2$ et chercher un réseau cohérent ou prouver qu'aucun existe. Le premier réseau trouvé minimise ainsi $(k, r)$.

Supposons maintenant que $(k, r)$ est fixé. Notre but est de calculer $N = \langle X, D, C \rangle$ sur un langage de taille $k$ et d'arité maximale $r$ qui est cohérent avec $E$. Nous modélisons ce problème en Max-Sat. Pour chaque relation $R$ du langage que nous cherchons à construire, nous avons trois types de variable booléenne :

— Pour chaque tuple $t$ dans $D^r$, une variable indique si $t$ appartient à $R$ ;
— Pour chaque portée $S$ dans $X^r$, une variable indique si la contrainte $\langle R, S \rangle$ appartient à $N$ ;
— Finalement, une variable indique si les deux variables précédentes sont vraies simultanément pour un couple de portée et tuple $(t, v) \in D^r \times X^r$. Cette variable est utilisée pour imposer que le réseau obtenu soit cohérent avec $E$.

Les contraintes de $N$ peuvent être retrouvées en observant les variables précédentes. Finalement, si plusieurs réseaux sont cohérents avec $E$ pour $(k, r)$ fixé, nous renvoyons celui ayant le plus grand nombre de contraintes. Notre intuition est qu'il est improbable d'observer fortuitement un faible nombre de relations applicables à de nombreuses portées et cohérentes avec les exemples. Cette intuition se vérifie très bien en pratique. Une description détaillée du modèle est disponible dans la version complète de l'article.

## 4 Résultats expérimentaux

Nous avons implémenté la méthode décrite dans la section précédente en Python. Notre programme permet de calculer un réseau de contrainte cohérent avec un ensemble d'exemples donné en générant les instances de Max-Sat correspondantes et en appelant un solveur externe (en l'occurrence Uwr-MaxSat [6]).

Nous avons évalué notre méthode sur plusieurs problèmes de référence et observé si :

— le langage appris est le langage cible ;
— le réseau appris est équivalent au réseau cible (c.-à-d. qu'ils ont exactement les mêmes solutions) ;
— le réseau appris est précisément le réseau cible (c.-à-d. avec exactement les mêmes contraintes).

Pour chacun de ces problèmes, nous faisons varier le nombre d'exemples utilisés pour former plusieurs expériences. Nous avons observé que les résultats dépendent des exemples utilisés. Ainsi, chaque expérience est exécutée 5 fois avec des exemples générés indépendamment pour obtenir une moyenne. L'ensemble $E$ est toujours composé d'exemples positifs et négatifs dans la même proportion. Nous générons les exemples positifs via un solveur avec une stratégie aléatoire de choix des valeurs et les négatifs en permutant des valeurs ou en modifiant une valeur dans une solution. Nous présentons un résumé des résultats dans la Figure 1.

| Problem | \|E\| | *Lang* | *Eq* | *Targ* | *Acc* |
|---|---|---|---|---|---|
| Sudoku | 100 | ✓ | ✗ | ✗ | 84% |
| | 200 | ✓ | ✓ | ✓ | 100% |
| Jigsaw Sudoku | 200-1400 | ✓ | ✓ | ✗ | 100% |
| Schur's Lemma | 50 | ✓ | ✗ | ✗ | 87% |
| | 800 | ✓ | ✓ | ✓ | 100% |
| Subgraph Isomorphism | 400 | ✗ | ✗ | ✗ | 98% |
| | 800 | ✗ | ✓ | ✗ | 100% |
| Golomb Ruler | 1600 | Durée maximale 12h dépassée | | | |
| | 3200 | ✗ | ✓ | ✗ | 100% |
| 8-Queens (coordonnées) | 184 | ✗ | ✗ | ✗ | 99% |

Figure 1 – $|E|$ : nombre d'exemples. *Lang* : langage cible trouvé 5 fois. *Eq* : réseau équivalent trouvé 5 fois. *Targ* : réseau cible trouvé 5 fois. *Acc* : précision moyenne mesurée sur 2 000 exemples. Les résultats pour Jigsaw dépendent des formes.

Ces résultats montrent que nous pouvons apprendre le réseau cible ou un équivalent pour une variété de problèmes, sans aucune hypothèse sur leur langage ou leur structure. Dans le cas du Golomb Ruler, le langage cible est d'arité 4. Pourtant, avec 3200 exemples, nous apprenons un réseau équivalent sur un langage avec seulement une relation ternaire. Cette relation est symétrique et appliquée à tous les triplets possibles de variables distinctes, révélant une structure cachée dans l'ensemble de solutions. Dans le cas du 8-Queens, nous n'apprenons ni le langage cible ni un réseau équivalent. Ce problème a peu de solutions, et notre protocole en requiert 50% ce qui impose un faible nombre d'exemples. De plus, le langage cible est de grande taille avec 8 relations distinctes. Nous obtenons un langage de taille 3 qui permet d'atteindre une précision de 99%.

Ces résultats soulignent des limitations suggérant de futurs travaux. Notamment, développer des notions plus sophistiquées de simplicité d'un langage et inférer des données topologiques sur le réseau cible.

## Remerciements

Travail soutenu par TAILOR – EU Horizon 2020 (GA N° 952215), ANITI – "Investing for the Future – PIA3" (GA N° ANR-19-PI3A-0004) et ISDM – AXIAUM (GA N° ANR-20 THIA-0005-01). Expériences réalisées avec la plateforme MESO@LR de l'Université de Montpellier.

## Références

[1] Nicolas Beldiceanu et Helmut Simonis : A model seeker : Extracting global constraint models from positive examples. *In* Michela Milano, éditeur : *Principles and Practice of Constraint Programming - 18th International Conference, CP 2012, Québec City, QC, Canada, October 8-12, 2012. Proceedings*, volume 7514 de *Lecture Notes in Computer Science*, pages 141–157. Springer, 2012.

[2] Christian Bessiere, Clément Carbonnel et Areski Himeur : Learning constraint networks over unknown constraint languages. *In Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, pages 1876–1883. ijcai.org, 2023.

[3] Christian Bessiere, Remi Coletta, Frédéric Koriche et Barry O'Sullivan : A sat-based version space algorithm for acquiring constraint satisfaction problems. *In* João Gama, Rui Camacho, Pavel Brazdil, Alípio Jorge et Luís Torgo, éditeurs : *Machine Learning : ECML 2005, 16th European Conference on Machine Learning, Porto, Portugal, October 3-7, 2005, Proceedings*, volume 3720 de *Lecture Notes in Computer Science*, pages 23–34. Springer, 2005.

[4] Christian Bessiere, Frédéric Koriche, Nadjib Lazaar et Barry O'Sullivan : Constraint acquisition. *Artif. Intell.*, 244:315–342, 2017.

[5] Mohit Kumar, Samuel Kolb et Tias Guns : Learning constraint programming models from data using generate-and-aggregate. *In* Christine Solnon, éditeur : *28th International Conference on Principles and Practice of Constraint Programming, CP 2022, July 31 to August 8, 2022, Haifa, Israel*, volume 235 de *LIPIcs*, pages 29 :1–29 :16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[6] Marek Piotrów : Uwrmaxsat : Efficient solver for maxsat and pseudo-boolean problems. *In 32nd IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2020, Baltimore, MD, USA, November 9-11, 2020*, pages 132–136. IEEE, 2020.

[7] Steven D. Prestwich, Eugene C. Freuder, Barry O'Sullivan et David Browne : Classifier-based constraint acquisition. *Ann. Math. Artif. Intell.*, 89(7):655–674, 2021.