Exploiter l'Entropie pour la Programmation Par Contraintes

A. Burlats[1], G. Pesant[2]

[1] UCLouvain, ICTEAM, Belgique, auguste.burlats@uclouvain.be
[2] Polytechnique Montréal, Canada, gilles.pesant@polymtl.ca

Résumé

L'introduction de la propagation de croyances dans la programmation par contraintes permet d'estimer la probabilité qu'une paire variable-valeur appartienne à une solution. Ces probabilités marginales nous permettent non seulement de développer des heuristiques de branchement mais aussi d'appliquer le concept d'entropie à la programmation par contraintes. Nous explorons comment les entropies des variables et du problème améliorent la recherche d'une solution à un problème combinatoire. Nous évaluons notre proposition sur un ensemble varié de problèmes.

Mots-clés

Programmation par contraintes, entropie, heuristique de branchement.

1 Introduction

La Programmation Par Contraintes (PPC) est une approche efficace pour la résolution de problèmes combinatoires. Elle permet en effet une forte réduction de l'espace de recherche en exploitant les contraintes du problème et les algorithmes d'inférence liés à celles-ci pour filtrer les paires variables-valeurs enfreignant les contraintes. Mais l'amplitude de cette réduction est fortement liée à l'ordre dans lequel sont fixées les variables du problème. Des heuristiques robustes et généralisables pour ordonner les variables sont donc cruciales. L'introduction de la Propagation De Croyances (PDC) en PPC permet d'estimer la probabilité qu'une paire variable-valeur fasse partie d'une solution du problème [5]. Ces probabilités marginales permettent de développer des heuristiques d'ordonnancement de variable [1] mais aussi d'appliquer le concept d'entropie à la PPC comme nous allons le développer dans cet article [1].

2 Adapter l'entropie à la PPC

2.1 Distributions marginales

Nous considérons $P = \langle X, D, C \rangle$, un problème de satisfaction de contraintes (PSC). Ce problème est défini par $X = \{x_1, x_2, \ldots, x_n\}$ est un ensemble fini de variables, $D = \{D(x_1), D(x_2), \ldots, D(x_n)\}$ est un ensemble fini de domaines et $C = \{c_1, c_2, \ldots, c_m\}$ est un ensemble fini de contraintes.

Une solution $\mathbf{s} = (v_1, v_2, \ldots, v_n)$ à P va affecter à chaque variable $x_i \in X$ une valeur v_i appartenant à son domaine $D(x_i)$ de manière à ce que toutes les contraintes de C soient respectées. Notons S^P l'ensemble de toutes les solutions de P et notons $\mathbf{s}[x]$ la valeur affectée à x dans la solution $\mathbf{s}$. On définit alors

$$\theta_x^P(v) = \frac{|\{\mathbf{s} \in S^P : \mathbf{s}[x] = v\}|}{|S^P|}$$

comme étant la proportion de solutions où x prend la valeur v [2]. Cette quantité est appelée la *marginale* de la paire variable-valeur (x, v), en référence à la probabilité marginale que x prenne la valeur v dans une solution choisie uniformément au hasard dans S. Ici nous supposons que S n'est pas vide, i.e. que P est *satisfiable*, dans le cas contraire toutes les marginales seront nulles.

2.2 Entropie

Les distributions marginales des variables permettent de quantifier l'incertitude que l'on a sur la valeur qu'une variable x devrait prendre à travers la notion d'*entropie*. Nous définissons cette *entropie* $H(x)$ à partir l'entropie de Shannon [6] :

$$H(x) = -\sum_{v \in D(x)} \theta_x(v) \log(\theta_x(v)).$$

Cette quantité non négative reflète l'incertitude sur la valeur que devrait prendre la variable x : une entropie nulle signifie que $\theta_x(v) = 1$ pour une valeur v et donc que $\theta_x(v') = 0 \ \forall v' \neq v$, i.e. que $x = v$ dans toutes les solutions ; à l'inverse, l'entropie maximale $\log(|D(x)|)$ est atteinte lorsque $\theta_x(v) = \frac{1}{|D(x)|} \ \forall v \in D(x)$ i.e. que la distribution marginale est uniforme. L'entropie normalisée (aussi appelée efficacité) de x divise son entropie par le logarithme de la cardinalité de son domaine, hormis dans le cas où son domaine ne possède qu'une valeur : son entropie (normalisée) est dans ce cas nulle. Nous définissons l'entropie du problème P comme

$$H(P) = \frac{\sum_{x \in X \,:\, |D(x)|>1} \frac{H(x)}{\log(|D(x)|)}}{|X|}.$$

Cette définition correspond à la moyenne des entropies normalisées et est donc comprise entre 0 et 1 inclus. Bien sûr notre définition de l'entropie repose sur des marginales dont nous ne connaissons pas la valeur exacte. La PDC nous permet de calculer une estimation de ces marginales [5].

1. Cet article est un résumé de celui publié dans les actes de la conférence CPAIOR 2023 [3].

2. Pour simplifier les notations, l'exposant P ne sera généralement pas noté.

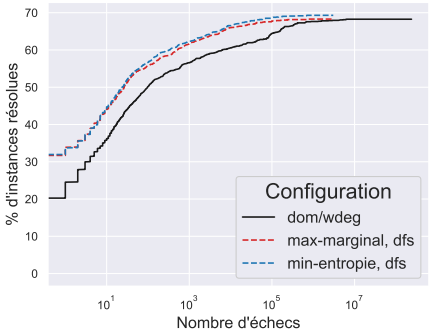

FIGURE 1 – % d'exemplaires résolus contre le nombre d'échecs pour trois heuristiques de branchement.

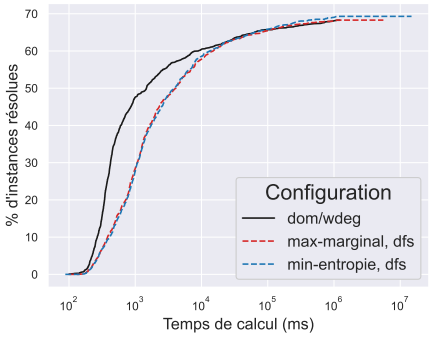

FIGURE 2 – % d'exemplaires résolus contre le temps de calcul pour trois heuristiques de branchement.

3 Exploiter l'entropie

L'introduction des notions d'entropies des variables et du problème nous permettent d'explorer plusieurs approches pour accélérer la recherche d'une solution ou pour optimiser l'usage de la PDC [3]. En particulier ici l'entropie des variables est un support puissant pour faire de meilleures décisions de branchement : plus faible est l'entropie d'une variable, plus faible est l'incertitude concernant la valeur que cette variable devrait prendre. Il est donc judicieux de la fixer en priorité. Nous proposons l'heuristique d'ordonnancement de variable *min-entropie* qui sélectionne la variable présentant la plus faible entropie et lui affecte en priorité la valeur de son domaine avec la marginale la plus élevée. Il faut noter que, dans le cas où les distributions marginales sont uniformes, la variable avec la plus faible entropie est celle avec le plus petit domaine. Ainsi, *min-entropie* peut être vue comme une généralisation de l'heuristique standard *plus petit domaine*.

4 Évaluation

Dans cette section nous cherchons à évaluer la qualité de notre stratégie de recherche, *min-entropie*. Pour la situer dans l'état de l'art, nous la comparons avec l'heuristique *dom/wdeg* [2] ainsi qu'avec une autre heuristique exploitant les distributions marginales, *max-marginal* [1] qui choisit de fixer la variable présentant la marginale la plus élevée. Pour chacune des heuristiques une recherche en profondeur a été exécutée pour trouver une solution et dans le cas de *dom/wdeg* des relances ont été exécutées au cours de la recherche afin d'en améliorer les performances (la première relance se fait après 50 échecs, et cette valeur est multipliée par 2 à chaque relance).

Les heuristiques ont été évaluées sur un jeu 1407 exemplaires de XCSP3 [3] et du concours Minizinc [4]. Elles ont été exécutées sur un serveur avec deux Intel E5-2683 v4 Broadwell @ 2.1GHz. Le solveur utilisé est MiniCPBP [5], un solveur basé sur MiniCP [4] et appliquant la PDC. Chaque test disposait de 12 Go de mémoire et le temps limite a été fixé à 20 minutes.

La figure 1 compare les performances des trois heuristiques. Les résultats sont présentés sous la forme de profils de performance : chaque point d'un graphe montre la proportion d'exemplaires (en ordonnée) résolus avec un nombre d'échecs inférieur ou égal à la valeur en abscisse. La métrique utilisée est le nombre d'échecs rencontrés au cours de la recherche d'une solution. Plus ce nombre est faible plus l'heuristique est une stratégie intéressante. On peut observer que les heuristiques exploitant les distributions marginales (*min-entropie* et *max-marginal*) montrent de meilleures performances car leurs courbes croissent plus vite que celle de *dom-wdeg*. *Min-entropie* semble être légèrement meilleure que *max-marginal*. Mais ces performances sont à relativiser : la PDC implique un surcoût conséquent sur le temps de calcul, comme cela peut être vu sur la figure 2, montrant des profils de performances en utilisant le temps de calcul comme métrique. On peut y observer que sur la majorité des problèmes, *dom/wdeg* égalise ou surpasse *min-entropie* lorsque l'on mesure le temps de calcul. Malgré cela *min-entropie* est capable de résoudre plus d'exemplaires que *dom/wdeg*. Mais l'usage de la PDC n'est pas optimisé et de nombreuses pistes peuvent être explorées pour réduire son coût. Certaines, non évoquées dans cet article, exploitent l'entropie du problème $H(P)$ pour réduire le nombre d'itérations de la PDC et pour améliorer la précision des estimations des distributions marginales.

5 Conclusion

Nous avons étudié l'entropie des PSC et de leurs variables à domaine fini, rendue possible par l'estimation des distributions marginales grâce à l'application de la combinaison de la PPC et de la PDC. Nos expériences sur 1407 exemplaires de 14 problèmes différents ont montré qu'exploiter l'entropie en choisissant de fixer la variable avec la plus faible entropie à chaque étape de la recherche est une heuristique d'ordonnancement de variable intéressante. Mais le coût de la PDC est important et il faut donc optimiser cette dernière pour que cette heuristique soit intéressante pour une large variété de problèmes.

3. http://www.xcsp.org/instances/
4. https://www.minizinc.org/challenge2022/mznc2022_probs.tar.gz
5. https://github.com/PesantGilles/MiniCPBP

Références

[1] Behrouz Babaki, Bilel Omrani, and Gilles Pesant. Combinatorial Search in CP-Based Iterated Belief Propagation. In Helmut Simonis, editor, *Principles and Practice of Constraint Programming*, pages 21–36, Cham, 2020. Springer International Publishing.

[2] Frédéric Boussemart, Fred Hemery, Christophe Lecoutre, and Lakhdar Sais. Boosting systematic search by weighting constraints. In Ramón López de Mántaras and Lorenza Saitta, editors, *Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004, including Prestigious Applicants of Intelligent Systems, PAIS 2004, Valencia, Spain, August 22-27, 2004*, pages 146–150. IOS Press, 2004.

[3] Auguste Burlats and Gilles Pesant. Exploiting Entropy in Constraint Programming. In *Proc. CPAIOR*, Lecture Notes in Computer Science. Springer, 2023.

[4] L. Michel, P. Schaus, and P. Van Hentenryck. MiniCP : a Lightweight Solver for Constraint Programming. *Mathematical Programming Computation*, 13(1) :133–184, 2021.

[5] Gilles Pesant. From Support Propagation to Belief Propagation in Constraint Programming. *Journal of Artificial Intelligence Research*, 66 :123–150, 2019.

[6] Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27(3) :379–423, July 1948.