Raffiner l'heuristique CHS à l'aide de bandits*†

Mohamed Sami Cherif **Djamal Habet** **Cyril Terrioux**
Aix Marseille Univ, Université de Toulon, CNRS, LIS, Marseille, France
{mohamed-sami.cherif, djamal.habet, cyril.terrioux}@univ-amu.fr

Résumé

Récemment, une heuristique efficace, appelée Conflict History Search (CHS), a été introduite pour la résolution du problème de satisfaction de contraintes (CSP). Elle repose sur une technique d'apprentissage par renforcement appelée *Exponential Recency Weighted Average* (ERWA) pour estimer la dureté des contraintes. CHS favorise les variables qui apparaissent souvent dans les échecs récents. Le paramètre de pas utilisé dans CHS est important car il contrôle l'estimation de la dureté des contraintes. Dans cet article, nous envisageons un raffinement de ce paramètre à l'aide d'un bandit manchot. Le bandit sélectionne une valeur appropriée de ce paramètre lors des redémarrages effectués par l'algorithme de recherche. Chaque bras correspond à l'heuristique CHS avec une valeur donnée pour le paramètre de pas et est récompensé selon sa capacité à mener une recherche efficace. Une phase d'entraînement est introduite en amont de la recherche pour aider le bandit à choisir un bras pertinent. L'évaluation expérimentale montre que cette approche conduit à des améliorations significatives.

1 Contexte

Une instance CSP est définie par la donnée d'un triplet (X, D, C), où $X = \{x_1, \ldots, x_n\}$ est un ensemble de n variables, $D = \{d_{x_1}, \ldots, d_{x_n}\}$ est un ensemble de domaines finis de taille au plus d, et $C = \{c_1, \ldots, c_e\}$ est un ensemble de e contraintes. Chaque contrainte c_i est un couple $(S(c_i), R(c_i))$, où $S(c_i) = \{x_{i_1}, \ldots, x_{i_k}\} \subseteq X$ définit la *portée* de c_i, et $R(c_i) \subseteq d_{x_{i_1}} \times \cdots \times d_{x_{i_k}}$ est une *relation de compatibilité*. Une affectation d'un sous-ensemble de X est dite *localement cohérente* si toutes les contraintes couvertes par ce sous-ensemble sont satisfaites. Une *solution* est une affectation localement cohérente de toutes les variables. Déterminer si une instance CSP possède une solution est NP-complet.

Dans ce contexte, l'heuristique de choix de variables permet de désigner la prochaine variable à instancier. Elle joue un rôle important dans la résolution des instances CSP. Son influence sur l'efficacité de la recherche est souvent considérable. Dans la littérature, de nombreuses heuristiques ont été proposées. Les heuristiques dynamiques et adaptatives (par exemple [2, 6, 8]), conduisent généralement aux meilleurs résultats.

Dans ce registre, l'heuristique CHS [4] maintient pour chaque contrainte c_i un score $q(c_i)$ (initialement nul). Lorsque c_i conduit à un échec, $q(c_i)$ est mis à jour avec la formule $q(c_i) = (1-\alpha) \times q(c_i) + \alpha \times r(c_i)$, dérivée d'ERWA. Le paramètre $0 < \alpha < 1$ (dit *paramètre de pas*) définit l'importance donnée à l'ancienne valeur de $q(c_i)$ par rapport à la valeur $r(c_i)$ de la récompense. Sa valeur, initialisée à une valeur α_0 donnée, décroit au cours du temps par pas de 10^{-6} jusqu'à un seuil fixé à $0,06$. La valeur de $r(c_i)$ dépend des échecs rencontrés récemment au niveau de c_i et est définie par $r(c_i) = \frac{1}{\#Conflicts - Conflict(c_i)+1}$. Initialisé à 0, $\#Conflicts$ est le nombre d'échecs rencontrés depuis le début de la recherche tandis que $Conflict(c_i)$ (initialisé à 0) mémorise, pour chaque contrainte c_i, la valeur qu'avait $\#Conflicts$ lors du dernier échec rencontré grâce à c_i. À chaque conflit, $\#Conflicts$ est incrémenté de 1. CHS choisit, comme prochaine variable, celle qui a le plus grand score chv avec $chv(x_j) = \frac{\sum_{c_i \in C \mid x_j \in S(c_i) \wedge |Uvars(c_i)|>1} (q(c_i)+\delta)}{|D_j|}$ où $Uvars(c_i)$ désigne l'ensemble des variables non instanciées de $S(c_i)$. Le paramètre δ permet de débuter la recherche avec des scores initiaux reflétant le nombre de contraintes dans lesquelles apparaît chaque variable. CHS privilégie donc les variables ayant un petit domaine et intervenant dans des contraintes qui sont à l'origine d'échecs récents et récurrents. Enfin, lors de chaque redémarrage, la valeur de α est réinitialisée à α_0 et les valeurs des $q(c_i)$ sont lissées suivant la formule $q(c_i) = q(c_i) \times 0,995^{\#Conflicts - Conflict(c_i)}$.

*Ce papier est un résumé de [3].

†Ce travail est soutenu par l'Agence Nationale de la Recherche dans le cadre du projet DEMOGRAPH (ANR-16-CE40-0028).

2 Raffiner CHS via un bandit manchot

Les résultats expérimentaux de CHS montrent que certaines instances ne sont résolues que pour certaines valeurs de α_0. Afin de gagner en efficacité, nous exploitons un bandit manchot dont chacun des K bras correspond à l'heuristique CHS pour une valeur donnée de α_0. À chaque redémarrage de l'algorithme de résolution, le bandit choisit un de ses bras et donc désigne la variante de CHS à utiliser jusqu'au prochain redémarrage. Ce choix est fait via la politique Upper Confidence Bound (UCB1) [1]. Il repose sur les récompenses attribuées à la fin de chaque relance pour estimer la performance de l'algorithme de résolution par rapport à l'heuristique de recherche employée.

Soit nc_t le nombre de conflits rencontrés durant la relance t. On note p_j le ratio de variables non instanciées lors du j-ème conflit. La fonction de récompense est calculée, à la fin de la relance, selon la formule $R_t(i) = \frac{\sum_{j=1}^{nc_t} p_j}{nc_t}$. Ainsi, cette fonction de récompense consiste en une moyenne des ratios de variables non instanciées rencontrées lors des conflits, évaluant la capacité de l'heuristique à identifier rapidement les instanciations incohérentes. En d'autres termes, moins il y a de variables instanciées, plus la valeur de la récompense sera élevée pour l'heuristique correspondante.

Le nombre de redémarrage semble souvent insuffisant pour évaluer pleinement le potentiel de chaque heuristique candidate, ce qui rend difficile pour le bandit de converger rapidement vers les bras les plus adéquats. Pour cette raison, une phase d'entraînement a été ajoutée. Elle consiste à choisir successivement chaque bras un par un, comme le ferait un algorithme de type tourniquet. Afin de ne pas favoriser un bras au détriment d'un autre, la durée entre deux redémarrages est constante dans cette phase et le nombre de redémarrage est un multiple de K. Cette phase d'entraînement peut être considérée comme une étape d'initialisation des paramètres utilisés par le bandit, notamment des valeurs des récompenses et de leurs moyennes. Une fois l'entraînement terminé, UCB1 prend le relais pour sélectionner un bras parmi les heuristiques candidates. Par ailleurs, durant la phase d'entraînement, les poids des CHS sont mis à jour à chaque exécution lorsqu'un conflit survient ou lors d'un redémarrage par lissage. Enfin, la phase d'entraînement seule peut suffire à résoudre certaines instances.

3 Résultats expérimentaux

Nous comparons, sur 12 901 instances du dépôt XCSP3 [1], notre approche avec entraînement (MAB-CHS+Train) et sans (MAB-CHS) à des heuristiques de l'état de l'art (à savoir *dom/wdeg* [2], ABS [6], wdeg$^{ca.cd}$ [8] et CHS) ainsi qu'à une approche basée sur un bandit multi-heuristiques (MAB-VH [7]). La résolution s'effectue via l'algorithme MAC avec redémarrages et enregistrement de nogoods [5] avec un temps limite de 30 minutes. MAB-CHS(+Train) exploite un bandit à 9 bras. Chaque bras correspond à CHS avec α_0 qui varie de 0,1 à 0,9 par pas de 0,1.

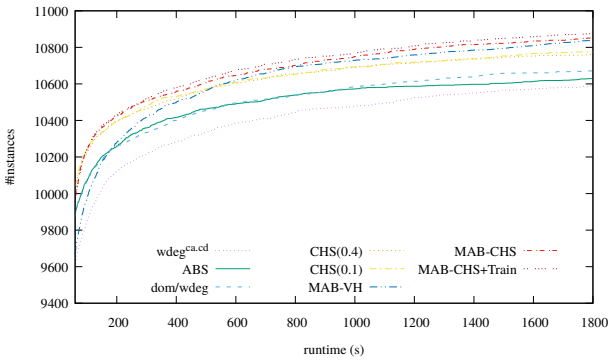

FIGURE 1 – Nombre d'instances résolues en fonction du temps écoulé (à partir de 60 s) pour chaque heuristique.

D'abord, d'après la figure 1, l'heuristique MAB-CHS+Train s'avère meilleure que MAB-CHS avec 10 875 instances résolues contre 10 853. Ensuite, comparée aux heuristiques de l'état de l'art, MAB-CHS+Train affiche un gain notable compris entre 95 et 279 instances. Enfin, MAB-CHS+Train se révèle plus efficace que MAB-VH en nombre d'instances résolues (38 instances de plus) et en temps (1,038 h contre 1,068 h).

1. http ://www.xcsp.org/series

Références

[1] Peter AUER, Nicolò CESA-BIANCHI et Paul FISCHER : Finite-time Analysis of the Multiarmed Bandit Problem. *Mach. Learn.*, 47(2-3):235–256, 2002.

[2] F. BOUSSEMART, F. HEMERY, C. LECOUTRE et L. SAÏS : Boosting Systematic Search by Weighting Constraints. *In ECAI*, pages 146–150, 2004.

[3] M. S. CHERIF, D. HABET et C. TERRIOUX : On the Refinement of Conflict History Search Through Multi-Armed Bandit. *In ICTAI*, pages 264–271, 2020.

[4] D. HABET et C. TERRIOUX : Conflict History based Search for Constraint Satisfaction Problem. *In SAC*, pages 1117–1122, 2019.

[5] C. LECOUTRE, L. SAIS, S. TABARY et V. VIDAL : Recording and Minimizing Nogoods from Restarts. *JSAT*, 1(3-4):147–167, 2007.

[6] L. MICHEL et P. Van HENTENRYCK : Activity-based search for black-box constraint programming solvers. *In CPAIOR*, pages 228–243, 2012.

[7] H. WATTEZ, F. KORICHE, C. LECOUTRE, A. PAPARRIZOU et S. TABARY : Learning Variable Ordering Heuristics with Multi-Armed Bandits and Restarts. *In ECAI*, 2020.

[8] H. WATTEZ, C. LECOUTRE, A. PAPARRIZOU et S. TABARY : Refining Constraint Weighting. *In ICTAI*, pages 71–77, 2019.