Une heuristique basée sur l'historique des conflits pour les problèmes de satisfaction de contraintes

Djamal Habet, Cyril Terrioux

Une heuristique basée sur l'historique des conflits pour les problèmes de satisfaction de contraintes*

Djamal Habet **Cyril Terrioux**

Aix Marseille Univ, Université de Toulon, CNRS, LIS, Marseille, France
{djamal.habet, cyril.terrioux}@lis-lab.fr

Résumé

L'heuristique de choix de variables est une brique importante pour les algorithmes de résolution du problème de satisfaction de contraintes (CSP). Elle a une influence souvent considérable sur l'efficacité de la recherche et permet, d'une certaine manière, d'exploiter la structure des instances. Dans cet article, nous proposons l'heuristique CHS (pour Conflict-History Search) qui est une heuristique dynamique et adaptative pour la résolution d'instances CSP. Elle repose sur les échecs rencontrés durant la recherche et considère leur temporalité tout au long de la résolution.

Une technique d'apprentissage par renforcement est exploitée pour estimer l'évolution de la difficulté des contraintes durant la recherche. Les expérimentations réalisées sur des instances au format XCSP3 permettent de montrer que l'intégration de CHS au sein d'un solveur basé sur l'algorithme MAC s'avère pertinente, conduisant notamment à des résultats meilleurs que ceux obtenus avec des heuristiques de l'état de l'art comme dom/wdeg et ABS.

Ce papier est un résumé de [2].

1 Contexte

Une instance CSP (pour Problème de Satisfaction de Contraintes) 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 un problème 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 (comme, par exemple, dom/wdeg [1] et ABS [5]), sont généralement celles conduisant aux meilleurs résultats.

2 L'heuristique CHS

Nous proposons ici l'heuristique CHS (pour Conflict-History Search), inspirée de l'heuristique CHB [4] et introduite dans le cadre du problème SAT. L'idée est de considérer l'historique des échecs rencontrés et de favoriser les variables qui apparaissent dans des échecs récents. Aussi, les échecs sont datés et une technique d'apprentissage par renforcement (ERWA pour *exponential recency weighted average*) est exploitée pour la pondération des contraintes.

Plus précisément, CHS maintient pour chaque contrainte c_j un score $q(c_j)$ qui est initialisé à 0 au début de la recherche. Lorsque que c_j conduit à un échec en vidant le domaine d'une des variables de $S(c_j)$ par propagation, $q(c_j)$ est mis à jour avec la formule ci-dessous, dérivée d'ERWA :

$$q(c_j) = (1 - \alpha) \times q(c_j) + \alpha \times r(c_j)$$

Le paramètre $0 < \alpha < 1$ définit l'importance donnée à l'ancienne valeur de $q(c_j)$ par rapport à la valeur $r(c_j)$ de la récompense. Sa valeur, initialisée à une valeur α_0 donnée, décroit au cours du temps par pas de 10^{-6} jusqu'à une valeur minimum fixée à $0,06$. Cette décroissance permet d'accorder plus d'importance à la dernière valeur de $q(c_j)$, l'idée sous-jacente étant que

*Ce travail est soutenu par l'Agence Nationale de la Recherche dans le cadre du projet DEMOGRAPH (ANR-16-C40-0028)

la valeur de $q(c_j)$ est de plus en plus pertinente au fur et à mesure que la recherche progresse.

La valeur de la récompense dépend directement des échecs rencontrés récemment au niveau de c_j. L'objectif est de récompenser davantage les contraintes qui conduisent régulièrement à des échecs sur des courtes périodes de temps. Cette valeur est calculée ainsi :

$$r(c_j) = \frac{1}{\#Conflicts - Conflict(c_j) + 1}$$

Initialisé à 0, $\#Conflicts$ est le nombre d'échecs rencontrés depuis le début de la recherche tandis que $Conflict(c_j)$ (initialisé à 0) mémorise, pour chaque contrainte c_j, la valeur qu'avait $\#Conflicts$ lors du dernier échec rencontré grâce à c_j. À chaque mise à jour de $r(c_i)$ et de $q(c_i)$, $\#Conflicts$ est incrémenté de un.

Nous pouvons maintenant définir le score associé à chaque variable x_i :

$$chv(x_i) = \frac{\sum_{c_j \in C \mid x_i \in S(c_j) \wedge |Uvars(c_j)|>1} (q(c_j) + \delta)}{|D_i|}$$

Dans cette formule, $Uvars(c_j)$ désigne l'ensemble des variables de $S(c_j)$ qui ne sont pas encore affectées. Le paramètre δ permet de débuter la recherche avec des scores initiaux reflétant le nombre de contraintes dans lesquelles apparaît chaque variable. Sa valeur est généralement inférieure à 10^{-3} afin d'avoir très rapidement un poids négligeable par rapport aux $q(c_j)$. Au final, CHS choisit donc, comme prochaine variable, celle qui a le plus grand score chv. En procédant ainsi, CHS privilégie les variables ayant un petit domaine et intervenant dans des contraintes qui sont à l'origine d'échecs récents et récurrents.

Les redémarrages constituent également une brique importante des solveurs actuels. Ils permettent notamment de réduire les conséquences négatives de mauvais choix que pourrait faire l'heuristique de choix de variables. Lorsqu'un redémarrage survient, la valeur de $Conflict(c_j)$ est conservée. Par contre, la valeur du paramètre α est, elle, réinitialisée à la valeur α_0 à chaque redémarrage. Dans le même temps, les valeurs des $q(c_j)$ sont lissées suivant la formule suivante :

$$q(c_j) = q(c_j) \times 0,995^{\#Conflicts - Conflict(c_j)}$$

Un redémarrage permet potentiellement d'explorer une nouvelle partie de l'espace de recherche. Aussi, ce lissage permet de diminuer l'importance d'anciens échecs correspondant à d'autres parties de l'espace de recherche. Par ailleurs, la réinitialisation de α a pour but de donner l'opportunité à l'heuristique d'explorer à son plein potentiel cette nouvelle partie de l'espace de recherche.

3 Résultats expérimentaux

Pour ces expérimentations, nous considérons 10 785 instances du dépôt XCSP3 [1], incluant notamment des instances structurées et excluant les instances purement aléatoires. Nous exploitons l'algorithme MAC avec redémarrages [3] pour la résolution des instances.

Nous avons d'abord fait varier les paramètres α_0 et δ. Nous avons alors constaté que l'heuristique CHS était peu sensible aux valeurs de ces paramètres. En effet, pour les valeurs étudiées, le nombre d'instances résolues varie de 9 515 à 9 525 pour un temps d'exécution cumulé allant de 491,7 h à 498,2 h. Nous avons également pu observer l'apport du lissage et de la réinitialisation de α à chaque redémarrage, la désactivation de l'une ou l'autre de ces fonctionnalités entraînant une dégradation de performances (jusqu'à 47 instances résolues en moins).

Enfin, comparée aux heuristiques de l'état de l'art (voir la table 1), CHS permet à MAC de résoudre plus d'instances qu'avec dom/wdeg ou ABS tout en requérant moins de temps.

Solveur	#instances	temps (h)
MAC+CHS	**9,525**	**493.41**
MAC+*dom/wdeg*	9,501	507.17
MAC+ABS	9,476	515.17

TABLE 1 – Nombre d'instances résolues par MAC avec dom/wdeg, ABS et CHS ($\alpha_0 = 0,4$ et $\delta = 10^{-4}$) et temps d'exécution cumulé en heures.

4 Conclusion

Nous avons proposé une nouvelle heuristique CHS qui repose sur l'historique des échecs rencontrés. Son utilisation au sein d'un solveur basé sur MAC a permis de résoudre plus d'instances que des heuristiques de l'état de l'art comme dom/wdeg ou ABS.

Références

[1] F. BOUSSEMART, F. HEMERY, C. LECOUTRE et L. SAIS : Boosting systematic search by weighting constraints. *In ECAI*, pages 146–150, 2004.

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

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

[4] J. H. LIANG, V. GANESH, P. POUPART et K. CZARNECKI : Exponential Recency Weighted Average Branching Heuristic for SAT Solvers. *In AAAI*, pages 3434–3440, 2016.

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

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